

Attribute 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's 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 Attributes</u>	69
<u>Preface</u>	85
<u>About This Manual</u>	86
<u>Additional References</u>	86
<u>How to Use the Documentation Set</u>	87
<u>Reporting Problems or Errors in Manuals</u>	88
<u>Customer Support</u>	89
<u>Cadence Online Support</u>	89
<u>Other Support Offerings</u>	89
<u>Messages</u>	90
<u>Man Pages</u>	91
<u>Command-Line Help</u>	92
<u>Getting the Syntax for a Command</u>	92
<u>Getting the Syntax for an Attribute</u>	92
<u>Searching for Attributes</u>	93
<u>Searching For Commands When You Are Unsure of the Name</u>	93
<u>Documentation Conventions</u>	94
<u>1</u>	
<u>Introduction</u>	95
<u>2</u>	
<u>General</u>	97
<u>List</u>	97
<u>Root Attributes</u>	101
<u>attribute_path</u>	101
<u>continue_on_error</u>	101
<u>cpu_runtime</u>	102
<u>dont_report_library</u>	102

Attribute Reference for Encounter RTL Compiler

<u>elapsed runtime</u>	102
<u>fail on error mesg</u>	102
<u>find fuzzy match</u>	103
<u>find inefficient threshold</u>	104
<u>find inefficient use</u>	104
<u>find help ls style</u>	104
<u>information level</u>	105
<u>limited access feature</u>	106
<u>limit lbr messages</u>	106
<u>log command error</u>	106
<u>max cpus per server</u>	107
<u>max super thread cache size</u>	107
<u>memory usage</u>	108
<u>peak memory</u>	108
<u>platform wordsize</u>	108
<u>print error info</u>	108
<u>program major version</u>	109
<u>program name</u>	110
<u>program short name</u>	110
<u>program version</u>	110
<u>real runtime</u>	110
<u>report library message summary</u>	111
<u>report tcl command error</u>	111
<u>restore history file</u>	112
<u>save history file</u>	113
<u>show command usage</u>	114
<u>show report options</u>	115
<u>source verbose</u>	115
<u>source verbose info</u>	116
<u>source suspend on error</u>	116
<u>source verbose proc</u>	117
<u>startup license</u>	117
<u>statistics db file</u>	118
<u>statistics db runtime</u>	118
<u>statistics enable power report</u>	119
<u>statistics log data</u>	119

Attribute Reference for Encounter RTL Compiler

<u>statistics run description</u>	120
<u>statistics run id</u>	121
<u>stdout log</u>	121
<u>super thread batch command</u>	122
<u>super thread cache</u>	123
<u>super thread debug directory</u>	123
<u>super thread equivalent licenses</u>	124
<u>super thread kill command</u>	125
<u>super thread peak memory</u>	126
<u>super thread rsh command</u>	126
<u>super thread servers</u>	127
<u>super thread status command</u>	128
<u>tcl result truncation length</u>	128
<u>Design Attributes</u>	129
<u>state</u>	129
<u>Message Attributes</u>	131
<u>count</u>	131
<u>help</u>	131
<u>help always visible</u>	131
<u>id</u>	131
<u>max print</u>	132
<u>print count</u>	132
<u>priority</u>	132
<u>screen max print</u>	133
<u>screen print count</u>	133
<u>severity</u>	133
<u>truncate</u>	134
<u>type</u>	134
<u>Attribute Attributes</u>	135
<u>additional help</u>	135
<u>category</u>	135
<u>check function</u>	136
<u>compute function</u>	137
<u>computed</u>	137
<u>data type</u>	138
<u>default value</u>	138

Attribute Reference for Encounter RTL Compiler

<u>help</u>	139
<u>set function</u>	139
<u>settable</u>	140
<u>units</u>	140
 3	
<u>GUI</u>	141
<u>Root Attributes</u>	142
<u>gui auto update</u>	142
<u>gui enabled</u>	142
<u>gui hv phys threshold</u>	142
<u>gui hv threshold</u>	143
<u>gui sv threshold</u>	143
<u>gui sv update</u>	143
<u>gui visible</u>	144
 4	
<u>ChipWare</u>	145
<u>List</u>	145
<u>hdl arch Attributes</u>	148
<u>location</u>	148
<u>hdl bind Attributes</u>	149
<u>avoid</u>	149
<u>constraint</u>	149
<u>operator</u>	149
<u>param association</u>	150
<u>pin association</u>	150
<u>priority</u>	150
<u>unbound oper pin</u>	151
<u>hdl comp Attributes</u>	152
<u>avoid</u>	152
<u>designware compatibility</u>	153
<u>location</u>	153
<u>obsolete</u>	153
<u>parameters</u>	154

Attribute Reference for Encounter RTL Compiler

<u>pins</u>	154
<u>report as datapath</u>	154
<u>sim model</u>	155
<u>hdl impl Attributes</u>	156
<u>avoid</u>	156
<u>language</u>	156
<u>legal</u>	156
<u>location</u>	157
<u>obsolete</u>	157
<u>pre_elab_script</u>	158
<u>preserve_techelts</u>	159
<u>priority</u>	159
<u>technology</u>	159
<u>hdl_inst Attributes</u>	160
<u>preferred_impl</u>	160
<u>hdl_label Attributes</u>	162
<u>preferred_comp</u>	162
<u>preferred_impl</u>	163
<u>hdl_lib Attributes</u>	164
<u>avoid</u>	164
<u>hdl_oper Attributes</u>	165
<u>signed</u>	165
<u>hdl_param Attribute</u>	166
<u>current_value</u>	166
<u>hdl_parameter</u>	166
<u>formula</u>	167
<u>hdl_pin Attributes</u>	168
<u>bit_width</u>	168
<u>direction</u>	168
<u>permutable_group</u>	168
<u>signed</u>	168
<u>Subdesign Attributes</u>	170
<u>candidate_impls</u>	170
<u>selected_impl</u>	171
<u>speed_grade</u>	171
<u>sub_arch</u>	172

<u>user speed grade</u>	172
5		
<u>Library</u>	173
<u>List</u>	173
<u>Library Attributes</u>	180
<u>cap scale in fF</u>	180
<u>default power rail</u>	180
<u>default wireload</u>	180
<u>files</u>	181
<u>input threshold pct fall</u>	181
<u>input threshold pct rise</u>	182
<u>leakage power scale in nw</u>	182
<u>liberty attributes</u>	182
<u>output threshold pct fall</u>	183
<u>output threshold pct rise</u>	183
<u>power rails</u>	184
<u>slew derate from library</u>	184
<u>slew lower threshold pct fall</u>	185
<u>slew lower threshold pct rise</u>	185
<u>slew upper threshold pct fall</u>	185
<u>slew upper threshold pct rise</u>	186
<u>time scale in ps</u>	186
<u>usable comb cells</u>	186
<u>usable seq cells</u>	187
<u>usable timing models</u>	187
<u>version</u>	187
<u>Libcell Attributes</u>	188
<u>adder</u>	188
<u>area</u>	188
<u>area multiplier</u>	189
<u>async clear</u>	189
<u>async preset</u>	189
<u>avoid</u>	190
<u>buffer</u>	190

Attribute Reference for Encounter RTL Compiler

<u>cell_bitwidth</u>	191
<u>cell_delay_multiplier</u>	191
<u>cell_internal_power</u>	192
<u>cell_leakage_power</u>	192
<u>cell_min_delay_multiplier</u>	193
<u>clock</u>	193
<u>clock_gating_integrated_cell</u>	194
<u>combinational</u>	194
<u>congestion_avoid</u>	194
<u>constraint_multiplier</u>	195
<u>data_pins</u>	195
<u>direction</u>	195
<u>flop</u>	196
<u>ground_direction</u>	196
<u>height</u>	197
<u>instance_probability</u>	197
<u>inverter</u>	197
<u>is_always_on</u>	198
<u>is_clock_gating_cell</u>	198
<u>is_isolation_cell</u>	198
<u>is_level_shifter</u>	198
<u>latch</u>	199
<u>latch_enable</u>	199
<u>liberty_attributes</u>	199
<u>liberty_level_shifter_type</u>	200
<u>master_slave_flop</u>	200
<u>master_slave_lssd_flop</u>	200
<u>max_ground_input_voltage</u>	200
<u>max_ground_output_voltage</u>	201
<u>max_input_voltage</u>	201
<u>max_output_voltage</u>	202
<u>min_ground_input_voltage</u>	202
<u>min_ground_output_voltage</u>	203
<u>min_input_voltage</u>	203
<u>min_output_voltage</u>	204
<u>non_seq_setup_arc</u>	204

Attribute Reference for Encounter RTL Compiler

<u>pad</u>	204
<u>power gating cell</u>	205
<u>power gating cell type</u>	205
<u>preserve</u>	206
<u>scan enable</u>	206
<u>scan in</u>	207
<u>sequential</u>	207
<u>sync clear</u>	207
<u>sync enable</u>	207
<u>sync preset</u>	208
<u>systematic probability</u>	208
<u>timing model</u>	208
<u>timing model reason</u>	209
<u>tristate</u>	209
<u>unusable reason</u>	209
<u>usable</u>	210
<u>user defined</u>	210
<u>width</u>	211
<u>Libpin Attributes</u>	212
<u>async clear phase</u>	212
<u>async preset phase</u>	212
<u>bundle</u>	212
<u>capacitance</u>	213
<u>clock gate clock pin</u>	213
<u>clock gate enable pin</u>	213
<u>clock gate obs pin</u>	213
<u>clock gate out pin</u>	214
<u>clock gate reset pin</u>	214
<u>clock gate test pin</u>	214
<u>clock phase</u>	214
<u>driver type</u>	215
<u>fall capacitance range</u>	215
<u>fanout load</u>	216
<u>function</u>	216
<u>generated clock</u>	216
<u>ground</u>	217

Attribute Reference for Encounter RTL Compiler

<u>higher_drive</u>	217
<u>incoming_timing_arcs</u>	217
<u>input</u>	217
<u>internal</u>	217
<u>is_always_on</u>	218
<u>is_iq_function</u>	218
<u>is_ign_function</u>	218
<u>isolation_cell_enable_pin</u>	218
<u>latch_enable_phase</u>	218
<u>level_shifter_enable_pin</u>	219
<u>liberty_attributes</u>	219
<u>lower_drive</u>	219
<u>max_capacitance</u>	220
<u>max_fanout</u>	220
<u>max_transition</u>	220
<u>min_capacitance</u>	221
<u>min_fanout</u>	221
<u>min_transition</u>	221
<u>non_seq_setup_arc</u>	222
<u>outgoing_timing_arcs</u>	222
<u>output</u>	222
<u>pad</u>	222
<u>polarity</u>	223
<u>power</u>	223
<u>power_gating_pin_class</u>	223
<u>power_gating_pin_phase</u>	224
<u>q_pin_of_d_pin</u>	224
<u>rise_capacitance_range</u>	225
<u>scan_enable_phase</u>	225
<u>scan_in_phase</u>	225
<u>signal_level</u>	225
<u>sync_clear_phase</u>	226
<u>sync_enable_phase</u>	226
<u>sync_preset_phase</u>	226
<u>tristate</u>	226
<u>use</u>	227

Attribute Reference for Encounter RTL Compiler

<u>user_defined</u>	227
<u>user_function</u>	227
<u>x_offset</u>	228
<u>y_offset</u>	228
<u>Libarc Attributes</u>	229
<u>enabled</u>	229
<u>from_pin</u>	229
<u>liberty_attributes</u>	229
<u>real_enabled</u>	230
<u>type</u>	230
<u>Seq Function Attributes</u>	231
<u>d_function</u>	231
<u>Operating Conditions Attributes</u>	232
<u>liberty_attributes</u>	232
<u>process</u>	232
<u>temperature</u>	233
<u>tree_type</u>	233
<u>voltage</u>	233
<u>Wireload Attributes</u>	234
<u>fanout_cap</u>	234
<u>liberty_attributes</u>	235
6	
<u>Input and Output</u>	237
<u>List</u>	237
<u>Root Attributes</u>	243
<u>bit_blasted_port_style</u>	243
<u>ccd_executable</u>	244
<u>command_log</u>	244
<u>detailed_sdc_messages</u>	244
<u>encounter_executable</u>	247
<u>error_on_lib_lef_pin_inconsistency</u>	247
<u>ets_executable</u>	248
<u>group_generate_portname_from_netname</u>	248
<u>group_instance_suffix</u>	250

Attribute Reference for Encounter RTL Compiler

<u>hdl allow inout const port connect</u>	250
<u>hdl allow instance name conflict</u>	251
<u>hdl ignore pragma names</u>	252
<u>hdl keep first module definition</u>	252
<u>hdl language</u>	252
<u>hdl link from any lib</u>	253
<u>hdl max memory address range</u>	253
<u>hdl nc compatible module linking</u>	253
<u>hdl primitive input multibit</u>	255
<u>hdl report case info</u>	256
<u>hdl search path</u>	256
<u>hdl track filename row col</u>	257
<u>hdl use current dir before hdl search path</u>	258
<u>hdl verilog defines</u>	259
<u>hdl vhdl assign width mismatch</u>	260
<u>hdl vhdl case</u>	261
<u>hdl vhdl environment</u>	261
<u>hdl vhdl lrm compliance</u>	262
<u>hdl vhdl preferred architecture</u>	262
<u>hdl vhdl range opto</u>	263
<u>hdl vhdl read version</u>	264
<u>hdl zero replicate is null</u>	264
<u>input assert one cold pragma</u>	265
<u>input assert one hot pragma</u>	265
<u>input asynchro reset blk pragma</u>	265
<u>input asynchro reset pragma</u>	266
<u>input case cover pragma</u>	267
<u>input case decode pragma</u>	267
<u>input map to mux pragma</u>	268
<u>input pragma keyword</u>	268
<u>input synchro enable blk pragma</u>	269
<u>input synchro enable pragma</u>	269
<u>input synchro reset blk pragma</u>	269
<u>input synchro reset pragma</u>	270
<u>inst prefix</u>	271
<u>lbr respect async controls priority</u>	271

Attribute Reference for Encounter RTL Compiler

<u>lec executable</u>	271
<u>lef add logical pins</u>	272
<u>lef library</u>	272
<u>lib search path</u>	273
<u>library</u>	274
<u>nc protect version</u>	275
<u>path</u>	275
<u>rc create verification directory</u>	276
<u>rc verification directory</u>	276
<u>script begin</u>	278
<u>script end</u>	278
<u>script search path</u>	279
<u>support appending libs</u>	280
<u>support multi seq elements</u>	280
<u>synthesis off command</u>	281
<u>synthesis on command</u>	281
<u>ungroup separator</u>	282
<u>use power ground pin from lef</u>	282
<u>wccd threshold percentage</u>	282
<u>wcdc clock dom comb propagation</u>	283
<u>wclp lib statetable</u>	284
<u>wlec add noblack box retime subdesign</u>	284
<u>wlec analyze abort</u>	285
<u>wlec analyze setup</u>	286
<u>wlec auto analyze</u>	286
<u>wlec compare threads</u>	287
<u>wlec cut point</u>	288
<u>wlec hier comp threshold</u>	288
<u>wlec lib statetable</u>	289
<u>wlec parallel threads</u>	289
<u>wlec set cdn synth root</u>	290
<u>wlec uniquify</u>	290
<u>wlec use lec model</u>	291
<u>write sv port wrapper</u>	292
<u>write vlog bit blast bus connections</u>	293
<u>write vlog bit blast constants</u>	294

Attribute Reference for Encounter RTL Compiler

<u>write vlog bit blast mapped ports</u>	296
<u>write vlog bit blast tech cell</u>	298
<u>write vlog convert onebit vector to scalar</u>	301
<u>write vlog declare wires</u>	302
<u>write vlog empty module for black box</u>	304
<u>write vlog empty module for logic abstract</u>	305
<u>write vlog generic gate define</u>	307
<u>write vlog line wrap limit</u>	308
<u>write vlog no negative index</u>	309
<u>write vlog preserve net name</u>	310
<u>write vlog top module first</u>	311
<u>write vlog unconnected port style</u>	312
<u>write vlog wor wand</u>	314
<u>Design Attributes</u>	317
<u>arch filename</u>	317
<u>embedded script</u>	318
<u>entity filename</u>	318
<u>hdl all filelist</u>	319
<u>hdl config name</u>	320
<u>hdl filelist</u>	321
<u>hdl user name</u>	323
<u>hdl v2001</u>	323
<u>protected</u>	324
<u>wcdc synchronizer type</u>	324
<u>Instance Attributes</u>	326
<u>hdl v2001</u>	326
<u>write vlog port association style</u>	327
<u>Pin Attributes</u>	328
<u>hdl v2001</u>	328
<u>Pgpin Attributes</u>	329
<u>hdl v2001</u>	329
<u>hdl_arch Attributes</u>	330
<u>encrypted</u>	330
<u>parameters</u>	330
<u>pins</u>	331
<u>protected</u>	331

Attribute Reference for Encounter RTL Compiler

<u>start source line</u>	331
<u>structural</u>	332
<u>verilog macros</u>	332
<u>hdl config Attributes</u>	333
<u>entity</u>	333
<u>location</u>	333
<u>hdl inst Attributes</u>	334
<u>component</u>	334
<u>hdl pack Attributes</u>	335
<u>default location</u>	335
<u>location</u>	335
<u>Port Attributes</u>	336
<u>hdl v2001</u>	336
<u>Subdesign Attributes</u>	337
<u>arch filename</u>	337
<u>embedded script</u>	338
<u>entity filename</u>	338
<u>hdl all filelist</u>	339
<u>hdl config name</u>	340
<u>hdl filelist</u>	341
<u>hdl user name</u>	342
<u>hdl v2001</u>	342
<u>protected</u>	343
<u>write vlog empty module for subdesign</u>	344
<u>Support Attributes</u>	346
<u>hdl v2001</u>	346
7	
<u>Physical</u>	349
<u>List</u>	349
<u>Root Attributes</u>	364
<u>cap table file</u>	364
<u>congestion effort</u>	364
<u>def output escape multibit</u>	365
<u>def output version</u>	365

Attribute Reference for Encounter RTL Compiler

<u>enc assign buffer</u>	366
<u>enc assign removal</u>	366
<u>enc clk gate recloning</u>	366
<u>enc cpu usage</u>	367
<u>enc design mode process</u>	367
<u>enc enable useful skew</u>	367
<u>enc force place incr</u>	368
<u>enc gzip interface files</u>	368
<u>enc launch servers</u>	369
<u>enc memory usage</u>	370
<u>enc module plan</u>	370
<u>enc postload script</u>	371
<u>enc pre place opt</u>	371
<u>enc preelexport script</u>	372
<u>enc preload script</u>	372
<u>enc save db</u>	373
<u>enc scan def file</u>	373
<u>enc temp dir</u>	373
<u>enc timing driven place</u>	374
<u>enc user constraint file</u>	375
<u>enc user mode file</u>	376
<u>force via resistance</u>	376
<u>interconnect mode</u>	377
<u>lef manufacturing grid</u>	377
<u>lef stop on error</u>	377
<u>lef units</u>	378
<u>lib lef consistency check enable</u>	378
<u>phys annotate ndr nets</u>	378
<u>phys checkout encounter license</u>	379
<u>phys fix multi height cells</u>	379
<u>phys flow effort</u>	380
<u>phys legalization enhancement</u>	380
<u>phys legalize</u>	381
<u>phys mp constraints</u>	381
<u>pqos ignore msv</u>	381
<u>pqos ignore scan chains</u>	382

Attribute Reference for Encounter RTL Compiler

<u>pqos placement effort</u>	382
<u>phys pre place iopt</u>	382
<u>phys premorph density</u>	383
<u>qos report power</u>	383
<u>qrc tech file</u>	384
<u>scale of cap per unit length</u>	384
<u>scale of res per unit length</u>	385
<u>shrink factor</u>	385
<u>use area from lef</u>	386
<u>via resistance</u>	386
Design Attributes	387
<u>aspect ratio</u>	387
<u>avoid no row libcell</u>	387
<u>blockages</u>	388
<u>def component mask shift</u>	388
<u>def extension</u>	389
<u>def file</u>	389
<u>def history</u>	389
<u>def technology</u>	390
<u>def version</u>	390
<u>die area</u>	391
<u>fplan height</u>	391
<u>fplan width</u>	391
<u>groups</u>	392
<u>init core utilization</u>	392
<u>number of routing layers</u>	393
<u>obstruction routing layer</u>	393
<u>pcells</u>	393
<u>preroute as obstruction</u>	394
<u>phys ignore nets</u>	395
<u>phys ignore special nets</u>	395
<u>physical aware mapping</u>	395
<u>physical aware restructuring</u>	396
<u>physical aware structuring</u>	396
<u>regions</u>	396
<u>sdp files</u>	397

Attribute Reference for Encounter RTL Compiler

<u>sdp_type</u>	397
<u>utilization</u>	397
<u>utilization_threshold</u>	398
<u>Pin Attributes</u>	399
<u>location_x</u>	399
<u>location_y</u>	399
<u>physical</u>	400
<u>physical_del</u>	400
<u>physical_res</u>	400
<u>placement_status</u>	401
<u>Pgpin Attributes</u>	402
<u>location_x</u>	402
<u>location_y</u>	402
<u>physical</u>	402
<u>placement_status</u>	403
<u>Net Attributes</u>	404
<u>physical_cap</u>	404
<u>Port Attributes</u>	405
<u>location_x</u>	405
<u>location_y</u>	405
<u>physical_del</u>	405
<u>physical_res</u>	406
<u>placement_status</u>	406
<u>Subdesign Attributes</u>	407
<u>fplan_height</u>	407
<u>fplan_width</u>	407
<u>physical_aware_restructuring</u>	408
<u>physical_aware_structuring</u>	408
<u>Instance Attributes</u>	409
<u>cx</u>	409
<u>cy</u>	409
<u>lx</u>	410
<u>ly</u>	410
<u>height</u>	411
<u>location_x</u>	411
<u>location_y</u>	411

Attribute Reference for Encounter RTL Compiler

<u>placement_status</u>	412
<u>skip_in_write_def</u>	412
<u>urx</u>	413
<u>ury</u>	413
<u>width</u>	413
Blockage Attributes	414
<u>boxes</u>	414
<u>component</u>	414
<u>density</u>	415
<u>has_fills</u>	415
<u>has_slots</u>	416
<u>is_exceptpgnet</u>	416
<u>is_partial</u>	416
<u>is_pushdown</u>	417
<u>is_soft</u>	417
<u>layer</u>	417
<u>location_x</u>	418
<u>location_y</u>	418
<u>mask</u>	418
<u>max_layer</u>	419
<u>min_layer</u>	419
<u>polygons</u>	419
<u>spacing</u>	420
<u>type</u>	420
<u>user_created</u>	420
<u>visible</u>	421
<u>width</u>	421
Bump Attributes	422
<u>def_name</u>	422
<u>height</u>	422
<u>location_x</u>	423
<u>location_y</u>	423
<u>model</u>	423
<u>orientation</u>	424
<u>placement_status</u>	424
<u>properties</u>	425

Attribute Reference for Encounter RTL Compiler

<u>urx</u>	425
<u>ury</u>	426
<u>weight</u>	426
<u>width</u>	427
DEF Pin Attributes	428
<u>direction</u>	428
<u>layers</u>	428
<u>location_x</u>	429
<u>location_y</u>	429
<u>net_expr</u>	429
<u>net_name</u>	430
<u>orientation</u>	430
<u>placement_status</u>	430
<u>polygons</u>	431
<u>ports</u>	431
<u>special</u>	432
<u>use</u>	432
<u>vias</u>	433
<u>visible</u>	433
Fill Attributes	434
<u>boxes</u>	434
<u>layer</u>	434
<u>mask</u>	435
<u>opc</u>	435
<u>polygons</u>	435
<u>via</u>	436
<u>via_mask</u>	436
<u>via_opc</u>	436
<u>via_points</u>	436
Gcell Attributes	437
<u>box</u>	437
<u>horizontal_remaining</u>	437
<u>instance_count</u>	438
<u>instances</u>	438
<u>pin_count</u>	439
<u>pin_density</u>	439

Attribute Reference for Encounter RTL Compiler

<u>pins</u>	439
<u>utilization</u>	440
<u>vertical remaining</u>	440
<u>Group Attributes</u>	441
<u>def name</u>	441
<u>members</u>	441
<u>properties</u>	442
<u>region</u>	442
<u>user created</u>	443
<u>Layer Attributes</u>	444
<u>cap multiplier</u>	444
<u>capacitance</u>	444
<u>cap table name</u>	445
<u>color</u>	445
<u>direction</u>	445
<u>layer index</u>	446
<u>min spacing</u>	446
<u>offset</u>	446
<u>offset x</u>	446
<u>offset y</u>	446
<u>pitch</u>	447
<u>pitch x</u>	447
<u>pitch y</u>	447
<u>resistance</u>	448
<u>type</u>	448
<u>utilization</u>	448
<u>visible</u>	449
<u>width</u>	449
<u>Nondefaultrule Attributes</u>	450
<u>from lef</u>	450
<u>hardspacing</u>	451
<u>layers</u>	451
<u>mincuts</u>	452
<u>properties</u>	452
<u>viarules</u>	453
<u>vias</u>	453

Attribute Reference for Encounter RTL Compiler

<u>Pcell Attributes</u>	454
<u>def_name</u>	454
<u>height</u>	454
<u>location_x</u>	455
<u>location_y</u>	455
<u>model</u>	456
<u>orientation</u>	456
<u>placement_status</u>	456
<u>properties</u>	457
<u>urx</u>	457
<u>ury</u>	458
<u>weight</u>	458
<u>width</u>	459
<u>Pdomain Attributes</u>	460
<u>boundary</u>	460
<u>boxes</u>	460
<u>cutouts</u>	461
<u>mingap</u>	461
<u>net</u>	461
<u>rsext</u>	462
<u>Pnet Attributes</u>	463
<u>capacitance</u>	463
<u>components</u>	463
<u>def_name</u>	464
<u>fixedbump</u>	464
<u>frequency</u>	464
<u>name</u>	464
<u>nondefaultrule</u>	465
<u>original_name</u>	465
<u>path_count</u>	465
<u>path_index</u>	465
<u>path_value</u>	466
<u>pattern</u>	466
<u>properties</u>	466
<u>rc_name</u>	467
<u>shieldnet</u>	467

Attribute Reference for Encounter RTL Compiler

<u>source</u>	467
<u>use</u>	468
<u>visible</u>	468
<u>weight</u>	468
<u>xtalk</u>	469
<u>Power Domain Attributes</u>	470
<u>box_list</u>	470
<u>disjoint_hinst_box_list</u>	470
<u>min_gaps</u>	470
<u>rs_ext</u>	471
<u>Region Attributes</u>	472
<u>boxes</u>	472
<u>def_name</u>	472
<u>derived_from_power_domain</u>	473
<u>group</u>	473
<u>location_x</u>	473
<u>location_y</u>	473
<u>properties</u>	474
<u>type</u>	474
<u>user_created</u>	474
<u>Row Attributes</u>	475
<u>height</u>	475
<u>is_horizontal</u>	475
<u>location_x</u>	476
<u>location_y</u>	476
<u>macro</u>	477
<u>orientation</u>	477
<u>user_created</u>	477
<u>visible</u>	478
<u>width</u>	478
<u>Site Attributes</u>	479
<u>class</u>	479
<u>height</u>	479
<u>symmetry</u>	479
<u>width</u>	480
<u>Slot Attributes</u>	481

Attribute Reference for Encounter RTL Compiler

<u>boxes</u>	481
<u>layer</u>	481
<u>polygons</u>	482
<u>Special Net Attributes</u>	483
<u>components</u>	483
<u>def_name</u>	483
<u>fixedbump</u>	483
<u>name</u>	484
<u>original_name</u>	484
<u>path_count</u>	484
<u>path_index</u>	485
<u>path_value</u>	485
<u>pattern</u>	485
<u>polygons</u>	486
<u>properties</u>	486
<u>rc_name</u>	486
<u>rectangles</u>	486
<u>source</u>	487
<u>style</u>	487
<u>type</u>	488
<u>use</u>	488
<u>voltage</u>	489
<u>weight</u>	489
<u>Style Attributes</u>	490
<u>index</u>	490
<u>polygon</u>	490
<u>Track Attributes</u>	491
<u>count</u>	491
<u>is_horizontal</u>	491
<u>is_used</u>	492
<u>layer</u>	492
<u>macro</u>	492
<u>mask</u>	493
<u>same_mask</u>	493
<u>start</u>	493
<u>step</u>	494

Attribute Reference for Encounter RTL Compiler

<u>Via Attributes</u>	495
<u>bottom_layer</u>	495
<u>boxes</u>	495
<u>cut_cols</u>	496
<u>cut_layer</u>	496
<u>cut_pattern</u>	496
<u>cut_rows</u>	497
<u>polygons</u>	497
<u>top_layer</u>	497
<u>viarule_name</u>	497
<u>xbottom_enclosure</u>	498
<u>xbottom_offset</u>	498
<u>xcut_size</u>	498
<u>xcut_spacing</u>	499
<u>xorigin_offset</u>	499
<u>xtop_enclosure</u>	499
<u>xtop_offset</u>	500
<u>ybottom_enclosure</u>	500
<u>ybottom_offset</u>	500
<u>ycut_size</u>	501
<u>ycut_spacing</u>	501
<u>yorigin_offset</u>	501
<u>ytop_enclosure</u>	502
<u>ytop_offset</u>	502
<u>SDP Column Attributes</u>	503
<u>flip</u>	503
<u>index</u>	503
<u>justify_by</u>	504
<u>orient</u>	504
<u>size_same</u>	504
<u>skip_value</u>	505
<u>SDP Group Attributes</u>	507
<u>hier_path</u>	507
<u>orient</u>	507
<u>origin</u>	508
<u>SDP Instance Attributes</u>	509

Attribute Reference for Encounter RTL Compiler

<u>flip</u>	509
<u>index</u>	509
<u>instance</u>	510
<u>justify_by</u>	510
<u>orient</u>	510
<u>size_fixed</u>	511
<u>skip_value</u>	511
<u>SDP Row Attributes</u>	513
<u>flip</u>	513
<u>index</u>	513
<u>justify_by</u>	514
<u>orient</u>	514
<u>size_same</u>	514
<u>skip_value</u>	515
8	
<u>Design for Manufacturing</u>	517
<u>List</u>	517
<u>Root Attributes</u>	518
<u>optimize_yield</u>	518
<u>Design Attributes</u>	519
<u>yield</u>	519
9	
<u>Constraint</u>	521
<u>List</u>	521
<u>Root Attributes</u>	527
<u>case_analysis_multi_driver_propagation</u>	527
<u>case_analysis_sequential_propagation</u>	528
<u>define_clock_with_new_cost_group</u>	529
<u>drc_first</u>	529
<u>drc_max_cap_first</u>	530
<u>drc_max_fanout_first</u>	531
<u>drc_max_trans_first</u>	533
<u>enable_break_timing_paths_by_mode</u>	534

Attribute Reference for Encounter RTL Compiler

<u>enable data check</u>	535
<u>fix min drcs</u>	536
<u>ignore scan combinational arcs</u>	536
<u>operating conditions</u>	537
<u>override library max drc</u>	537
<u>scale factor group path weights</u>	538
<u>tim ignore data check for non endpoint pins</u>	538
<u>time recovery arcs</u>	539
<u>timing disable non sequential checks</u>	539
<u>timing no path segmentation</u>	540
<u>use multi clks latency uncertainty optimize</u>	541
<u>use multi clks latency uncertainty report</u>	541
<u>wireload mode</u>	542
<u>wireload selection</u>	543
Design Attributes	545
<u>cell delay multiplier</u>	545
<u>cell min delay multiplier</u>	546
<u>force wireload</u>	547
<u>ideal seq async pins</u>	548
<u>ignore library drc</u>	549
<u>ignore library max fanout</u>	550
<u>latch borrow</u>	551
<u>latch borrow by mode</u>	552
<u>latch max borrow</u>	554
<u>latch max borrow by mode</u>	555
<u>max capacitance</u>	557
<u>max fanout</u>	558
<u>max transition</u>	559
<u>timing disable internal inout net arcs</u>	560
Mode Attributes	561
<u>default</u>	561
Instance Attributes	562
<u>cell delay multiplier</u>	562
<u>cell min delay multiplier</u>	563
<u>disabled arcs</u>	564
<u>disabled arcs by mode</u>	565

Attribute Reference for Encounter RTL Compiler

<u>hard region</u>	567
<u>latch borrow</u>	568
<u>latch borrow by mode</u>	569
<u>latch max borrow</u>	571
<u>latch max borrow by mode</u>	572
<u>Pin Attributes</u>	574
<u>break timing paths</u>	574
<u>break timing paths by mode</u>	576
<u>clock hold uncertainty</u>	578
<u>clock hold uncertainty by mode</u>	579
<u>clock network early latency</u>	580
<u>clock network early latency by mode</u>	582
<u>clock network late latency</u>	584
<u>clock network late latency by mode</u>	586
<u>clock setup uncertainty</u>	588
<u>clock setup uncertainty by mode</u>	589
<u>clock source early latency</u>	590
<u>clock source early latency by mode</u>	592
<u>clock source late latency</u>	594
<u>clock source late latency by mode</u>	596
<u>hold uncertainty by clock</u>	598
<u>ideal driver</u>	599
<u>ideal network</u>	600
<u>latch max borrow</u>	601
<u>latch max borrow by mode</u>	602
<u>network early latency by clock</u>	604
<u>network late latency by clock</u>	605
<u>setup uncertainty by clock</u>	607
<u>source early latency by clock</u>	609
<u>source late latency by clock</u>	610
<u>timing case logic value</u>	612
<u>timing case logic value by mode</u>	613
<u>Port Attributes</u>	616
<u>break timing paths</u>	616
<u>break timing paths by mode</u>	617
<u>clock hold uncertainty</u>	619

Attribute Reference for Encounter RTL Compiler

<u>clock hold uncertainty by mode</u>	620
<u>clock network early latency</u>	621
<u>clock network early latency by mode</u>	623
<u>clock network late latency</u>	625
<u>clock network late latency by mode</u>	627
<u>clock setup uncertainty</u>	629
<u>clock setup uncertainty by mode</u>	631
<u>clock source early latency</u>	632
<u>clock source early latency by mode</u>	634
<u>clock source late latency</u>	636
<u>clock source late latency by mode</u>	638
<u>external driven pin fall</u>	640
<u>external driven pin rise</u>	640
<u>external driver</u>	641
<u>external driver from pin</u>	641
<u>external driver input slew</u>	642
<u>external fanout load</u>	643
<u>external non tristate drivers</u>	643
<u>external pin cap</u>	644
<u>external resistance</u>	644
<u>external wire cap</u>	645
<u>external wire res</u>	645
<u>external wireload fanout</u>	646
<u>external wireload model</u>	646
<u>fixed slew</u>	647
<u>hold uncertainty by clock</u>	648
<u>ideal driver</u>	649
<u>ideal network</u>	649
<u>ignore external driver drc</u>	650
<u>max capacitance</u>	651
<u>max fanout</u>	652
<u>max transition</u>	653
<u>network early latency by clock</u>	654
<u>network late latency by clock</u>	655
<u>setup uncertainty by clock</u>	657
<u>source early latency by clock</u>	659

Attribute Reference for Encounter RTL Compiler

<u>source late latency by clock</u>	660
<u>timing case logic value</u>	663
<u>timing case logic value by mode</u>	664
<u>Subdesign Attributes</u>	666
<u>force wireload</u>	666
<u>hard region</u>	667
<u>Clock Attributes</u>	668
<u>clock hold uncertainty</u>	668
<u>clock network early latency</u>	669
<u>clock network late latency</u>	670
<u>clock setup uncertainty</u>	672
<u>clock source early latency</u>	673
<u>clock source late latency</u>	675
<u>comment</u>	676
<u>inverted sources</u>	677
<u>latch max borrow</u>	677
<u>non inverted sources</u>	677
<u>slew</u>	678
<u>Cost Group Attributes</u>	679
<u>weight</u>	679
<u>Exception Attributes</u>	680
<u>comment</u>	680
<u>max</u>	681
<u>user priority</u>	681
<u>External Delay Attributes</u>	682
<u>clock network latency included</u>	682
<u>clock source latency included</u>	683
10	
<u>Elaboration and Synthesis</u>	685
<u>List</u>	685
<u>Root Attributes</u>	696
<u>auto partition</u>	696
<u>auto super thread</u>	696
<u>auto ungroup</u>	697

Attribute Reference for Encounter RTL Compiler

<u>bank based multibit inferencing</u>	698
<u>boundary optimize constant hier pins</u>	699
<u>boundary optimize equal opposite hier pins</u>	700
<u>boundary optimize feedthrough hier pins</u>	701
<u>boundary optimize invert hier pins</u>	702
<u>boundary optimize invert hier pins rename nets</u>	703
<u>boundary optimize invert hier pins renaming extension</u>	704
<u>bus naming style</u>	704
<u>comb seq merge message threshold</u>	705
<u>constant prop through iso cell</u>	706
<u>control logic optimization</u>	707
<u>delete flops on preserved net</u>	707
<u>delete hier insts on preserved net</u>	708
<u>delete unloaded insts</u>	708
<u>delete unloaded seqs</u>	709
<u>derive bussed pins</u>	709
<u>dont use qbar seq pins</u>	710
<u>dp area mode</u>	710
<u>dp csa</u>	711
<u>dp postmap downsize</u>	711
<u>dp postmap upsize</u>	712
<u>dp rewriting</u>	712
<u>dp sharing</u>	713
<u>dp speculation</u>	714
<u>dp ungroup separator</u>	715
<u>driver for unloaded hier pins</u>	716
<u>exact match seq async ctrls</u>	716
<u>exact match seq sync ctrls</u>	717
<u>force merge combos into multibit cells</u>	718
<u>force merge seqs into multibit cells</u>	719
<u>gen module prefix</u>	720
<u>hdl append generic ports</u>	720
<u>hdl array naming style</u>	722
<u>hdl async set reset</u>	722
<u>hdl auto async set reset</u>	723
<u>hdl auto exec sdc scripts</u>	723

Attribute Reference for Encounter RTL Compiler

<u>hdl auto sync set reset</u>	725
<u>hdl bidirectional assign</u>	725
<u>hdl bus wire naming style</u>	726
<u>hdl case mux threshold</u>	727
<u>hdl case sensitive instances</u>	728
<u>hdl create label for unlabeled generate</u>	729
<u>hdl decimal parameter name</u>	730
<u>hdl delete transparent latch</u>	732
<u>hdl enable non default library domain binding</u>	732
<u>hdl enable proc name</u>	732
<u>hdl error on blackbox</u>	733
<u>hdl error on latch</u>	733
<u>hdl error on logic abstract</u>	734
<u>hdl error on negedge</u>	735
<u>hdl ff keep explicit feedback</u>	735
<u>hdl ff keep feedback</u>	736
<u>hdl flatten complex port</u>	737
<u>hdl generate index style</u>	739
<u>hdl generate separator</u>	742
<u>hdl index mux threshold</u>	744
<u>hdl infer unresolved from logic abstract</u>	744
<u>hdl instance array naming style</u>	745
<u>hdl interface separator</u>	747
<u>hdl latch keep feedback</u>	748
<u>hdl max loop limit</u>	749
<u>hdl max map to mux control width</u>	750
<u>hdl max recursion limit</u>	750
<u>hdl parameter naming style</u>	750
<u>hdl preserve async sr priority logic</u>	752
<u>hdl preserve dangling output nets</u>	753
<u>hdl preserve supply nets</u>	753
<u>hdl preserve sync ctrl logic</u>	754
<u>hdl preserve sync set reset</u>	755
<u>hdl preserve unused flop</u>	755
<u>hdl preserve unused latch</u>	757
<u>hdl preserve unused registers</u>	759

Attribute Reference for Encounter RTL Compiler

<u>hdl record naming style</u>	762
<u>hdl reg naming style</u>	762
<u>hdl rename cdn flop pins</u>	764
<u>hdl sv module wrapper</u>	765
<u>hdl sync set reset</u>	765
<u>hdl track module elab memory and runtime</u>	766
<u>hdl unconnected input port value</u>	767
<u>hdl undriven output port value</u>	771
<u>hdl undriven signal value</u>	773
<u>hdl use block prefix</u>	775
<u>hdl use cw first</u>	776
<u>hdl use default parameter values in design name</u>	776
<u>hdl use default parameter values in name</u>	777
<u>hdl use for generate prefix</u>	778
<u>hdl use if generate prefix</u>	779
<u>hdl use parameterized module by name</u>	781
<u>hdl use port default value</u>	783
<u>hdl use techelt first</u>	783
<u>ignore preserve in tiecell insertion</u>	784
<u>incr retime</u>	784
<u>iopt allow tiecell with inversion</u>	785
<u>iopt enable floating output check</u>	786
<u>iopt enable parallelization</u>	786
<u>iopt force constant removal</u>	787
<u>iopt remap avoided cells</u>	787
<u>iopt sequential duplication</u>	787
<u>iopt sequential resynthesis</u>	788
<u>iopt sequential resynthesis min effort</u>	788
<u>iopt temp directory</u>	789
<u>iopt ultra optimization</u>	789
<u>lbr seq in out phase opto</u>	790
<u>map drc first</u>	791
<u>map latch allow async decomp</u>	792
<u>map respect rtl clk phase</u>	792
<u>map to master slave lssd</u>	793
<u>map to multiple output gates</u>	794

Attribute Reference for Encounter RTL Compiler

<u>merge combinational hier instances</u>	794
<u>merge multibit power area based</u>	794
<u>minimize uniquify</u>	796
<u>multibit allow async phase map</u>	796
<u>multibit allow unused bits</u>	798
<u>multibit cells from different busses</u>	798
<u>multibit debug</u>	799
<u>multibit mapping effort level</u>	800
<u>multibit predefined allow unused bits</u>	801
<u>multibit prefix string</u>	801
<u>multibit preserve inferred instances</u>	802
<u>multibit preserved net check</u>	803
<u>multibit seqs instance naming style</u>	803
<u>multibit seqs members naming style</u>	805
<u>multibit short prefix string</u>	806
<u>multibit seqs name concat string</u>	807
<u>multibit skip exception check</u>	807
<u>multibit split string</u>	808
<u>multibit unused input value</u>	809
<u>optimize constant 0 flops</u>	810
<u>optimize constant 1 flops</u>	810
<u>optimize constant latches</u>	810
<u>optimize merge flops</u>	811
<u>optimize merge latches</u>	811
<u>optimize net area</u>	812
<u>optimize seq x to</u>	812
<u>propagate constant from timing model</u>	812
<u>proto feasible target</u>	813
<u>proto feasible target adjust slack pct</u>	813
<u>proto feasible target threshold</u>	814
<u>proto feasible target threshold clock pct</u>	814
<u>proto hdl</u>	815
<u>remove assigns</u>	816
<u>retime async reset</u>	817
<u>retime effort level</u>	817
<u>retime move mux loop with reg</u>	818

Attribute Reference for Encounter RTL Compiler

<u>retime optimize reset</u>	819
<u>retime reg naming suffix</u>	819
<u>retime verification flow</u>	820
<u>retiming clocks</u>	821
<u>stop at iopt state</u>	822
<u>tns opto</u>	822
<u>ultra global mapping</u>	823
<u>uniquify naming style</u>	824
<u>use compatibility based grouping</u>	825
<u>use multibit cells</u>	826
<u>use multibit combo cells</u>	827
<u>use multibit seq and tristate cells</u>	828
<u>use nextstate type only to assign sync ctrls</u>	829
<u>use scan seqs for non dft</u>	829
<u>use tiehilo for const</u>	830
<u>Design Attributes</u>	832
<u>control logic optimization</u>	832
<u>delete unloaded seqs</u>	833
<u>dp csa</u>	833
<u>dp rewriting</u>	834
<u>dp sharing</u>	835
<u>dp speculation</u>	836
<u>dp verify ok</u>	836
<u>hdl cw list</u>	837
<u>preserve</u>	838
<u>retime</u>	839
<u>undesirable libcells</u>	840
<u>Instance Attributes</u>	841
<u>dont infer multibit</u>	841
<u>dont retime</u>	842
<u>dont split multibit</u>	842
<u>dont use qbar pin</u>	843
<u>hdl proc name</u>	843
<u>infer multibit</u>	844
<u>inherited preserve</u>	845
<u>map to multibit bank label</u>	846

Attribute Reference for Encounter RTL Compiler

<u>map to multibit register</u>	847
<u>map to mux</u>	848
<u>map to register</u>	848
<u>merge combinational hier instance</u>	849
<u>multibit allow async phase map</u>	849
<u>optimize constant 0 seq</u>	852
<u>optimize constant 1 seq</u>	853
<u>optimize constant feedback seq</u>	853
<u>optimize merge seq</u>	854
<u>preserve</u>	855
<u>propagate constant from timing model</u>	856
<u>trace retime</u>	857
<u>undesirable libcells</u>	858
<u>ungroup ok</u>	858
<u>unresolved</u>	859
<u>hdl arch Attributes</u>	860
<u>blackbox</u>	860
<u>hdl error on blackbox</u>	860
<u>hdl error on latch</u>	861
<u>hdl error on logic abstract</u>	861
<u>hdl error on negedge</u>	862
<u>hdl ff keep explicit feedback</u>	862
<u>hdl ff keep feedback</u>	863
<u>hdl latch keep feedback</u>	863
<u>hdl preserve unused flop</u>	864
<u>hdl preserve unused latch</u>	865
<u>hdl preserve unused registers</u>	865
<u>ungroup</u>	866
<u>hdl block Attributes</u>	867
<u>group</u>	867
<u>hdl comp Attributes</u>	869
<u>ungroup</u>	869
<u>hdl impl Attributes</u>	870
<u>ungroup</u>	870
<u>hdl inst Attributes</u>	871
<u>ungroup</u>	871

Attribute Reference for Encounter RTL Compiler

<u>hdl_proc Attributes</u>	872
<u>group</u>	872
<u>hdl_subp Attributes</u>	875
<u>map_to_module</u>	875
<u>map_to_operator</u>	875
<u>return_port</u>	875
<u>Net Attributes</u>	876
<u>preserve</u>	876
<u>Pin Attributes</u>	878
<u>boundary_optimize_constant_hier_pins</u>	878
<u>boundary_optimize_equal_opposite_hier_pins</u>	879
<u>boundary_optimize_feedthrough_hier_pins</u>	881
<u>boundary_optimize_hier_pin_invertible</u>	882
<u>boundary_optimize_invert_hier_pins</u>	883
<u>iopt_avoid_tiecell_replacement</u>	884
<u>lssd_master_clock</u>	885
<u>preserve</u>	886
<u>prune_unused_logic</u>	887
<u>Pgpin Attributes</u>	888
<u>preserve</u>	888
<u>Port Attributes</u>	890
<u>iopt_avoid_tiecell_replacement</u>	890
<u>lssd_master_clock</u>	890
<u>Subdesign Attributes</u>	892
<u>boundary_opto</u>	892
<u>boundary_optimize_constant_hier_pins</u>	893
<u>boundary_optimize_equal_opposite_hier_pins</u>	894
<u>boundary_optimize_feedthrough_hier_pins</u>	896
<u>boundary_optimize_invert_hier_pins</u>	897
<u>control_logic_optimization</u>	899
<u>delete_unloaded_insts</u>	900
<u>delete_unloaded_seqs</u>	901
<u>dp_csa</u>	901
<u>dp_rewriting</u>	902
<u>dp_sharing</u>	903
<u>dp_speculation</u>	904

Attribute Reference for Encounter RTL Compiler

<u>dp_verify_ok</u>	904
<u>hdl_cw_list</u>	905
<u>minimize_uniquify</u>	906
<u>multibit_allow_async_phase_map</u>	907
<u>preserve</u>	909
<u>retime</u>	910
<u>retime_hard_region</u>	910
<u>user_sub_arch</u>	911
<u>ungroup_ok</u>	911
<u>Subport Attributes</u>	913
<u>boundary_optimize_hier_pin_invertible</u>	913
<u>iopt_avoid_tiecell_replacement</u>	914
<u>lssd_master_clock</u>	914
<u>Clock Attributes</u>	915
<u>clock_library_cells</u>	915
11 Analysis	917
<u>List</u>	917
<u>Root Attributes</u>	928
<u>active_operating_conditions</u>	928
<u>report_timing_show_wire_length</u>	928
<u>show_wns_in_log</u>	929
<u>Design Attributes</u>	931
<u>arch_name</u>	931
<u>area</u>	931
<u>average_net_length</u>	932
<u>cell_area</u>	932
<u>cell_count</u>	932
<u>constant_0_nets</u>	933
<u>constant_1_nets</u>	933
<u>entity_name</u>	933
<u>hdl_elab_command_params</u>	934
<u>hdl_parameters</u>	935
<u>hdl_vdp_list</u>	936

Attribute Reference for Encounter RTL Compiler

<u>language</u>	937
<u>library_name</u>	937
<u>logic_abstract</u>	937
<u>max_cap_cost</u>	938
<u>max_fanout_cost</u>	938
<u>max_trans_cost</u>	939
<u>min_cap_cost</u>	939
<u>min_fanout_cost</u>	940
<u>min_trans_cost</u>	940
<u>net_area</u>	940
<u>physical_cell_area</u>	941
<u>pin_count</u>	941
<u>seq_reason_deleted</u>	941
<u>slack</u>	942
<u>slack_by_mode</u>	943
<u>tns</u>	944
<u>total_net_length</u>	945
<u>wireload</u>	945
<u>Instance Attributes</u>	946
<u>buffer</u>	946
<u>cell_area</u>	946
<u>combinational</u>	947
<u>exceptions</u>	947
<u>flop</u>	948
<u>hdl_instantiated</u>	948
<u>hierarchical</u>	948
<u>inverted_phase</u>	948
<u>inverter</u>	949
<u>latch</u>	949
<u>libcell</u>	949
<u>loop_breaker_is_ignored_for_all_modes</u>	950
<u>module</u>	950
<u>negative_edge_clock</u>	950
<u>pin_count</u>	951
<u>primitive_function</u>	951
<u>retime_original_registers</u>	951

Attribute Reference for Encounter RTL Compiler

<u>sequential</u>	952
<u>slack</u>	952
<u>subdesign</u>	953
<u>timing case disabled arcs</u>	953
<u>timing case disabled arcs by mode</u>	954
<u>timing model</u>	955
<u>tristate</u>	955
<u>unique versions</u>	956
Constant Attributes	957
<u>pin capacitance</u>	957
<u>unique versions</u>	958
<u>wire capacitance</u>	958
<u>wire length</u>	959
<u>wire resistance</u>	959
Pin Attributes	960
<u>boundary change</u>	960
<u>capturer</u>	962
<u>causes ideal net</u>	962
<u>clock sense negative</u>	963
<u>clock sense positive</u>	964
<u>clock sense stop propagation</u>	964
<u>clock sources inverted</u>	965
<u>clock sources non inverted</u>	966
<u>connect delay</u>	966
<u>direction</u>	967
<u>drivers</u>	967
<u>endpoint</u>	967
<u>exceptions</u>	968
<u>external delays</u>	968
<u>external delays by mode</u>	969
<u>has min delay</u>	971
<u>launcher</u>	971
<u>libpin</u>	972
<u>loads</u>	972
<u>min slew</u>	972
<u>min timing arcs</u>	973

Attribute Reference for Encounter RTL Compiler

<u>net</u>	973
<u>pin capacitance</u>	974
<u>propagated clocks</u>	974
<u>propagated clocks by mode</u>	976
<u>propagated ideal network</u>	978
<u>rf slack</u>	979
<u>slack</u>	980
<u>slack by mode</u>	980
<u>slew</u>	982
<u>slew by mode</u>	982
<u>startpoint</u>	983
<u>timing arcs</u>	983
<u>timing case computed value</u>	983
<u>timing case computed value by mode</u>	985
<u>timing info</u>	987
<u>timing info favor startpoint</u>	990
<u>unique versions</u>	991
<u>wire capacitance</u>	991
<u>wire length</u>	992
<u>wire resistance</u>	992
<u>wireload model</u>	992
<u>Pgpin Attributes</u>	993
<u>direction</u>	993
<u>drivers</u>	993
<u>libpin</u>	994
<u>loads</u>	994
<u>net</u>	994
<u>unique versions</u>	994
<u>Pin Bus Attributes</u>	996
<u>bits</u>	996
<u>Net Attributes</u>	997
<u>constant</u>	997
<u>driven by supply0</u>	997
<u>driven by supply1</u>	998
<u>drivers</u>	999
<u>ideal</u>	999

Attribute Reference for Encounter RTL Compiler

<u>is net part of bus</u>	1000
<u>loads</u>	1000
<u>num drivers</u>	1000
<u>num loads</u>	1000
<u>type</u>	1001
<u>unique versions</u>	1001
Port Attributes	1002
<u>bus</u>	1002
<u>capturer</u>	1002
<u>causes ideal net</u>	1003
<u>clock sources inverted</u>	1004
<u>clock sources non inverted</u>	1004
<u>connect delay</u>	1005
<u>direction</u>	1005
<u>drivers</u>	1005
<u>endpoint</u>	1006
<u>exceptions</u>	1006
<u>external delays</u>	1007
<u>external delays by mode</u>	1007
<u>has min delay</u>	1009
<u>launcher</u>	1009
<u>loads</u>	1010
<u>min slew</u>	1010
<u>min port delay</u>	1011
<u>net</u>	1011
<u>pin capacitance</u>	1012
<u>port delay</u>	1012
<u>propagated clocks</u>	1013
<u>propagated clocks by mode</u>	1015
<u>rf slack</u>	1017
<u>slack</u>	1017
<u>slack by mode</u>	1018
<u>slew</u>	1019
<u>slew by mode</u>	1019
<u>startpoint</u>	1020
<u>timing case computed value</u>	1020

Attribute Reference for Encounter RTL Compiler

<u>timing_case_computed_value_by_mode</u>	1021
<u>timing_info</u>	1023
<u>timing_info_favor_startpoint</u>	1025
<u>unique_versions</u>	1025
<u>wire_capacitance</u>	1026
<u>wire_length</u>	1026
<u>wire_resistance</u>	1027
<u>Port Bus Attributes</u>	1028
<u>bits</u>	1028
<u>direction</u>	1028
<u>order</u>	1029
<u>Subdesign Attributes</u>	1030
<u>arch_name</u>	1030
<u>area</u>	1030
<u>cell_area</u>	1031
<u>cell_count</u>	1031
<u>constant_0_nets</u>	1031
<u>constant_1_nets</u>	1032
<u>entity_name</u>	1032
<u>hdl_elab_command_params</u>	1032
<u>hdl_parameters</u>	1033
<u>hdl_pipeline_comp</u>	1034
<u>hdl_vdp_list</u>	1034
<u>instances</u>	1035
<u>language</u>	1035
<u>library_name</u>	1036
<u>logic_abstract</u>	1036
<u>logical_hier</u>	1037
<u>net_area</u>	1037
<u>physical_cell_area</u>	1037
<u>pin_count</u>	1038
<u>wireload</u>	1038
<u>Support Attributes</u>	1039
<u>bus</u>	1039
<u>causes_ideal_net</u>	1039
<u>direction</u>	1040

Attribute Reference for Encounter RTL Compiler

<u>drivers</u>	1040
<u>loads</u>	1041
<u>net</u>	1041
<u>pin capacitance</u>	1041
<u>unique versions</u>	1042
<u>wire capacitance</u>	1042
<u>wire length</u>	1043
<u>wire resistance</u>	1043
<u>Support Bus Attributes</u>	1044
<u>bits</u>	1044
<u>direction</u>	1044
<u>order</u>	1045
<u>Timing Bin Attributes</u>	1046
<u>is sub bin</u>	1046
<u>path count</u>	1046
<u>root</u>	1047
<u>Timing Path Attributes</u>	1048
<u>bin</u>	1048
<u>end point</u>	1048
<u>exceptions</u>	1049
<u>mode</u>	1049
<u>slack</u>	1050
<u>startpoint</u>	1050
<u>Clock Attributes</u>	1051
<u>clock sense negative</u>	1051
<u>clock sense positive</u>	1052
<u>clock sense stop propagation</u>	1052
<u>divide fall</u>	1053
<u>divide period</u>	1053
<u>divide rise</u>	1054
<u>divide waveform</u>	1054
<u>exceptions</u>	1055
<u>fall</u>	1056
<u>period</u>	1056
<u>rise</u>	1056
<u>waveform</u>	1057

Attribute Reference for Encounter RTL Compiler

<u>Clock Domain Attributes</u>	1058
<u>exceptions</u>	1058
<u>Cost Group Attributes</u>	1059
<u>exceptions</u>	1059
<u>slack</u>	1060
<u>slack_by_mode</u>	1060
<u>tns</u>	1062
<u>Exception Attributes</u>	1063
<u>adjust_value</u>	1063
<u>cost_group</u>	1063
<u>delay_value</u>	1063
<u>domain</u>	1064
<u>exception_type</u>	1064
<u>from_points</u>	1065
<u>lenient</u>	1065
<u>paths</u>	1066
<u>precluded_path_adjusts</u>	1066
<u>priority</u>	1066
<u>shift_capture</u>	1067
<u>shift_launch</u>	1067
<u>through_points</u>	1067
<u>to_points</u>	1068
<u>External Delay Attributes</u>	1069
<u>clock</u>	1069
<u>clock_rise</u>	1069
<u>delay</u>	1069
<u>exceptions</u>	1070
<u>external_delay_pins</u>	1070
<u>input_delay</u>	1071
<u>level_sensitive</u>	1071
<u>12</u>	
<u>Design For Test</u>	1073
<u>List</u>	1073
<u>Root Attributes</u>	1093

Attribute Reference for Encounter RTL Compiler

<u>dft apply sdc constraints</u>	1093
<u>dft boundary cell module prefix</u>	1093
<u>dft clock waveform divide fall</u>	1094
<u>dft clock waveform divide period</u>	1094
<u>dft clock waveform divide rise</u>	1095
<u>dft clock waveform fall</u>	1095
<u>dft clock waveform period</u>	1096
<u>dft clock waveform rise</u>	1096
<u>dft fence slow speed domains</u>	1097
<u>dft generate et no testpoint file</u>	1097
<u>dft identify internal test clocks</u>	1098
<u>dft identify test signals</u>	1099
<u>dft identify top level test clocks</u>	1099
<u>dft identify xsource violations from timing models</u>	1099
<u>dft include controllable pins in abstract model</u>	1100
<u>dft include test signal outputs in abstract model</u>	1104
<u>dft jtag module name</u>	1106
<u>dft opcg block input to flop paths</u>	1107
<u>dft opcg domain blocking</u>	1107
<u>dft prefix</u>	1108
<u>dft propagate test signals from hookup pins only</u>	1108
<u>dft report empty test clocks</u>	1109
<u>dft rtl insertion</u>	1109
<u>dft scan style</u>	1110
<u>dft scanbit waveform analysis</u>	1110
<u>dft shift register identification mode</u>	1111
<u>dft true time flow</u>	1112
<u>dft wait for license</u>	1112
<u>et license options</u>	1113
<u>pmbist enable multiple views</u>	1114
<u>unmap scan flops</u>	1114
<u>Design Attributes</u>	1116
<u>boundary type</u>	1116
<u>dft boundary scan exists</u>	1117
<u>dft clock edge for head of scan chains</u>	1117
<u>dft clock edge for tail of scan chains</u>	1118

Attribute Reference for Encounter RTL Compiler

<u>dft connect scan data pins during mapping</u>	1119
<u>dft connect shift enable during mapping</u>	1120
<u>dft dont scan</u>	1121
<u>dft jtag macro exists</u>	1121
<u>dft lockup element type</u>	1122
<u>dft lockup element type for tail of scan chains</u>	1123
<u>dft max length of scan chains</u>	1124
<u>dft min number of scan chains</u>	1124
<u>dft mix clock edges in scan chains</u>	1125
<u>dft scan map mode</u>	1126
<u>dft scan output preference</u>	1127
<u>dft tap tck period</u>	1128
<u>iocell enable</u>	1129
<u>iocell input</u>	1129
<u>iocell output</u>	1130
<u>mbist enable shared library domain set</u>	1130
<u>Instance Attributes</u>	1132
<u>dft abstract dont scan</u>	1132
<u>dft custom se</u>	1132
<u>dft dont scan</u>	1133
<u>dft exempt from system clock check</u>	1134
<u>dft force blackbox for atpg</u>	1134
<u>dft is blackbox for atpg</u>	1134
<u>dft mapped</u>	1135
<u>dft part of segment</u>	1135
<u>dft scan chain</u>	1135
<u>dft status</u>	1136
<u>dft test clock</u>	1137
<u>dft test clock edge</u>	1137
<u>dft test clock source</u>	1138
<u>dft violation</u>	1138
<u>mbist instruction set</u>	1139
<u>pmbist instruction set</u>	1140
<u>Pin Attributes</u>	1141
<u>dft constant value</u>	1141
<u>dft controllable</u>	1142

Attribute Reference for Encounter RTL Compiler

<u>dft driven by clock</u>	1142
<u>dft opcg domain clock pin</u>	1143
<u>dft opcg domain fanout pin</u>	1143
<u>dft opcg domain launch clock</u>	1144
<u>dft opcg domain se input pin</u>	1145
<u>dft opcg domain unfenced capture</u>	1145
<u>user differential negative pin</u>	1146
<u>user from core data</u>	1146
<u>user from core enable</u>	1147
<u>user test receiver acmode</u>	1148
<u>user test receiver data output</u>	1148
<u>user test receiver init clock</u>	1148
<u>user test receiver init data</u>	1148
<u>user to core data</u>	1149
<u>user to core enable</u>	1149
<u>wrapper control</u>	1150
<u>wrapper segment</u>	1150
<u>Net Attributes</u>	1151
<u>dft clock domain info</u>	1151
<u>dft constant value</u>	1152
<u>Subdesign Attributes</u>	1153
<u>dft dont scan</u>	1153
<u>Support Attributes</u>	1155
<u>dft driven by clock</u>	1155
<u>dft opcg domain clock pin</u>	1155
<u>dft opcg domain fanout pin</u>	1156
<u>dft opcg domain launch clock</u>	1156
<u>dft opcg domain se input pin</u>	1157
<u>dft opcg domain unfenced capture</u>	1157
<u>wrapper control</u>	1158
<u>wrapper segment</u>	1159
<u>Port Attributes</u>	1160
<u>dft driven by clock</u>	1160
<u>dft enable hookup pin</u>	1160
<u>dft enable hookup polarity</u>	1161
<u>dft opcg asserted domain</u>	1162

Attribute Reference for Encounter RTL Compiler

<u>dft opcg domain clock pin</u>	1162
<u>dft opcg domain fanout pin</u>	1163
<u>dft opcg domain launch clock</u>	1163
<u>dft opcg domain se input pin</u>	1164
<u>dft opcg domain unfenced capture</u>	1165
<u>dft sdi output hookup pin</u>	1166
<u>dft sdo input hookup pin</u>	1166
<u>wrapper control</u>	1166
<u>wrapper segment</u>	1167
<u>Boundary-Scan Segment Attributes</u>	1168
<u>acdcsel 11496</u>	1168
<u>acpclk 11496</u>	1168
<u>acpsen 11496</u>	1169
<u>acptrenbl 11496</u>	1169
<u>acpulse 11496</u>	1170
<u>bsdl</u>	1170
<u>capturedr</u>	1171
<u>clockdr</u>	1171
<u>differential_pairs</u>	1172
<u>highz</u>	1172
<u>instance</u>	1173
<u>modea</u>	1173
<u>modeb</u>	1174
<u>modec</u>	1174
<u>shiftdr</u>	1175
<u>tdi</u>	1175
<u>tdo</u>	1176
<u>updatedr</u>	1176
<u>JTAG Instruction Attributes</u>	1177
<u>capture</u>	1177
<u>length</u>	1177
<u>opcode</u>	1178
<u>private</u>	1178
<u>register</u>	1178
<u>register_capturedr</u>	1179
<u>register_clockdr</u>	1179

Attribute Reference for Encounter RTL Compiler

<u>register_decode</u>	1180
<u>register_reset</u>	1180
<u>register_reset_polarity</u>	1180
<u>register_rnidle</u>	1181
<u>register_shiftdr</u>	1181
<u>register_shiftdr_polarity</u>	1181
<u>register_tck</u>	1182
<u>register_tdi</u>	1182
<u>register_tdo</u>	1182
<u>register_updatedr</u>	1183
<u>tap_decode</u>	1183
<u>tap_tdi</u>	1184
<u>tap_tdo</u>	1184
<u>JTAG Instruction Register Attributes</u>	1185
<u>capture</u>	1185
<u>length</u>	1185
<u>JTAG Macro Attributes</u>	1186
<u>boundary_tdo</u>	1186
<u>bsr_clockdr</u>	1186
<u>bsr_shiftdr</u>	1186
<u>bsr_updatedr</u>	1187
<u>capturedr</u>	1187
<u>clockdr</u>	1187
<u>dot6_acdcsel</u>	1188
<u>dot6_acmode</u>	1188
<u>dot6_acpulse</u>	1188
<u>dot6_preset_clock</u>	1188
<u>dot6_trcell_enable</u>	1189
<u>exitdr</u>	1189
<u>highz</u>	1189
<u>instance</u>	1189
<u>mode_a</u>	1190
<u>mode_b</u>	1190
<u>mode_c</u>	1190
<u>por</u>	1191
<u>reset</u>	1191

Attribute Reference for Encounter RTL Compiler

<u>runidle</u>	1191
<u>shiftdr</u>	1192
<u>tck</u>	1192
<u>tdi</u>	1192
<u>tdo</u>	1193
<u>tdo_enable</u>	1193
<u>tms</u>	1193
<u>trst</u>	1194
<u>updatedr</u>	1194
<u>user defined macro</u>	1194
JTAG Port Attributes	1195
<u>aio_pin</u>	1195
<u>bcell_location</u>	1195
<u>bcell_required</u>	1196
<u>bcell_segment</u>	1196
<u>bcell_type</u>	1197
<u>bdy_enable</u>	1197
<u>bdy_in</u>	1198
<u>bdy_out</u>	1198
<u>bsr_dummy_after</u>	1198
<u>bsr_dummy_before</u>	1199
<u>cell</u>	1200
<u>comp_enable</u>	1201
<u>custom_bcell</u>	1201
<u>differential</u>	1201
<u>index</u>	1202
<u>pin</u>	1202
<u>pinmap</u>	1202
<u>sys_enable</u>	1203
<u>sys_use</u>	1203
<u>test_use</u>	1204
<u>tr_bdy_in</u>	1204
<u>tr_cell</u>	1204
<u>trcell_acmode</u>	1205
<u>trcell_clock</u>	1205
<u>trcell_enable</u>	1205

Attribute Reference for Encounter RTL Compiler

<u>type</u>	1206
<u>TAP Port Attributes</u>	1207
<u>dft hookup pin</u>	1207
<u>dft hookup polarity</u>	1207
<u>pin</u>	1207
<u>type</u>	1208
<u>DFT Configuration Mode Attributes</u>	1209
<u>current mode</u>	1209
<u>decoded pin</u>	1209
<u>jtag instruction</u>	1210
<u>mode enable high</u>	1210
<u>mode enable low</u>	1211
<u>type</u>	1212
<u>usage</u>	1212
<u>user defined</u>	1213
<u>Memory Data Bit Structure Attributes</u>	1214
<u>column order</u>	1214
<u>partial row order</u>	1214
<u>row order</u>	1215
<u>Memory Libcell Attributes</u>	1216
<u>address limit</u>	1216
<u>data order</u>	1216
<u>memory libcell</u>	1217
<u>read delay</u>	1217
<u>wrapper</u>	1217
<u>Memory Libpin Action Attributes</u>	1219
<u>value</u>	1219
<u>Memory Libpin Alias Attributes</u>	1220
<u>base port name</u>	1220
<u>Memory Spare Column Attributes</u>	1221
<u>address bits</u>	1221
<u>banks</u>	1221
<u>data bits</u>	1222
<u>enable</u>	1222
<u>srclk</u>	1222
<u>srsi</u>	1223

Attribute Reference for Encounter RTL Compiler

<u>srs0</u>	1223
<u>Memory Spare Column Map Address Attributes</u>	1224
<u>address_logical_value</u>	1224
<u>address_port</u>	1224
<u>address_port_value</u>	1225
<u>Memory Spare Column Map Data Attributes</u>	1226
<u>data_logical_value</u>	1226
<u>data_port</u>	1226
<u>data_port_value</u>	1227
<u>Memory Spare Row Attributes</u>	1228
<u>address_bits</u>	1228
<u>banks</u>	1228
<u>data_bits</u>	1229
<u>enable</u>	1229
<u>srclk</u>	1229
<u>srsi</u>	1230
<u>srs0</u>	1230
<u>Memory Spare Row Map Address Attributes</u>	1231
<u>address_logical_value</u>	1231
<u>address_port</u>	1231
<u>address_port_value</u>	1232
<u>Write Mask Bit Attributes</u>	1233
<u>masked_bits</u>	1233
<u>Direct Access Function Attributes</u>	1234
<u>active</u>	1234
<u>clocked_by_mtclk</u>	1234
<u>source</u>	1235
<u>Programmable Direct Access Function Attributes</u>	1236
<u>active</u>	1236
<u>source</u>	1236
<u>MBIST Clock Attributes</u>	1237
<u>dft hookup_pin</u>	1237
<u>dft hookup_polarity</u>	1237
<u>hookup_period</u>	1238
<u>internal</u>	1238
<u>is_jtag_tck</u>	1238

Attribute Reference for Encounter RTL Compiler

<u>period</u>	1239
<u>sources</u>	1239
<u>Domain Macro Parameters Attributes</u>	1240
<u>counter_length</u>	1240
<u>max_num_pulses</u>	1240
<u>target_period</u>	1241
<u>trigger_delay</u>	1241
<u>OPCG Domain Attributes</u>	1242
<u>divide_by</u>	1242
<u>domain_macro_parameter</u>	1242
<u>location</u>	1242
<u>min_domain_period</u>	1243
<u>opcg_trigger</u>	1243
<u>osc_source</u>	1243
<u>scan_clock</u>	1244
<u>OPCG Mode Attributes</u>	1245
<u>jtag_controlled</u>	1245
<u>mode_init</u>	1245
<u>OPCG Trigger Attributes</u>	1246
<u>active</u>	1246
<u>inside_inst</u>	1246
<u>osc_source</u>	1247
<u>pin</u>	1247
<u>scan_clock</u>	1247
<u>Osc Source Attributes</u>	1248
<u>max_input_period</u>	1248
<u>max_output_period</u>	1248
<u>min_input_period</u>	1249
<u>min_output_period</u>	1249
<u>pin</u>	1250
<u>ref_clock_pin</u>	1250
<u>Osc Source Reference Attributes</u>	1251
<u>osc_source_period</u>	1251
<u>ref_clk_period</u>	1251
<u>Actual Scan Chain Attributes</u>	1252
<u>analyzed</u>	1252

Attribute Reference for Encounter RTL Compiler

<u>compressed</u>	1252
<u>connected shift enable</u>	1253
<u>dft hookup pin sdi</u>	1253
<u>dft hookup pin sdo</u>	1254
<u>domain</u>	1254
<u>edge</u>	1254
<u>elements</u>	1255
<u>has opcg segments</u>	1255
<u>head lockup</u>	1256
<u>mode name</u>	1256
<u>other clocks</u>	1256
<u>power domain</u>	1257
<u>reg count</u>	1257
<u>scan clock a</u>	1257
<u>scan clock b</u>	1258
<u>scan in</u>	1258
<u>scan out</u>	1259
<u>sdi compression signal</u>	1259
<u>shared output</u>	1260
<u>shared select</u>	1260
<u>shift enable</u>	1260
<u>terminal lockup</u>	1261
<u>Actual Scan Segment Attributes</u>	1262
<u>active</u>	1262
<u>clock</u>	1263
<u>clock edge</u>	1263
<u>connected scan clock a</u>	1264
<u>connected scan clock b</u>	1264
<u>connected shift enable</u>	1264
<u>core wrapper</u>	1265
<u>dft tail test clock</u>	1265
<u>dft tail test clock edge</u>	1266
<u>dft test clock</u>	1266
<u>dft test clock edge</u>	1267
<u>elements</u>	1268
<u>instance</u>	1268

Attribute Reference for Encounter RTL Compiler

<u>other_clocks</u>	1268
<u>power_domain</u>	1269
<u>reg_count</u>	1270
<u>reorderable</u>	1270
<u>scan_clock_a</u>	1271
<u>scan_clock_b</u>	1271
<u>scan_in</u>	1271
<u>scan_out</u>	1272
<u>shift_enable</u>	1273
<u>skew_safe</u>	1273
<u>tail_clock</u>	1274
<u>tail_clock_edge</u>	1274
<u>type</u>	1275
<u>Violations Attributes</u>	1276
<u>description</u>	1276
<u>endpoints</u>	1278
<u>file_name</u>	1279
<u>fixed</u>	1279
<u>id</u>	1280
<u>line_number</u>	1281
<u>reg_count</u>	1281
<u>registers</u>	1282
<u>root_node</u>	1283
<u>segments</u>	1283
<u>tristate_net_drivers</u>	1284
<u>tristate_net_load</u>	1284
<u>type</u>	1284
<u>Scan Chain Attributes</u>	1285
<u>body</u>	1285
<u>complete</u>	1285
<u>dft hookup pin_sdi</u>	1286
<u>dft hookup pin_sdo</u>	1286
<u>domain</u>	1286
<u>edge</u>	1287
<u>head</u>	1287
<u>max_length</u>	1287

Attribute Reference for Encounter RTL Compiler

<u>scan_clock_a</u>	1288
<u>scan_clock_b</u>	1288
<u>scan_in</u>	1289
<u>scan_out</u>	1289
<u>sdi_compression_signal</u>	1289
<u>shared_output</u>	1290
<u>shared_select</u>	1290
<u>shift_enable</u>	1290
<u>tail</u>	1291
<u>terminal_lockup</u>	1291
<u>Scan Segment Attributes</u>	1292
<u>active</u>	1292
<u>clock</u>	1292
<u>clock_edge</u>	1293
<u>connected_scan_clock_a</u>	1293
<u>connected_scan_clock_b</u>	1294
<u>connected_shift_enable</u>	1294
<u>core_wrapper</u>	1294
<u>core_wrapper_ports</u>	1295
<u>core_wrapper_type</u>	1295
<u>core_wrapper_usage</u>	1296
<u>dft_dont_scan</u>	1296
<u>dft_status</u>	1297
<u>dft_tail_test_clock</u>	1298
<u>dft_tail_test_clock_edge</u>	1298
<u>dft_test_clock</u>	1299
<u>dft_test_clock_edge</u>	1299
<u>dftViolation</u>	1300
<u>elements</u>	1300
<u>instance</u>	1301
<u>other_clocks</u>	1301
<u>power_domain</u>	1301
<u>reg_count</u>	1302
<u>reorderable</u>	1303
<u>scan_clock_a</u>	1303
<u>scan_clock_b</u>	1303

Attribute Reference for Encounter RTL Compiler

<u>scan_in</u>	1304
<u>scan_out</u>	1304
<u>shift_enable</u>	1305
<u>skew_safe</u>	1305
<u>tail_clock</u>	1305
<u>tail_clock_edge</u>	1306
<u>test_modes</u>	1306
<u>type</u>	1307
<u>user_defined_segment</u>	1308
<u>Test Bus Port Attributes</u>	1309
<u>dft hookup pin</u>	1309
<u>dft hookup polarity</u>	1309
<u>function</u>	1310
<u>index</u>	1310
<u>pin</u>	1310
<u>Test Clock Attributes</u>	1311
<u>at_speed</u>	1311
<u>atpg_use</u>	1312
<u>blocking_se</u>	1312
<u>controllable</u>	1313
<u>dft hookup pin</u>	1313
<u>dft hookup polarity</u>	1313
<u>dft mask_clk</u>	1314
<u>dft misr_clock</u>	1314
<u>divide_fall</u>	1315
<u>divide_period</u>	1315
<u>divide_rise</u>	1315
<u>fall</u>	1316
<u>off_state</u>	1316
<u>period</u>	1317
<u>rise</u>	1317
<u>root_source_pins</u>	1317
<u>root_source_polarity</u>	1318
<u>sources</u>	1318
<u>user_defined_signal</u>	1319
<u>Test Signal Attributes</u>	1320

Attribute Reference for Encounter RTL Compiler

<u>active</u>	1320
<u>atpg_use</u>	1321
<u>dedicated_pin</u>	1322
<u>default_shift_enable</u>	1322
<u>dft_compression_signal</u>	1323
<u>dft hookup pin</u>	1323
<u>dft hookup polarity</u>	1324
<u>divide_fall</u>	1324
<u>divide_period</u>	1325
<u>divide_rise</u>	1325
<u>fall</u>	1325
<u>has_fanout</u>	1326
<u>ideal</u>	1326
<u>lec_value</u>	1327
<u>master_signal</u>	1327
<u>period</u>	1328
<u>pin</u>	1328
<u>pmbist_use</u>	1329
<u>rise</u>	1330
<u>scan_shift</u>	1330
<u>type</u>	1331
<u>user_defined_signal</u>	1331

13

<u>Low Power Synthesis</u>	1333
<u>List</u>	1333
<u>Root Attributes</u>	1338
<u> leakage_power_effort</u>	1338
<u> lp_clock_gating_exceptions_aware</u>	1339
<u> lp_clock_gating_infer_enable</u>	1339
<u> lp_clock_gating_prefix</u>	1340
<u> lp_clock_gating_register_aware</u>	1341
<u> lp_default_probability</u>	1341
<u> lp_default_toggle_rate</u>	1342
<u> lp_display_negative_internal_power</u>	1342

Attribute Reference for Encounter RTL Compiler

<u>lp_insert_clock_gating</u>	1343
<u>lp_power_analysis_effort</u>	1344
<u>lp_power_unit</u>	1345
<u>lp_pso_aware_estimation</u>	1346
<u>lp_toggle_rate_unit</u>	1346
<u>lp_x_transition_probability_count</u>	1347
<u>lp_x_transition_toggle_count</u>	1347
<u>lp_z_transition_probability_count</u>	1348
<u>lp_z_transition_toggle_count</u>	1348
<u>Design Attributes</u>	1349
<u>lp_clock_gating_add_obs_port</u>	1349
<u>lp_clock_gating_add_reset</u>	1350
<u>lp_clock_gating_auto_cost_group_initial_target</u>	1351
<u>lp_clock_gating_auto_cost_grouping</u>	1351
<u>lp_clock_gating_auto_path_adjust</u>	1352
<u>lp_clock_gating_auto_path_adjust_fixed_delay</u>	1352
<u>lp_clock_gating_auto_path_adjust_modes</u>	1353
<u>lp_clock_gating_auto_path_adjust_multiplier</u>	1354
<u>lp_clock_gating_cell</u>	1354
<u>lp_clock_gating_control_point</u>	1355
<u>lp_clock_gating_exclude</u>	1357
<u>lp_clock_gating_extract_common_enable</u>	1357
<u>lp_clock_gating_max_flops</u>	1358
<u>lp_clock_gating_min_flops</u>	1358
<u>lp_clock_gating_module</u>	1359
<u>lp_clock_gating_style</u>	1359
<u>lp_clock_gating_test_signal</u>	1360
<u>lp_clock_tree_buffers</u>	1361
<u>lp_clock_tree_leaf_max_fanout</u>	1361
<u>lp_default_probability</u>	1361
<u>lp_default_toggle_percentage</u>	1362
<u>lp_default_toggle_rate</u>	1363
<u>lp_internal_power</u>	1363
<u>lp_leakage_power</u>	1364
<u>lp_net_power</u>	1364
<u>lp_power_optimization_weight</u>	1365

Attribute Reference for Encounter RTL Compiler

<u>lp_pso_aware_tcf</u>	1365
<u>max_dynamic_power</u>	1366
<u>max_leakage_power</u>	1367
<u>Instance Attributes</u>	1368
<u>instance_internal_power</u>	1368
<u>instance_leakage_power</u>	1369
<u>lp_clock_gating_add_reset</u>	1369
<u>lp_clock_gating_cell</u>	1370
<u>lp_clock_gating_exclude</u>	1371
<u>lp_clock_gating_gated_clock_gates</u>	1372
<u>lp_clock_gating_gated_flops</u>	1372
<u>lp_clock_gating_is_flop_rc_gated</u>	1373
<u>lp_clock_gating_is_flop_user_gated</u>	1373
<u>lp_clock_gating_is_leaf_clock_gate</u>	1373
<u>lp_clock_gating_module</u>	1374
<u>lp_clock_gating_rc_inserted</u>	1374
<u>lp_clock_gating_stage</u>	1374
<u>lp_clock_gating_test_signal</u>	1375
<u>lp_default_probability</u>	1376
<u>lp_default_toggle_rate</u>	1377
<u>lp_dynamic_analysis_scope</u>	1377
<u>lp_internal_power</u>	1378
<u>lp_leakage_power</u>	1378
<u>lp_net_power</u>	1379
<u>Pin Attributes</u>	1380
<u>lp_asserted_probability</u>	1380
<u>lp_asserted_toggle_rate</u>	1381
<u>lp_computed_probability</u>	1381
<u>lp_computed_toggle_rate</u>	1382
<u>lp_default_probability</u>	1383
<u>lp_default_toggle_rate</u>	1384
<u>lp_net_power</u>	1385
<u>lp_probability_type</u>	1386
<u>lp_toggle_rate_type</u>	1387
<u>Net Attributes</u>	1389
<u>lp_asserted_probability</u>	1389

Attribute Reference for Encounter RTL Compiler

<u>lp asserted toggle rate</u>	1390
<u>lp computed probability</u>	1390
<u>lp computed toggle rate</u>	1391
<u>lp net power</u>	1392
<u>lp probability type</u>	1393
<u>lp toggle rate type</u>	1394
Port Attributes	1396
<u>lp asserted probability</u>	1396
<u>lp asserted toggle rate</u>	1396
<u>lp computed probability</u>	1397
<u>lp computed toggle rate</u>	1398
<u>lp default probability</u>	1399
<u>lp default toggle rate</u>	1400
<u>lp net power</u>	1401
<u>lp probability type</u>	1402
<u>lp toggle rate type</u>	1403
Subdesign Attributes	1404
<u>lp clock gating add reset</u>	1404
<u>lp clock gating auto path adjust</u>	1405
<u>lp clock gating auto path adjust fixed delay</u>	1406
<u>lp clock gating auto path adjust multiplier</u>	1406
<u>lp clock gating cell</u>	1407
<u>lp clock gating exclude</u>	1408
<u>lp clock gating module</u>	1409
<u>lp clock gating test signal</u>	1409
Support Attributes	1411
<u>lp asserted probability</u>	1411
<u>lp asserted toggle rate</u>	1412
<u>lp computed probability</u>	1412
<u>lp computed toggle rate</u>	1413
<u>lp net power</u>	1414
<u>lp probability type</u>	1415
<u>lp toggle rate type</u>	1416
Clock Attributes	1417
<u>lp default toggle percentage</u>	1417

14

<u>Advanced Low Power</u>	1419
<u>List</u>	1419
<u>Root Attributes</u>	1424
<u>clp ignore ls high to low</u>	1424
<u>clp treat errors as warnings</u>	1424
<u>Library Domain Attributes</u>	1426
<u>active operating conditions</u>	1426
<u>default</u>	1427
<u>library</u>	1427
<u>operating conditions</u>	1429
<u>power library</u>	1429
<u>wireload selection</u>	1430
<u>Level Shifter Group Attributes</u>	1431
<u>direction</u>	1431
<u>ground direction</u>	1431
<u>level shifter cells</u>	1432
<u>max ground input voltage</u>	1432
<u>max ground output voltage</u>	1433
<u>max input voltage</u>	1433
<u>max output voltage</u>	1433
<u>min ground input voltage</u>	1434
<u>min ground output voltage</u>	1434
<u>min input voltage</u>	1435
<u>min output voltage</u>	1435
<u>valid location</u>	1435
<u>Libcell Attributes</u>	1436
<u>aux enables</u>	1436
<u>valid location</u>	1436
<u>Design Attributes</u>	1437
<u>base mode</u>	1437
<u>library domain</u>	1437
<u>lp isolate domain crossing constants</u>	1438
<u>pi relax map iso cell checks</u>	1438
<u>pi relax map ls cell checks</u>	1439

Attribute Reference for Encounter RTL Compiler

<u>power_domain</u>	1439
<u>Constant Attributes</u>	1440
<u>power_domain</u>	1440
<u>Port Attributes</u>	1441
<u>isolation_rule</u>	1441
<u>level_shifter_rule</u>	1441
<u>power_domain</u>	1442
<u>hdl_arch Attributes</u>	1443
<u>library_domain</u>	1443
<u>Subdesign Attributes</u>	1444
<u>library_domain</u>	1444
<u>Support Attributes</u>	1446
<u>isolation_rule</u>	1446
<u>level_shifter_rule</u>	1447
<u>power_domain</u>	1447
<u>Instance Attributes</u>	1448
<u>library_domain</u>	1448
<u>power_domain</u>	1448
<u>secondary_domain</u>	1449
<u>state_retention_rule</u>	1449
<u>Pin Attributes</u>	1450
<u>isolation_rule</u>	1450
<u>level_shifter_rule</u>	1450
<u>power_domain</u>	1451
<u>Isolation Rule Attributes</u>	1452
<u>cells</u>	1452
<u>cpf_pins</u>	1452
<u>enable_driver</u>	1453
<u>enable_polarity</u>	1453
<u>exclude_pins</u>	1453
<u>from_power_domain</u>	1454
<u>location</u>	1454
<u>off_domain</u>	1454
<u>output_value</u>	1455
<u>pins</u>	1455
<u>prefix</u>	1455

Attribute Reference for Encounter RTL Compiler

<u>secondary domain</u>	1456
<u>to power domain</u>	1456
<u>within hierarchy</u>	1456
<u>Level Shifter Rule Attributes</u>	1457
<u>cells</u>	1457
<u>cpf_pins</u>	1457
<u>direction</u>	1457
<u>exclude_pins</u>	1458
<u>from power domain</u>	1458
<u>location</u>	1458
<u>pins</u>	1459
<u>prefix</u>	1459
<u>to power domain</u>	1459
<u>within hierarchy</u>	1460
<u>Nominal Condition Attributes</u>	1461
<u>ground voltage</u>	1461
<u>library set</u>	1462
<u>voltage</u>	1462
<u>Power Domain Attributes</u>	1463
<u>base domains</u>	1463
<u>default</u>	1463
<u>dft iso rule</u>	1464
<u>has external shutoff</u>	1464
<u>library domain</u>	1465
<u>library domain by mode</u>	1465
<u>shutoff condition</u>	1466
<u>Power Mode Attributes</u>	1467
<u>constraint mode</u>	1467
<u>default</u>	1467
<u>domain conditions</u>	1467
<u>State Retention Rule Attributes</u>	1468
<u>cell type</u>	1468
<u>cells</u>	1468
<u>restore</u>	1469
<u>restore phase</u>	1469
<u>save</u>	1469

Attribute Reference for Encounter RTL Compiler

<u>save_phase</u>	1469
<u>secondary_domain</u>	1470
15		
<u>Customization</u>	1471
<u>List</u>	1471
<u>Root Attributes</u>	1473
<u>ui_respects_preserve</u>	1473
<u>Design Attributes</u>	1474
<u>user_defined</u>	1474
<u>Instance Attributes</u>	1475
<u>user_defined</u>	1475
<u>Pin Attributes</u>	1476
<u>user_defined</u>	1476
<u>Pgpin Attributes</u>	1477
<u>user_defined</u>	1477
<u>Net Attributes</u>	1478
<u>user_defined</u>	1478
<u>Port Attributes</u>	1479
<u>user_defined</u>	1479
<u>Subdesign Attributes</u>	1480
<u>user_defined</u>	1480
<u>Subport Attributes</u>	1481
<u>user_defined</u>	1481
<u>Message Group Attributes</u>	1482
<u>is_user</u>	1482
A		
<u>Applets</u>	1483
<u>List</u>	1483
<u>Root Attributes</u>	1484
<u>applet_mode</u>	1484
<u>applet_replay</u>	1485
<u>applet_search_path</u>	1485
<u>applet_server</u>	1486

Attribute Reference for Encounter RTL Compiler

<u>applet server pass</u>	1487
<u>applet server user</u>	1487
<u>Index</u>	1489

Alphabetical List of Attributes

A

acdcsel_11496 [1168](#)
 acpclk_11496 [1168](#)
 acpsen_11496 [1169](#)
 acptrenbl_11496 [1169](#)
 acpulse_11496 [1170](#)
 active [1234, 1236, 1246, 1262, 1292, 1320](#)
 active_operating_conditions [928, 1426](#)
 adder [188](#)
 additional_help [135](#)
 address_limit [1216](#)
 adjust_value [1063](#)
 aio_pin [1195](#)
 analyzed [1252](#)
 applet_mode [1484](#)
 applet_replay [1485](#)
 applet_search_path [1485](#)
 applet_server [1486](#)
 applet_server_pass [1487](#)
 applet_server_user [1487](#)
 arch_filename [317, 337](#)
 arch_name [931](#)
 area [188, 931, 1030](#)
 area_multiplier [189](#)
 aspect_ratio [387](#)
 async_clear [189](#)
 async_clear_phase [212](#)
 async_preset [189](#)
 async_preset_phase [212](#)
 at_speed [1311](#)
 atpg_use [1312, 1321](#)
 attribute_path [101](#)
 auto_partition [696](#)
 auto_super_thread [696](#)
 auto_ungroup [697](#)
 aux_enables [1436](#)
 average_net_length [932](#)
 avoid [149, 152, 156, 164, 190](#)
 avoid_no_row_libcell [387](#)

B

bank_based_multibit_inferencing [698](#)
 base_domains [1463](#)

base_mode [1437](#)
 bcell_location [1195](#)
 bcell_required [1196](#)
 bcell_segment [1196](#)
 bcell_type [1197](#)
 bdy_enable [1197](#)
 bdy_in [1198](#)
 bdy_out [1198](#)
 bin [1048](#)
 bit_blasted_port_style [243](#)
 bit_width [168](#)
 bits [996, 1028, 1044](#)
 blackbox [860](#)
 blockages [388](#)
 blocking_se [1312](#)
 body [1285](#)
 bottom_layer [495](#)
 boundary [460](#)
 boundary_change [960](#)
 boundary_optimize_constant_hier_pins [6, 99, 878, 893](#)
 boundary_optimize_equal_opposite_hier_pins [700, 879, 894](#)
 boundary_optimize_feedthrough_hier_pins [701, 881, 883, 896, 897](#)
 boundary_optimize_hier_pin_invertible [88, 2, 913](#)
 boundary_optimize_invert_hier_pins [702, 883, 897](#)
 boundary_optimize_invert_hier_pins_rename_nets [703](#)
 boundary_optimize_invert_hier_pins_renaming_extension [704](#)
 boundary_opto [892](#)
 boundary_tdo [1186](#)
 boundary_type [1116](#)
 box [437](#)
 box_list [470](#)
 boxes [414, 434, 472, 481, 495](#)
 break_timing_paths [574, 616](#)
 break_timing_paths_by_mode [576, 617](#)
 bsdl [1170](#)
 bsr_clockdr [1186](#)
 bsr_dummy_after [1198](#)
 bsr_dummy_before [1199](#)
 bsr_shiftdr [1186](#)

bsr_updatedr [1187](#)
 buffer [190](#), [946](#)
 bundle [212](#)
 bus [1002](#), [1039](#)
 bus_naming_style [704](#)

C

candidate_impls [170](#)
 cap_multiplier [444](#)
 cap_scale_in_fF [180](#)
 cap_table_file [364](#)
 cap_table_name [445](#)
 capacitance [213](#), [444](#), [463](#)
 capture [1177](#), [1185](#)
 capturedr [1171](#), [1187](#)
 capturer [962](#), [1002](#)
 case_analysis_multi_driver_propagation [5](#)
[27](#)
 case_analysis_sequential_propagation [52](#)
[8](#)
 category [135](#)
 causes_ideal_net [962](#), [1003](#), [1039](#)
 ccd_executable [244](#)
 cell [1200](#)
 cell_area [932](#), [946](#), [1031](#)
 cell_bitwidth [191](#)
 cell_count [932](#), [1031](#)
 cell_delay_multiplier [191](#), [545](#), [562](#)
 cell_internal_power [192](#)
 cell_leakage_power [192](#)
 cell_min_delay_multiplier [193](#), [546](#), [563](#)
 cell_type [1468](#)
 cells [1452](#), [1457](#), [1468](#)
 check_function [136](#)
 clock [193](#), [1069](#), [1263](#), [1292](#)
 clock_edge [1263](#), [1293](#)
 clock_gate_clock_pin [213](#)
 clock_gate_enable_pin [213](#)
 clock_gate_obs_pin [213](#)
 clock_gate_out_pin [214](#)
 clock_gate_reset_pin [214](#)
 clock_gate_test_pin [214](#)
 clock_gating_integrated_cell [194](#)
 clock_hold_uncertainty [578](#), [619](#), [668](#)
 clock_hold_uncertainty_by_mode [579](#),
[620](#)
 clock_library_cells [915](#)
 clock_network_early_latency [580](#), [621](#),
[669](#)

clock_network_early_latency_by_mode [5](#)
[82](#), [623](#)
 clock_network_late_latency [584](#), [625](#), [670](#)
 clock_network_late_latency_by_mode [58](#)
[6](#), [627](#)
 clock_network_latency_included [682](#)
 clock_phase [214](#)
 clock_rise [1069](#)
 clock_sense_negative [963](#), [1051](#)
 clock_sense_positive [964](#), [1052](#)
 clock_sense_stop_propagation [964](#), [1052](#)
 clock_setup_uncertainty [588](#), [629](#), [672](#)
 clock_setup_uncertainty_by_mode [589](#),
[631](#)
 clock_source_early_latency [590](#), [632](#), [673](#)
 clock_source_early_latency_by_mode [59](#)
[2](#), [634](#)
 clock_source_late_latency [594](#), [636](#), [675](#)
 clock_source_late_latency_by_mode [596](#),
[638](#)
 clock_source_latency_included [683](#)
 clock_sources_inverted [965](#), [1004](#)
 clock_sources_non_inverted [966](#), [1004](#)
 clockdr [1171](#), [1187](#)
 clocked_by_mtclk [1234](#)
 clp_ignore_ls_high_to_low [1424](#)
 clp_treat_errors_as_warnings [1424](#)
 color [445](#)
 column_order [1214](#)
 comb_seq_merge_message_threshold [70](#)
[5](#)
 combinational [194](#), [947](#)
 command_log [244](#)
 comment [676](#), [680](#)
 comp_enable [1201](#)
 complete [1285](#)
 component [334](#), [414](#)
 components [463](#), [483](#)
 compressed [1252](#)
 compute_function [137](#)
 computed [137](#)
 congestion_avoid [194](#)
 congestion_effort [364](#)
 connect_delay [966](#), [1005](#)
 connected_scan_clock_a [1264](#), [1293](#)
 connected_scan_clock_b [1264](#), [1294](#)
 connected_shift_enable [1253](#), [1264](#), [1294](#)
 constant [997](#)
 constant_0_nets [933](#), [1031](#)
 constant_1_nets [933](#), [1032](#)
 constant_prop_through_iso_cell [706](#)

constraint [149](#)
 constraint_mode [1467](#)
 constraint_multiplier [195](#)
 continue_on_error [101](#)
 control_logic_optimization [707, 832, 899](#)
 controllable [1313](#)
 core_wrapper [1265, 1294](#)
 core_wrapper_ports [1295](#)
 core_wrapper_type [1295](#)
 core_wrapper_usage [1296](#)
 cost_group [1063](#)
 count [131, 491](#)
 counter_length [1240](#)
 cpf_pins [1452, 1453, 1457, 1458](#)
 cpu_runtime [102](#)
 current_mode [1209](#)
 current_value [166](#)
 cut_cols [496](#)
 cut_layer [496](#)
 cut_pattern [496](#)
 cut_rows [497](#)
 cutouts [461](#)
 cx [409](#)
 cy [409](#)

D

d_function [231](#)
 data_order [1216](#)
 data_pins [195](#)
 data_type [138](#)
 decoded_pin [1209](#)
 dedicated_pin [1322](#)
 def_component_mask_shift [388](#)
 def_extension [389](#)
 def_file [389](#)
 def_history [389](#)
 def_name [422, 441, 454, 464, 472, 483](#)
 def_output_escape_multibit [365](#)
 def_output_version [365](#)
 def_technology [390](#)
 def_version [390](#)
 default [561, 1427, 1463, 1467](#)
 default_location [335](#)
 default_power_rail [180](#)
 default_shift_enable [1322](#)
 default_value [138](#)
 default_wireload [180](#)
 define_clock_with_new_cost_group [529](#)
 delay [1069](#)

delay_value [1063](#)
 delete_flops_on_preserved_net [707](#)
 delete_hier_insts_on_preserved_net [708](#)
 delete_unloaded_insts [708, 900](#)
 delete_unloaded_seqs [709, 833, 900, 901](#)
 density [415](#)
 derive_bussed_pins [709](#)
 derived_from_power_domain [473](#)
 description [1276](#)
 designware_compatibility [153](#)
 detailed_sdc_messages [244](#)
 dft [1128](#)
 dft_abstract_dont_scan [1132](#)
 dft_apply_sdc_constraints [1093](#)
 dft_boundary_cell_module_prefix [1093](#)
 dft_boundary_scan_exists [1117](#)
 dft_capture_clock_edge_for_head_of_scan_chains [1117](#)
 dft_clock_domain_info [1151](#)
 dft_clock_edge_for_tail_of_scan_chains [118](#)
 dft_clock_waveform_divide_fall [1094](#)
 dft_clock_waveform_divide_period [1094](#)
 dft_clock_waveform_divide_rise [1095](#)
 dft_clock_waveform_fall [1095](#)
 dft_clock_waveform_period [1096](#)
 dft_clock_waveform_rise [1096](#)
 dft_compression_signal [1323](#)
 dft_connect_scan_data_pins_during_mapping [1119](#)
 dft_connect_shift_enable_during_mapping [1120](#)
 dft_constant_value [1141, 1152](#)
 dft_controllable [1142](#)
 dft_custom_se [1132](#)
 dft_dont_scan [1121, 1133, 1153](#)
 dft_driven_by_clock [1142, 1155, 1160](#)
 dft_enable hookup_pin [1160](#)
 dft_enable hookup_polarity [1161](#)
 dft_exempt_from_system_clock_check [1134](#)
 dft_fence_slow_speed_domains [1097](#)
 dft_force_blackbox_for_atpg [1134](#)
 dft_generate_et_no_testpoint_file [1097](#)
 dft_hookup_pin [1207, 1237, 1309, 1313, 1323](#)
 dft_hookup_pin_sdi [1253, 1286](#)
 dft_hookup_pin_sdo [1254, 1286](#)
 dft_hookup_polarity [1207, 1237, 1309, 1313, 1324](#)
 dft_identify_internal_test_clocks [1098](#)

Attribute Reference for Encounter RTL Compiler

dft_identify_test_signals [1099](#)
dft_identify_top_level_test_clocks [1099](#)
dft_identify_xsource_violations_from_timing
models [1099](#)
dft_include_controllable_pins_in_abstract_
model [1100](#)
dft_include_test_signal_outputs_in_abstract
model [1104](#)
dft_is_blackbox_for_atpg [1134](#)
dft_iso_rule [1464](#)
dft_jtag_macro_exists [1121](#)
dft_jtag_module_name [1106](#)
dft_lockup_element_type [1122](#)
dft_lockup_element_type_for_tail_of_scan_
chains [1123](#)
dft_mapped [1135](#)
dft_mask_clock [1314](#)
dft_max_length_of_scan_chains [1124](#)
dft_min_number_of_scan_chains [1124](#)
dft_misr_clock [1314](#)
dft_mix_clock_edges_in_scan_chains [112](#)
[5](#)
dft_opcg_asserted_domain [1162](#)
dft_opcg_block_input_to_flop_paths [1107](#)
dft_opcg_domain_blocking [1107](#)
dft_opcg_domain_clock_pin [1143, 1155,](#)
[1162](#)
dft_opcg_domain_fanout_pin [1143, 1156,](#)
[1163](#)
dft_opcg_domain_launch_clock [1144,](#)
[1156, 1163](#)
dft_opcg_domain_se_input_pin [1145,](#)
[1157, 1164](#)
dft_opcg_domain_unfenced_capture [1145](#)
[, 1157, 1165](#)
dft_part_of_segment [1135](#)
dft_prefix [1108](#)
dft_propagate_test_signals_from_hookup_
pins_only [1108](#)
dft_report_empty_test_clocks [1109](#)
dft_rtl_insertion [1109](#)
dft_scan_chain [1135](#)
dft_scan_map_mode [1126](#)
dft_scan_output_preference [1127](#)
dft_scan_style [1110](#)
dft_scanbit_waveform_analysis [1110](#)
dft_sdi_input_hookup_pin [1166](#)
dft_sdi_output_hookup_pin [1166](#)
dft_shift_register_identification_mode [111](#)
[1](#)
dft_status [1136, 1297](#)

dft_tail_test_clock [1265, 1298](#)
dft_tail_test_clock_edge [1266, 1298](#)
dft_tap_tck_period [1128](#)
dft_test_clock [1137, 1266, 1299](#)
dft_test_clock_edge [1137, 1267, 1299](#)
dft_test_clock_source [1138](#)
dft_true_time_flow [1112](#)
dftViolation [1138, 1300](#)
dft_wait_for_license [1112](#)
die_area [391](#)
differential_pairs [1172](#)
direction [168, 195, 428, 445, 967, 993,](#)
[1005, 1028, 1040, 1044, 1431, 1457](#)
disabled_arcs [564](#)
disabled_arcs_by_mode [565](#)
disjoint_hinst_box_list [470](#)
divide_by [1242](#)
divide_fall [1053, 1315, 1324](#)
divide_period [1053, 1315, 1325](#)
divide_rise [1054, 1315, 1325](#)
divide_waveform [1054](#)
domain [1064, 1254, 1286](#)
domain_conditions [1467](#)
domain_macro_parameter [1242](#)
dont_infer_multibit [841](#)
dont_report_library [102](#)
dont_retime [842](#)
dont_split_multibit [842](#)
dont_use_qbar_pin [843](#)
dont_use_qbar_seq_pins [710](#)
dot6_acdcsel [1188](#)
dot6_acmode [1188](#)
dot6_acpulse [1188](#)
dot6_preset_clock [1188](#)
dot6_trcell_enable [1189](#)
dp_area_mode [710](#)
dp_csa [711, 833, 901](#)
dp_postmap_downsize [711](#)
dp_postmap_upsize [712](#)
dp_rewriting [712, 834, 902](#)
dp_sharing [713, 835, 903](#)
dp_speculation [714, 836, 904](#)
dp_ungroup_separator [715](#)
dp_verify_ok [836, 904](#)
drc_first [529](#)
drc_max_cap_first [530](#)
drc_max_fanout_first [531](#)
drc_max_trans_first [533](#)
driven_by_supply0 [997](#)
driven_by_supply1 [998](#)
driver_for_unloaded_hier_pins [716](#)

Attribute Reference for Encounter RTL Compiler

drivers [967](#), [993](#), [999](#), [1005](#), [1040](#)

E

edge [1254](#), [1287](#)
elapsed_runtime [102](#)
elements [1255](#), [1268](#), [1300](#)
embedded_script [318](#), [338](#)
enable_break_timing_paths_by_mode [53](#)
[4](#)
enable_data_check [535](#)
enable_driver [1453](#)
enable_polarity [1453](#)
enabled [229](#)
enc_assign_buffer [366](#)
enc_assign_removal [366](#)
enc_clk_gate_recloning [366](#)
enc_cpu_usage [367](#)
enc_design_mode_process [367](#)
enc_enable_useful_skew [367](#)
enc_force_place_incr [368](#)
enc_gzip_interface_files [368](#)
enc_launch_servers [369](#)
enc_memory_usage [370](#)
enc_module_plan [370](#)
enc_postload_script [371](#)
enc_pre_place_opt [371](#)
enc_preexport_script [372](#)
enc_preload_script [372](#)
enc_save_db [373](#)
enc_scan_def_file [373](#)
enc_temp_dir [373](#)
enc_timing_driven_place [374](#)
enc_user_constraint_file [375](#)
enc_user_mode_file [376](#)
encounter_executable [247](#)
encrypted [330](#)
end_point [1048](#)
endpoint [967](#), [1006](#)
endpoints [1278](#)
entity [333](#)
entity_filename [318](#), [338](#)
entity_name [933](#), [1032](#)
error_on_lib_lef_pin_inconsistency [247](#)
et_license_options [1113](#)
ets_executable [248](#)
exact_match_seq_async_ctrls [716](#)
exact_match_seq_sync_ctrls [717](#)
exception_type [1064](#)
exceptions [947](#), [968](#), [1006](#), [1049](#), [1055](#),

[1058](#), [1059](#), [1070](#)
exitdr [1189](#)
external_delay_pins [1070](#)
external_delays [968](#), [1007](#)
external_delays_by_mode [1007](#)
external_driven_pin_fall [640](#)
external_driven_pin_rise [640](#)
external_driver [641](#)
external_driver_from_pin [641](#)
external_driver_input_slew [642](#)
external_fanout_load [643](#)
external_non_tristate_drivers [643](#)
external_pin_cap [644](#)
external_resistance [644](#)
external_wire_cap [645](#)
external_wire_res [645](#)
external_wireload_fanout [646](#)
external_wireload_model [646](#)

F

fail_on_error_mesg [102](#)
fall [1056](#), [1316](#), [1325](#)
fall_capacitance_range [215](#)
fanout_cap [234](#)
fanout_load [216](#)
file_name [1279](#)
files [181](#)
find_fuzzy_match [103](#)
find_help_ls_style [104](#)
find_inefficient_threshold [104](#)
find_inefficient_use [104](#)
fix_min_drcs [536](#)
fixed [1279](#)
fixed_slew [647](#)
fixedbump [464](#), [483](#)
flip [503](#), [509](#), [513](#)
flop [196](#), [948](#)
force_merge_combos_into_multibit_cells
[718](#)
force_merge_seqs_into_multibit_cells [719](#)
force_via_resistance [376](#)
force_wireload [547](#), [666](#)
formula [167](#)
fplan_height [391](#), [407](#)
fplan_width [391](#), [407](#)
frequency [464](#)
from_lef [450](#)
from_pin [229](#)
from_points [1065](#)

from_power_domain [1454](#), [1458](#)
function [216](#), [1310](#)

G

gen_module_prefix [720](#)
generated_clock [216](#)
ground [217](#)
ground_direction [196](#), [1431](#)
ground_voltage [1461](#)
group [867](#), [872](#)
group_generate_portname_from_netname
 [248](#)
group_instance_suffix [250](#)
groups [392](#)
gui_auto_update [142](#)
gui_enabled [142](#)
gui_hv_phys_threshold [142](#)
gui_hv_threshold [143](#)
gui_sv_threshold [143](#)
gui_sv_update [143](#)
gui_visible [144](#)

H

hard_region [567](#), [667](#)
hardspacing [451](#)
has_external_shutoff [1464](#)
has_fanout [1326](#)
has_fills [415](#)
has_min_delay [971](#), [1009](#)
has_opcg_segments [1255](#)
has_slots [416](#)
hdl_all_filelist [319](#), [339](#)
hdl_allow_inout_const_port_connect [250](#)
hdl_allow_instance_name_conflict [251](#)
hdl_append_generic_ports [720](#), [721](#)
hdl_array_naming_style [722](#)
hdl_async_set_reset [722](#)
hdl_auto_async_set_reset [723](#)
hdl_auto_exec_sdc_scripts [723](#)
hdl_auto_sync_set_reset [725](#)
hdl_bidirectional_assign [725](#)
hdl_bus_wire_naming_style [726](#)
hdl_case_mux_threshold [727](#)
hdl_case_sensitive_instances [728](#)
hdl_config_name [320](#), [340](#)
hdl_create_label_for_unlabeled_generate
 [729](#)

hdl_cw_list [837](#), [905](#)
hdl_decimal_parameter_name [730](#)
hdl_delete_transparent_latch [732](#)
hdl_elab_command_params [934](#), [1032](#)
hdl_elab_param_override [934](#), [1032](#)
hdl_enable_non_default_library_domain_bounding [732](#)
hdl_enable_proc_name [732](#)
hdl_error_on_blackbox [733](#), [860](#)
hdl_error_on_latch [733](#), [861](#)
hdl_error_on_logic_abstract [734](#), [861](#)
hdl_error_on_negedge [735](#), [862](#)
hdl_ff_keep_explicit_feedback [735](#), [862](#)
hdl_ff_keep_feedback [736](#), [863](#)
hdl_filelist [321](#), [341](#)
hdl_flatten_complex_port [737](#)
hdl_generate_index_style [739](#)
hdl_generate_separator [742](#)
hdl_ignore_pragma_names [252](#)
hdl_index_mux_threshold [744](#)
hdl_infer_unresolved_from_logic_abstract
 [744](#)
hdl_instance_array_naming_style [745](#)
hdl_instantiated [948](#)
hdl_interface_separator [747](#)
hdl_keep_first_module_definition [252](#)
hdl_language [252](#)
hdl_latch_keep_feedback [748](#), [863](#)
hdl_link_from_any_lib [253](#)
hdl_max_loop_limit [749](#)
hdl_max_map_to_mux_control_width [750](#)
hdl_max_memory_address_range [253](#)
hdl_max_recursion_limit [750](#)
hdl_nc_compatible_module_linking [253](#)
hdl_parameter [166](#)
hdl_parameter_naming_style [750](#)
hdl_parameters [935](#), [1033](#)
hdl_pipeline_comp [1034](#)
hdl_preserve_async_sr_priority_logic [752](#)
hdl_preserve_dangling_output_nets [753](#)
hdl_preserve_supply_nets [753](#)
hdl_preserve_sync_ctrl_logic [754](#)
hdl_preserve_sync_set_reset [755](#)
hdl_preserve_unused_flop [755](#), [864](#)
hdl_preserve_unused_latch [757](#), [865](#)
hdl_preserve_unused_registers [759](#), [865](#)
hdl_primitive_input_multibit [255](#)
hdl_proc_name [843](#)
hdl_record_naming_style [762](#)
hdl_reg_naming_style [762](#)
hdl_rename_cdn_flop_pins [764](#)

hdl_report_case_info [256](#)
 hdl_search_path [256](#)
 hdl_sv_module_wrapper [765](#)
 hdl_sync_set_reset [765](#)
 hdl_track_filename_row_col [257](#)
 hdl_track_module_elab_memory_and_runti
me [766](#)
 hdl_unconnected_input_port_value [767](#)
 hdl_undriven_output_port_value [771](#)
 hdl_undriven_signal_value [773](#)
 hdl_use_block_prefix [775](#)
 hdl_use_current_dir_before_hdl_search_pa
th [258](#)
 hdl_use_cw_first [776](#)
 hdl_use_default_parameter_values_in_desi
gn_name [776](#)
 hdl_use_default_parameter_values_in_nam
e [777](#)
 hdl_use_for_generate_prefix [778](#)
 hdl_use_if_generate_prefix [779](#)
 hdl_use_parameterized_module_by_name
[781](#)
 hdl_use_port_default_value [783](#)
 hdl_use_techelt_first [783](#)
 hdl_user_name [323, 342](#)
 hdl_v2001 [323, 326, 328, 329, 336, 342,
346](#)
 hdl_vdp_list [936, 1034](#)
 hdl_verilogDefines [259](#)
 hdl_vhdl_assign_width_mismatch [260](#)
 hdl_vhdl_case [261](#)
 hdl_vhdl_environment [261](#)
 hdl_vhdl_lrm_compliance [262](#)
 hdl_vhdl_preferred_architecture [262](#)
 hdl_vhdl_range_opto [263](#)
 hdl_vhdl_read_version [264](#)
 hdl_zero_replicate_is_null [264](#)
 head [1287](#)
 head_lockup [1256](#)
 height [197, 411, 422, 454, 475](#)
 help [131, 139](#)
 help_always_visible [131](#)
 hier_path [507](#)
 hierarchical [948](#)
 higher_drive [217](#)
 highz [1172, 1189](#)
 hold_uncertainty_by_clock [598, 648](#)
 hookup_period [1238](#)
 horizontal_remaining [437](#)

I
 id [131, 1280](#)
 ideal [999, 1326](#)
 ideal_driver [599, 649](#)
 ideal_network [600, 649](#)
 ideal_seq_async_pins [548](#)
 ignore_external_driver_drc [650](#)
 ignore_library_drc [549](#)
 ignore_library_max_fanout [550](#)
 ignore_preserve_in_tiecell_insertion [784](#)
 ignore_scan_combinational_arcs [536](#)
 incoming_timing_arcs [217](#)
 index [490, 503, 509, 513, 1202, 1310](#)
 infer_multibit [844](#)
 information_level [105](#)
 inherited_preserve [845](#)
 init_core_utilization [392](#)
 input [217](#)
 input_assert_one_coldPragma [265](#)
 input_assert_one_hotPragma [265](#)
 input_asynchro_reset_blkPragma [265](#)
 input_asynchro_resetPragma [266](#)
 input_case_coverPragma [267](#)
 input_case_decodePragma [267](#)
 input_delay [1071](#)
 input_map_to_muxPragma [268](#)
 inputPragma_keyword [268](#)
 input_synchro_enable_blkPragma [269](#)
 input_synchro_enable_Pragma [269](#)
 input_synchro_reset_blkPragma [269](#)
 input_synchro_reset_Pragma [270](#)
 input_threshold_pct_fall [181](#)
 input_threshold_pct_rise [182](#)
 inside_inst [1246](#)
 inst_prefix [271](#)
 instance [510, 1173, 1189, 1268, 1301](#)
 instance_count [438](#)
 instance_internal_power [1368](#)
 instance_leakage_power [1369](#)
 instance_probability [197](#)
 instances [438, 1035](#)
 interconnect_mode [377](#)
 internal [217, 1238](#)
 inverted_phase [948](#)
 inverted_sources [677](#)
 inverter [197, 949](#)
 iocell_enable [1129](#)
 iocell_input [1129](#)
 iocell_output [1130](#)

iopt_allow_tiecell_with_inversion [785](#)
 iopt_avoid_tiecell_replacement [884](#), [890](#),
 [914](#)
 iopt_enable_floating_output_check [786](#)
 iopt_enable_parallelization [786](#)
 iopt_force_constant_removal [787](#)
 iopt_remap_avoided_cells [787](#)
 iopt_sequential_duplication [787](#)
 iopt_sequential_resynthesis [788](#)
 iopt_sequential_resynthesis_min_effort [78](#)
 [8](#)
 iopt_temp_directory [789](#)
 iopt_ultra_optimization [789](#)
 is_always_on [198](#), [218](#)
 is_clock_gating_cell [198](#)
 is_exceptpgnet [416](#)
 is_horizontal [475](#), [491](#)
 is_iq_function [218](#)
 is_iqn_function [218](#)
 is_isolation_cell [198](#)
 is_itag_tck [1238](#)
 is_level_shifter [198](#)
 is_net_part_of_bus [1000](#)
 is_partial [416](#)
 is_pushdown [417](#)
 is_soft [417](#)
 is_sub_bin [1046](#)
 is_user [1482](#)
 isolation_cell_enable_pin [218](#)
 isolation_rule [1441](#), [1446](#), [1450](#)

J

jtag_controlled [1245](#)
 jtag_instruction [1210](#)
 justify_by [504](#), [510](#), [514](#)

L

language [156](#), [937](#), [1035](#)
 latch [199](#), [949](#)
 latch_borrow [551](#), [568](#)
 latch_borrow_by_mode [552](#), [569](#)
 latch_enable [199](#)
 latch_enable_phase [218](#)
 latch_max_borrow [554](#), [571](#), [601](#), [677](#)
 latch_max_borrow_by_mode [572](#)
 launcher [971](#), [1009](#)
 layer [417](#), [434](#), [481](#), [492](#)

layers [428](#), [451](#)
 lbr_respect_async_controls_priority [271](#)
 lbr_seq_in_out_phase_opto [790](#)
 leakage_power_effort [1338](#)
 leakage_power_scale_in_nW [182](#)
 lec_executable [271](#)
 lec_value [1327](#)
 lef_add_logical_pins [272](#)
 lef_library [272](#)
 lef_manufacturing_grid [377](#)
 lef_stop_on_error [377](#)
 lef_units [378](#)
 legal [156](#)
 length [1177](#), [1185](#)
 lenient [1065](#)
 level_sensitive [1071](#)
 level_shifter_cells [1432](#)
 level_shifter_enable_pin [219](#)
 level_shifter_rule [1441](#), [1447](#), [1450](#)
 lib_lef_consistency_check_enable [378](#)
 lib_search_path [273](#)
 libcell [949](#)
 liberty_attributes [182](#), [199](#), [219](#), [229](#), [232](#),
 [235](#)
 liberty_level_shifter_type [200](#)
 libpin [972](#), [994](#)
 library [274](#), [1427](#)
 library_domain [1437](#), [1443](#), [1444](#), [1448](#),
 [1465](#)
 library_domain_by_mode [1465](#)
 library_name [937](#), [1036](#)
 library_set [1462](#)
 limit_lbr_messages [106](#)
 limited_acces_feature [106](#)
 line_number [1281](#)
 llx [410](#)
 loads [972](#), [994](#), [1000](#), [1010](#), [1041](#)
 location [148](#), [153](#), [157](#), [333](#), [335](#), [1242](#),
 [1454](#), [1458](#)
 location_x [399](#), [402](#), [405](#), [411](#), [418](#), [423](#),
 [429](#), [455](#), [473](#), [476](#)
 location_y [399](#), [402](#), [405](#), [411](#), [418](#), [423](#),
 [429](#), [455](#), [473](#), [476](#)
 log_command_error [106](#)
 logic_abstract [937](#), [1036](#)
 logical_hier [1037](#)
 loop_breaker_is_ignored_for_all_modes [9](#)
 [50](#)
 lower_drive [219](#)
 lp_asserted_probability [1380](#), [1389](#), [1396](#),
 [1411](#)

lp_asserted_toggle_rate [1381](#), [1390](#), [1396](#), [1412](#)
 lp_clock_gating_add_obs_port [1349](#)
 lp_clock_gating_add_reset [1350](#), [1369](#), [1370](#), [1404](#)
 lp_clock_gating_auto_cost_group_initial_target [1351](#)
 lp_clock_gating_auto_cost_grouping [1351](#)
 lp_clock_gating_auto_path_adjust [1352](#), [1405](#)
 lp_clock_gating_auto_path_adjust_fixed_delay [1352](#), [1406](#)
 lp_clock_gating_auto_path_adjust_modes [1353](#)
 lp_clock_gating_auto_path_adjust_multiple [1354](#), [1406](#)
 lp_clock_gating_cell [1354](#), [1370](#), [1407](#)
 lp_clock_gating_control_point [1355](#)
 lp_clock_gating_exceptions_aware [1339](#)
 lp_clock_gating_exclude [1357](#), [1371](#), [1408](#)
 lp_clock_gating_extract_common_enable [1357](#)
 lp_clock_gating_gated_clock_gates [1372](#)
 lp_clock_gating_gated_flops [1372](#)
 lp_clock_gating_infer_enable [1339](#)
 lp_clock_gating_is_flop_rc_gated [1373](#)
 lp_clock_gating_is_flop_user_gated [1373](#)
 lp_clock_gating_is_leaf_clock_gate [1373](#)
 lp_clock_gating_max_flops [1358](#)
 lp_clock_gating_min_flops [1358](#)
 lp_clock_gating_module [1359](#), [1374](#), [1409](#)
 lp_clock_gating_prefix [1340](#)
 lp_clock_gating_rc_inserted [1374](#)
 lp_clock_gating_register_aware [1341](#)
 lp_clock_gating_stage [1374](#)
 lp_clock_gating_style [1359](#)
 lp_clock_gating_test_signal [1360](#), [1375](#), [1409](#)
 lp_clock_tree_buffers [1361](#)
 lp_clock_tree_leaf_max_fanout [1361](#)
 lp_computed_probability [1381](#), [1390](#), [1397](#), [1412](#)
 lp_computed_toggle_rate [1382](#), [1391](#), [1398](#), [1413](#)
 lp_default_probability [1341](#), [1361](#), [1376](#), [1383](#), [1399](#)
 lp_default_toggle_percentage [1362](#), [1417](#)
 lp_default_toggle_rate [1342](#), [1363](#), [1377](#), [1384](#), [1400](#)
 lp_display_negative_internal_power [1342](#)

lp_dynamic_analysis_scope [1377](#)
 lp_insert_clock_gating [1343](#)
 lp_internal_power [1363](#), [1378](#)
 lp_isolate_domain_crossing_constants [14](#), [38](#)
 lp_leakage_power [1364](#), [1378](#)
 lp_net_power [1364](#), [1379](#), [1385](#), [1392](#), [1401](#), [1414](#)
 lp_power_analysis_effort [1344](#)
 lp_power_optimization_weight [1365](#)
 lp_power_unit [1345](#)
 lp_probability_type [1386](#), [1393](#), [1402](#), [1415](#)
 lp_pso_aware_estimation [1346](#)
 lp_pso_aware_tcf [1365](#)
 lp_toggle_rate_type [1387](#), [1394](#), [1403](#), [1416](#)
 lp_toggle_rate_unit [1346](#)
 lp_x_transition_probability_count [1347](#)
 lp_x_transition_toggle_count [1347](#)
 lp_z_transition_probability_count [1348](#)
 lp_z_transition_toggle_count [1348](#)
 lssd_master_clock [885](#)

M

macro [477](#), [492](#)
 map_drc_first [791](#)
 map_latch_allow_async_decomp [792](#)
 map_respect_rtl_clk_phase [792](#)
 map_to_master_slave_lssd [793](#)
 map_to_module [875](#)
 map_to_multibit_bank_label [846](#)
 map_to_multibit_register [847](#)
 map_to_multiple_output_gates [794](#)
 map_to_mux [848](#)
 map_to_operator [875](#)
 map_to_register [848](#)
 mask [418](#), [435](#), [493](#)
 master_signal [1327](#)
 master_slave_flop [200](#)
 master_slave_lssd_flop [200](#)
 max [681](#)
 max_cap_cost [938](#)
 max_capacitance [220](#), [557](#), [651](#)
 max_cpus_per_server [107](#)
 max_dynamic_power [1366](#)
 max_fanout [220](#), [558](#), [652](#)
 max_fanout_cost [938](#)
 max_ground_input_voltage [200](#), [1432](#)

Attribute Reference for Encounter RTL Compiler

max_ground_output_voltage [201](#), [1433](#)
max_input_period [1248](#)
max_input_voltage [201](#), [1433](#)
max_layer [419](#)
max_leakage_power [1367](#)
max_length [1287](#)
max_num_pulses [1240](#)
max_output_period [1248](#)
max_output_voltage [202](#), [1433](#)
max_print [132](#)
max_super_thread_cache_size [107](#)
max_trans_cost [939](#)
max_transition [220](#), [559](#), [653](#)
mbist_enable_shared_library_domain_set
 [1130](#)
mbist_instruction_set [1139](#)
members [441](#)
memory_libcell [1217](#)
memory_usage [108](#)
merge_combinational_hier_instance [849](#)
merge_combinational_hier_instances [794](#)
merge_multibit_power_area_based [794](#)
min_cap_cost [939](#)
min_capacitance [221](#)
min_domain_period [1243](#)
min_fanout [221](#)
min_fanout_cost [940](#)
min_gaps [470](#)
min_ground_input_voltage [202](#), [203](#), [1434](#)
min_input_period [1249](#)
min_input_voltage [203](#), [1435](#)
min_layer [419](#)
min_output_period [1249](#)
min_output_voltage [204](#), [1435](#)
min_port_delay [1011](#)
min_slew [972](#), [1010](#)
min_spacing [446](#)
min_timing_arcs [973](#)
min_trans_cost [940](#)
min_transition [221](#)
mincuts [452](#)
mingap [461](#)
minimize_uniquify [796](#), [906](#)
mode [1049](#)
mode_a [1190](#)
mode_b [1190](#)
mode_c [1190](#)
mode_enable_high [1210](#)
mode_enable_low [1211](#)
mode_init [1245](#)
mode_name [1256](#)

modea [1173](#)
modeb [1174](#)
modec [1174](#)
model [423](#), [456](#)
module [950](#)
multibit_allow_async_phase_map [796](#),
 [849](#), [907](#)
multibit_allow_unused_bits [798](#)
multibit_cells_from_different_busses [798](#)
multibit_debug [799](#)
multibit_mapping_effort_level [800](#)
multibit_predefined_allow_unused_bits [80](#)
 [1](#)
multibit_prefix_string [801](#)
multibit_preserve_inferred_instances [802](#)
multibit_preserved_net_check [803](#)
multibit_seqs_instance_naming_style [803](#)
multibit_seqs_members_naming_style [80](#)
 [5](#)
multibit_seqs_name_concat_string [807](#)
multibit_short_prefix_string [806](#)
multibit_skip_exception_check [807](#)
multibit_split_string [808](#)
multibit_unused_input_value [809](#)

N

name [464](#), [484](#)
nc_protect_version [275](#)
negative_edge_clock [950](#)
net [973](#), [994](#), [1011](#), [1041](#)
net_area [940](#), [1037](#)
net_expr [429](#)
net_name [430](#)
network_early_latency_by_clock [604](#), [654](#)
network_late_latency_by_clock [605](#), [655](#)
newlink physical_del [405](#)
non_inverted_sources [677](#)
non_seq_setup_arc [204](#)
nondefaultrule [465](#)
num_drivers [1000](#)
num_loads [1000](#)
number_of_routing_layers [393](#)

O

obsolete [153](#), [157](#)
obstruction_routing_layer [393](#)
off_state [1316](#)

offset [446](#)
 offset_x [446](#)
 offset_y [446](#)
 opc [435](#)
 opcg_trigger [1243](#)
 opcode [1178](#)
 operating_conditions [537](#), [1429](#)
 operator [149](#)
 optimize_constant_0_flops [810](#)
 optimize_constant_0_seq [852](#)
 optimize_constant_1_flops [810](#)
 optimize_constant_1_seq [853](#)
 optimize_constant_feedback_seq [853](#)
 optimize_constant_latches [810](#)
 optimize_merge_flops [811](#)
 optimize_merge_latches [811](#)
 optimize_merge_seq [854](#)
 optimize_net_area [812](#)
 optimize_seq_x_to [812](#)
 optimize_yield [518](#)
 order [1029](#), [1045](#)
 orient [504](#), [507](#), [510](#), [514](#)
 orientation [424](#), [430](#), [456](#), [477](#)
 origin [508](#)
 original_name [465](#), [484](#)
 osc_source [1243](#), [1247](#)
 osc_source_period [1251](#)
 other_clocks [1256](#), [1268](#), [1301](#)
 outgoing_timing_arcs [222](#)
 output [222](#)
 output_threshold_pct_fall [183](#)
 output_threshold_pct_rise [183](#)
 output_value [1455](#)
 override_library_max_drc [537](#)

P

pad [204](#), [222](#)
 param_association [150](#)
 parameters [154](#), [330](#)
 partial_row_order [1214](#)
 path [275](#)
 path_count [465](#), [484](#), [1046](#)
 path_index [465](#), [485](#)
 path_value [466](#), [485](#)
 paths [1066](#)
 pattern [466](#), [485](#)
 pcells [393](#)
 peak_memory [108](#)
 period [1056](#), [1239](#), [1317](#), [1328](#)

permutable_group [168](#)
 phys_annotation_ndr_nets [378](#)
 phys_checkout_encounter_license [379](#)
 phys_fix_multi_height_cells [379](#)
 phys_flow_effort [380](#)
 phys_ignore_nets [395](#)
 phys_ignore_special_nets [395](#)
 phys_legalization_enhancement [380](#)
 phys_legalize [381](#)
 phys_mp_constraints [381](#)
 phys_pre_place_iopt [382](#)
 phys_premorph_density [383](#)
 physical [400](#), [402](#)
 physical_aware_mapping [395](#)
 physical_aware_restructuring [396](#), [408](#)
 physical_aware_structuring [396](#), [408](#)
 physical_cap [404](#)
 physical_cell_area [941](#), [1037](#)
 physical_del [400](#), [405](#)
 physical_res [400](#), [406](#)
 pi_relax_map_iso_cell_checks [1438](#)
 pi_relax_map_ls_cell_checks [1439](#)
 pin [1202](#), [1207](#), [1247](#), [1250](#), [1310](#), [1328](#)
 pin_association [150](#)
 pin_capacitance [957](#), [974](#), [1012](#), [1041](#)
 pin_count [439](#), [941](#), [951](#), [1038](#)
 pin_density [439](#)
 pinmap [1202](#)
 pins [154](#), [331](#), [439](#), [1455](#), [1459](#)
 pitch [447](#)
 pitch_x [447](#)
 pitch_y [447](#)
 placement_status [401](#), [403](#), [406](#), [412](#), [424](#),
[430](#), [456](#)
 platform_wordsize [108](#)
 pmbist_enable_multiple_views [1114](#)
 pmbist_instruction_set [1140](#)
 pmbist_use [1329](#)
 polarity [223](#)
 polygon [490](#)
 polygons [419](#), [431](#), [435](#), [482](#), [486](#), [497](#)
 por [1191](#)
 port_delay [1012](#)
 ports [431](#)
 power [223](#)
 power_domain [1257](#), [1269](#), [1301](#), [1439](#),
[1440](#), [1442](#), [1447](#), [1448](#), [1451](#)
 power_gating_cell [205](#)
 power_gating_cell_type [205](#)
 power_gating_pin_class [223](#)
 power_gating_pin_phase [224](#)

Attribute Reference for Encounter RTL Compiler

power_library [1429](#)
power_rails [184](#)
pqos_ignore_msv [381](#)
pqos_ignore_scan_chains [382](#)
pqos_placement_effort [382](#)
pre_elab_script [158](#)
precluded_path_adjusts [1066](#)
preferred_comp [162](#)
preferred_impl [160, 163](#)
prefix [1455, 1459](#)
preroute_as_obstruction [394](#)
preserve [206, 838, 855, 876, 886, 888, 909](#)
preserve_techelts [159](#)
primitive_function [951](#)
print_count [132](#)
print_error_info [108](#)
priority [132, 150, 159, 1066](#)
private [1178](#)
process [232](#)
program_major_version [109](#)
program_name [110](#)
program_short_name [110](#)
program_version [110](#)
propagate_constant_from_timing_model [8
12, 856](#)
propagated_clocks [974, 1013](#)
propagated_clocks_by_mode [976, 1015](#)
propagated_ideal_network [978](#)
properties [425, 442, 452, 457, 466, 474,
486](#)
protected [324, 331, 343](#)
proto_feasible_target [813](#)
proto_feasible_target_adjust_slack_pct [81
3](#)
proto_feasible_target_threshold [814](#)
proto_feasible_target_threshold_clock_pct
[814](#)
proto_hdl [815](#)
prune_unused_logic [887](#)

Q

q_pin_of_d_pin [224](#)
qos_report_power [383](#)
qrc_tech_file [384](#)

R

rc_create_verification_directory [276](#)

rc_name [467, 486](#)
rc_verification_directory [276](#)
read_delay [1217](#)
real_enabled [230](#)
real_runtime [110](#)
rectangles [486](#)
ref_clk_period [1251](#)
ref_clock_pin [1250](#)
reg_count [1257, 1270, 1281, 1302](#)
region [442](#)
regions [396](#)
register [1178](#)
register_captureddr [1179](#)
register_clockdr [1179](#)
register_decode [1180](#)
register_reset [1180](#)
register_reset_polarity [1180](#)
register_runidle [1181](#)
register_shiftdr [1181](#)
register_shiftdr_polarity [1181](#)
register_tck [1182](#)
register_tdi [1182](#)
register_tdo [1182](#)
register_updateddr [1183](#)
registers [1282](#)
remove_assigns [816](#)
reorderable [1270, 1303](#)
report_as_datapath [154](#)
report_library_message_summary [111](#)
report_tcl_command_error [111](#)
report_timing_show_wire_length [928](#)
reset [1191](#)
resistance [448](#)
restore [1469](#)
restore_history_file [112](#)
restore_phase [1469](#)
retime [839, 910](#)
retime_async_reset [817](#)
retime_effort_level [817](#)
retime_hard_region [910](#)
retime_move_mux_loop_with_reg [818](#)
retime_optimize_reset [819](#)
retime_original_registers [951](#)
retime_reg_naming_suffix [819](#)
retime_verification_flow [820](#)
retiming_clocks [821](#)
return_port [875](#)
rf_slack [979, 1017](#)
rise [1056, 1317, 1330](#)
rise_capacitance_range [225](#)
root [1047](#)

root_node [1283](#)
root_source_pins [1317](#)
root_source_polarity [1318](#)
row_order [1215](#)
rs_ext [471](#)
rsext [462](#)
runidle [1191, 1192](#)

S

same_mask [493](#)
save [1469](#)
save_history_file [113](#)
save_phase [1469](#)
scale_factor_group_path_weights [538](#)
scale_of_cap_per_unit_length [384](#)
scale_of_res_per_unit_length [385](#)
scan_clock [1244, 1247](#)
scan_clock_a [1257, 1271, 1288, 1303](#)
scan_clock_b [1258, 1271, 1288, 1303](#)
scan_enable [206](#)
scan_enable_phase [225](#)
scan_in [207, 1258, 1271, 1289, 1304](#)
scan_in_phase [225](#)
scan_out [1259, 1272, 1289, 1304](#)
scan_shift [1330](#)
screen_max_print [133](#)
screen_print_count [133](#)
script_begin [278](#)
script_end [278](#)
script_search_path [279](#)
sdi_compression_signal [1259, 1289](#)
sdp_files [397](#)
sdp_type [397](#)
secondary_domain [1449, 1456, 1470](#)
segments [1283](#)
selected_impl [171](#)
seq_reason_deleted [941](#)
sequential [207, 952](#)
set_function [139](#)
settable [140](#)
setup_uncertainty_by_clock [607, 657](#)
severity [133](#)
shared_output [1260, 1290](#)
shared_select [1260, 1290](#)
shieldnet [467](#)
shift_capture [1067](#)
shift_enable [1260, 1273, 1290, 1305](#)
shift_launch [1067](#)
shiftdr [1175](#)

show_command_usage [114](#)
show_report_options [115](#)
show_wns_in_log [929](#)
shrink_factor [385](#)
shutoff_condition [1466](#)
signal_level [225](#)
signed [165, 168](#)
sim_model [155](#)
size_fixed [511](#)
size_same [504, 514](#)
skew_safe [1273, 1305](#)
skip_in_write_def [412](#)
skip_value [505, 511, 515](#)
slack [942, 952, 980, 1017, 1050, 1060](#)
slack_by_mode [943, 980, 1018, 1060](#)
slew [678, 982, 1019](#)
slew_by_mode [982, 1019](#)
slew_derate_from_library [184](#)
slew_lower_threshold_pct_fall [185](#)
slew_lower_threshold_pct_rise [185](#)
slew_upper_threshold_pct_fall [185](#)
slew_upper_threshold_pct_rise [186](#)
source [467, 487, 1235, 1236](#)
source_early_latency_by_clock [609](#)
source_late_latency_by_clock [610](#)
source_suspend_on_error [116](#)
source_verbose [115](#)
source_verbose_info [116](#)
source_verbose_proc [117](#)
sources [1239, 1318](#)
spacing [420](#)
special [432](#)
speed_grade [171](#)
start [493](#)
start_source_line [331](#)
startpoint [983, 1020, 1050](#)
startup_license [117](#)
state [129](#)
state_retention_rule [1449](#)
statistics_db_file [118](#)
statistics_db_runtime [118](#)
statistics_enable_power_report [119](#)
statistics_log_data [119](#)
statistics_run_description [120](#)
statistics_run_id [121](#)
stdout_log [121](#)
step [494](#)
stop_at_ipt_state [822](#)
structural [332](#)
style [487](#)
sub_arch [172](#)

subdesign [953](#)
 super_thread_batch_command [122](#)
 super_thread_cache [123](#)
 super_thread_debug_directory [123](#)
 super_thread_equivalent_licenses [124](#)
 super_thread_kill_command [125](#)
 super_thread_peak_memory [126](#)
 super_thread_rsh_command [126](#)
 super_thread_servers [127](#)
 super_thread_status_command [128](#)
 support_appending_libs [280](#)
 support_multi_seq_elements [280](#)
 sync_clear [207](#)
 sync_clear_phase [226](#)
 sync_enable [207](#)
 sync_enable_phase [226](#)
 sync_preset [208](#)
 sync_preset_phase [226](#)
 synthesis_off_command [281](#)
 synthesis_on_command [281](#)
 sys_enable [1203](#)
 sys_use [1203](#)
 systematic_probability [208](#)

T

tail [1291](#)
 tail_clock [1274, 1305](#)
 tail_clock_edge [1274, 1306](#)
 tap_decode [1183](#)
 tap_tdi [1184](#)
 tap_tdo [1184](#)
 target_period [1241](#)
 tck [1192](#)
 tcl_result_truncation_length [128](#)
 tdi [1175, 1192](#)
 tdo [1176, 1193](#)
 tdo_enable [1193](#)
 technology [159](#)
 temperature [233](#)
 terminal_lockup [1261, 1291](#)
 test_use [1204](#)
 through_points [1067](#)
 tim_ignore_data_check_for_non_endpoint_pins [538](#)
 time_recovery_arcs [539](#)
 time_scale_in_ps [186](#)
 timing_arcs [983](#)
 timing_case_computed_value [983, 1020](#)
 timing_case_computed_value_by_mode [9](#)

[85, 1007, 1018, 1021](#)
 timing_case_disabled_arcs [953](#)
 timing_case_disabled_arcs_by_mode [954](#)
 timing_case_logic_value [612, 663](#)
 timing_case_logic_value_by_mode [613, 664](#)
 timing_disable_internal inout_net_arcs [560](#)
 timing_disable_non_sequential_checks [539](#)
 timing_info [987, 1023](#)
 timing_info_favor_startpoint [990, 1025](#)
 timing_model [208](#)
 timing_model_reason [209](#)
 timing_no_path_segmentation [540](#)
 tms [1193](#)
 tns [944, 1062](#)
 tns_opto [822](#)
 to_points [1068](#)
 to_power_domain [1456, 1459](#)
 top_layer [497](#)
 total_net_length [945](#)
 tr_bdy_in [1204](#)
 trace_retime [857](#)
 trcell [1204](#)
 trcell_acmode [1205](#)
 trcell_clock [1205](#)
 trcell_enable [1205](#)
 tree_type [233](#)
 trigger_delay [1241](#)
 tristate [209, 226, 955](#)
 tristate_net_drivers [1284](#)
 tristate_net_load [1284](#)
 trst [1194](#)
 truncate [134](#)
 type [134, 230, 420, 448, 474, 488, 1001, 1206, 1208, 1275, 1284, 1307, 1331](#)

U

ui_respects_preserve [1473](#)
 ultra_global_mapping [823](#)
 unbound_oper_pin [151](#)
 undesirable_libcells [840, 858](#)
 ungroup [866, 869, 870, 871](#)
 ungroup_ok [858, 911](#)
 ungroup_separator [282](#)
 unique_versions [956, 958, 991, 994, 1001, 1025, 1042](#)
 uniquify_naming_style [824](#)

units [140](#)
 unmap_scan_flops [1114](#)
 unresolved [859](#)
 unusable_reason [209](#)
 updatedr [1176, 1194](#)
 urx [413, 425, 457](#)
 ury [413, 426, 458](#)
 usable [210](#)
 usable_comb_cells [186](#)
 usable_seq_cells [187](#)
 usable_timing_models [187](#)
 usage [1212](#)
 use [227, 432, 468, 488](#)
 use_area_from_lef [386](#)
 use_compatibility_based_grouping [825](#)
 use_multi_clks_latency_uncertainty_optimiz
e [541](#)
 use_multi_clks_latency_uncertainty_report
541
 use_multibit_cells [826](#)
 use_multibit_combo_cells [827](#)
 use_multibit_seq_and_tristate_cells [828](#)
 use_nextstate_type_only_to_assign_sync
ctrls [829](#)
 use_power_ground_pin_from_lef [282](#)
 use_scan_seqs_for_non_dft [829](#)
 use_tiehilo_for_const [830](#)
 user_created [420, 443, 474, 477](#)
 user_defined [210, 227, 1213, 1474, 1475,
1476, 1477, 1478, 1479, 1480, 1481](#)
 user_defined_macro [1194](#)
 user_defined_segment [1308](#)
 user_defined_signal [1319, 1331](#)
 user_differential_negative_pin [1146](#)
 user_from_core_data [1146](#)
 user_from_core_enable [1147](#)
 user_function [227](#)
 user_priority [681](#)
 user_speed_grade [172](#)
 user_sub_arch [911](#)
 user_test_receiver_acmode [1148](#)
 user_test_receiver_data_output [1148](#)
 user_test_receiver_init_clock [1148](#)
 user_test_receiver_init_data [1148](#)
 user_to_core_data [1149](#)
 user_to_core_enable [1149](#)
 utilization [397, 440, 448](#)
 utilization_threshold [398](#)

V

valid_location [1435, 1436](#)
 value [1219](#)
 verilog_macros [332](#)
 version [187](#)
 vertical_remaining [440](#)
 via [436](#)
 via_mask [436](#)
 via_opc [436](#)
 via_points [436](#)
 via_resistance [386](#)
 viarule_name [497](#)
 viarules [453](#)
 vias [433, 453](#)
 visible [421, 433, 449, 468, 478](#)
 voltage [233, 489, 1462](#)

W

waveform [1057](#)
 wccd_threshold_percentage [282](#)
 wcdc_clock_dom_comb_propagation [283](#)
 wcdc_synchronizer_type [324](#)
 wlcp_lib_statetable [284](#)
 weight [426, 458, 468, 489, 679](#)
 width [211, 413, 421, 427, 449, 459, 478](#)
 wire_capacitance [958, 991, 1026, 1042](#)
 wire_length [959, 992, 1026, 1043](#)
 wire_resistance [959, 992, 1027, 1043](#)
 wireload [945, 1038](#)
 wireload_mode [542](#)
 wireload_model [992](#)
 wireload_selection [543, 1430](#)
 within_hierarchy [1456, 1460](#)
 wlec_add_noblock_box_retime_subdesign
284
 wlec_analyze_abort [285](#)
 wlec_analyze_setup [286](#)
 wlec_auto_analyze [286](#)
 wlec_compare_threads [287, 289](#)
 wlec_cut_point [288](#)
 wlec_hier_comp_threshold [288](#)
 wlec_lib_statetable [289](#)
 wlec_set_cdn_synth_root [290](#)
 wlec_uniquify [290](#)
 wlec_use_lec_model [291](#)
 wrapper [1217](#)
 wrapper_control [1150, 1158, 1166](#)

wrapper_segment [1150, 1159, 1167](#)
write_sv_port_wrapper [292](#)
write_vlog_bit_blast_bus_connections [29](#)
 [3](#)
write_vlog_bit_blast_constants [294](#)
write_vlog_bit_blast_mapped_ports [296](#)
write_vlog_bit_blast_tech_cell [298](#)
write_vlog_convert_onebit_vector_to_scala
 r [301](#)
write_vlog_declare_wires [302](#)
write_vlog_empty_module_for_black_box
 [304](#)
write_vlog_empty_module_for_logic_abstra
 ct [305](#)
write_vlog_empty_module_for_subdesign
 [344](#)
write_vlog_generic_gate_define [307](#)
write_vlog_line_wrap_limit [308](#)
write_vlog_no_negative_index [309](#)
write_vlog_port_association_style [327](#)
write_vlog_preserve_net_name [310](#)
write_vlog_top_module_first [311](#)
write_vlog_unconnected_port_style [312](#)
write_vlog_wor_wand [314](#)

X

x_offset [228](#)
xbottom_enclosure [498](#)
xbottom_offset [498](#)
xcut_size [498](#)
xcut_spacing [499](#)
xorigin_offset [499](#)
xtalk [469](#)
xtop_enclosure [499](#)
xtop_offset [500](#)

Y

y_offset [228](#)
ybottom_enclosure [500](#)
ybottom_offset [500](#)
ycut_size [501](#)
ycut_spacing [501](#)
yield [519](#)
yorigin_offset [501](#)
ytop_enclosure [502](#)
ytop_offset [502](#)

Preface

- [About This Manual](#) on page 86
- [Additional References](#) on page 86
- [How to Use the Documentation Set](#) on page 87
- [Reporting Problems or Errors in Manuals](#) on page 88
- [Customer Support](#) on page 89
- [Messages](#) on page 90
- [Man Pages](#) on page 91
- [Command-Line Help](#) on page 92
- [Documentation Conventions](#) on page 94

About This Manual

This manual provides a concise reference of the attributes available to the user when using the Encounter™ RTL Compiler software.

Attributes can be used to control the way in which the RTL Compiler shell operates. Changing the settings of these attributes is performed using the set_attribute command.

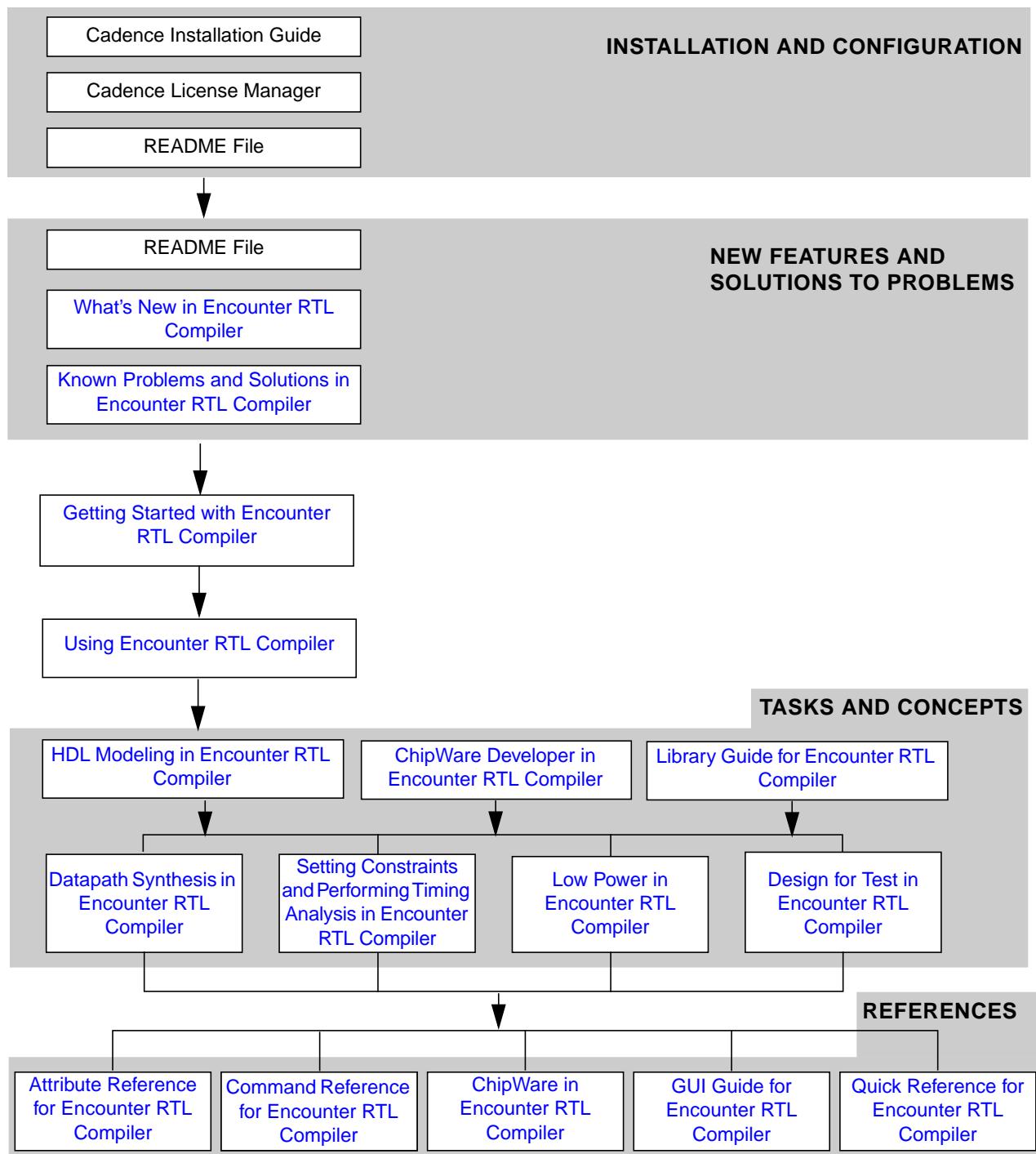
Additional References

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.

- 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 E-mail the message ID you would like improved to:

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 the man pages from the UNIX shell:

1. Set your environment to view the correct directory:

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

2. Enter the name of the command or attribute that you want. 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

The list below describes the syntax conventions used for the RTL Compiler attributes.

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

Introduction

This document describes the syntax of the RTL Compiler attributes.

An *attribute* is a setting that controls how RTL Compiler operates on objects; for example during synthesis and technology mapping.

An *object* is anything RTL Compiler can manipulate, such as libraries, designs, subdesigns, instances, ports, constraints, scan chains, and so on.

Design data is originally stored in the design hierarchy on the corresponding objects when reading in the libraries, the HDL files, and the constraints. During the synthesis session, the design information hierarchy (including the objects and attributes) is continuously updated.

- To retrieve the value of an attribute on an object, use the following command:

```
get_attribute attribute_name object
```

- To change the setting of an attribute on an object, use the following command:

```
set_attribute attribute_name attribute_value objects
```

In this book, the attributes are organized according to functional categories:

- Input and Output attributes affect how the HDL files are read in or written out
- Design For Test attributes affect scan chain insertion
- Low Power Synthesis attributes control clock-gating insertion, leakage and dynamic power optimization and so on

In each functional category, attributes are listed with the object types they can be set on.

Some attributes can apply to several objects. You can refer to

- List of Attributes and Associated Object Types for an alphabetical list of all attributes with the objects they apply to

Attribute Reference for Encounter RTL Compiler

Introduction

General

Root Attributes

- [attribute_path](#) on page 101
- [continue_on_error](#) on page 101
- [cpu_runtime](#) on page 102
- [dont_report_library](#) on page 102
- [elapsed_runtime](#) on page 102
- [fail_on_error_mesg](#) on page 102
- [find_fuzzy_match](#) on page 103
- [find_inefficient_threshold](#) on page 104
- [find_inefficient_use](#) on page 104
- [find_help_is_style](#) on page 104
- [information_level](#) on page 105
- [limited_access_feature](#) on page 106
- [log_command_error](#) on page 106
- [max_cpus_per_server](#) on page 107
- [max_super_thread_cache_size](#) on page 107
- [memory_usage](#) on page 108
- [peak_memory](#) on page 108
- [platform_wordsize](#) on page 108
- [print_error_info](#) on page 108
- [program_major_version](#) on page 109

Attribute Reference for Encounter RTL Compiler

General—List

- [program_name](#) on page 110
- [program_short_name](#) on page 110
- [program_version](#) on page 110
- [real_runtime](#) on page 110
- [report_library_message_summary](#) on page 111
- [report_tcl_command_error](#) on page 111
- [restore_history_file](#) on page 112
- [save_history_file](#) on page 113
- [show_command_usage](#) on page 114
- [show_report_options](#) on page 115
- [source_verbose](#) on page 115
- [source_verbose_info](#) on page 116
- [source_suspend_on_error](#) on page 116
- [source_verbose_proc](#) on page 117
- [startup_license](#) on page 117
- [statistics_db_file](#) on page 118
- [statistics_db_runtime](#) on page 118
- [statistics_enable_power_report](#) on page 119
- [statistics_log_data](#) on page 119
- [statistics_run_description](#) on page 120
- [statistics_run_id](#) on page 121
- [stdout_log](#) on page 121
- [super_thread_batch_command](#) on page 122
- [super_thread_cache](#) on page 123
- [super_thread_debug_directory](#) on page 123
- [super_thread_equivalent_licenses](#) on page 124
- [super_thread_kill_command](#) on page 125

Attribute Reference for Encounter RTL Compiler

General—List

- [super_thread_peak_memory](#) on page 126
- [super_thread_rsh_command](#) on page 126
- [super_thread_servers](#) on page 127
- [super_thread_status_command](#) on page 128
- [tcl_result_truncation_length](#) on page 128

Design Attributes

- [state](#) on page 129

Message Attributes

- [count](#) on page 131
- [help](#) on page 131
- [help_always_visible](#) on page 131
- [id](#) on page 131
- [max_print](#) on page 132
- [print_count](#) on page 132
- [priority](#) on page 132
- [screen_max_print](#) on page 133
- [screen_print_count](#) on page 133
- [severity](#) on page 133
- [truncate](#) on page 134
- [type](#) on page 134

Attribute Attributes

- [additional_help](#) on page 135
- [category](#) on page 135
- [check_function](#) on page 136
- [compute_function](#) on page 137

Attribute Reference for Encounter RTL Compiler

General—List

- [computed](#) on page 137
- [data_type](#) on page 138
- [default_value](#) on page 138
- [help](#) on page 139
- [set_function](#) on page 139
- [settable](#) on page 140
- [units](#) on page 140

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

attribute_path

```
attribute_path {basename | pathname | vname}
```

Default: basename

Read-write `root` attribute. Specifies the format to use for the object names in the command output of the `set_attribute` and `reset_attribute` commands. The format can help you find an object when multiple objects in the design have the same (base) name. The attribute can have the following values:

basename	Specifies to return the object name.
pathname	Specifies to return the full vdir path to the object.
vname	Specifies to return the Verilog style names where appropriate, otherwise defaults to the basename.

Related Information

Related commands: [reset_attribute](#)
[set_attribute](#)

continue_on_error

```
continue_on_error {false | true}
```

Default: false

Read-write `root` attribute. Controls whether to continue processing the scripts even if an error occurs. Set this attribute to `true` to continue processing when an error occurred.

cpu_runtime

```
cpu_runtime float
```

Read-only root attribute. Returns the total runtime (in seconds) which is computed as the sum of the cpu time of the main/foreground process and the cpu time across all background threads for a super-threaded session. In a single-threaded run, this will be the runtime consumed by the main process.

Note: Since this attribute keeps track of total CPU seconds, it is very dependent on the processor and clock-speed. Faster processors will generally have lower run-times, all else being equal. Run-time is affected by machine-loading to the order of 10-15% because increased loading leads to more context-switching, which in turn contributes to more stalls and general inefficiency in the program's execution. Run-time is also affected more significantly by memory. Processes that start to page affect the CPU run-time.

dont_report_library

```
dont_report_library {false | true}
```

Default: false

Read-write root attribute. If set to true, the library information in the report headers is suppressed.

elapsed_runtime

```
elapsed_runtime integer
```

Read-only root attribute. Returns the elapsed wall clock time (in seconds) for the current RTL Compiler session.

fail_on_error_mesg

```
fail_on_error_mesg {false | true}
```

Default: false

Read-write root attribute. If set to true, RTL Compiler commands will fail (stop) whenever they produce an ERROR message.

Note: This behavior applies automatically to all commands implemented in C++. However, for a command implemented in Tcl, the attribute has no effect unless the command checks the value of this attribute to drive its behavior.

find_fuzzy_match

```
find_fuzzy_match {false | true}
```

Default: false

Read-write root attribute. Enables matching of component names that may have been ungrouped, with the names in a netlist before ungrouping.

The search is influenced by the settings of the `hdl_generate_index_style` and `hdl_generate_separator` attributes which affect the vector naming style.

Note: Use this attribute on an as needed basis, because this setting completely changes the normal find behavior.

Example

The following attribute settings will allow the tool to match the component name `a/b/c_d_e` in a DEF file with the component name `a/b/c/d/e` in the netlist.

```
set_attribute hdl_generate_index_style %s_%d_ /  
# Format of 'for generate' block labels, default %s[%d]  
set_attribute hdl_generate_separator _ /  
#String to separate generate block labels, default '.'  
set_attribute find_fuzzy_match true /
```

Related Information

Affected by these attributes:

[hdl_generate_index_style](#) on page 739

[hdl_generate_separator](#) on page 742

find_inefficient_threshold

`find_inefficient_threshold integer`

Default: 100000

Read-write root attribute. Specifies the number of vdirs that the `find` command can search before the tool should issue the TUI-77 message.

For example, searching from root to find a message is not always inefficient (`find / -message messages/TUI/TUI-13`), but searching from root for one pin in a huge design is inefficient.

Related Information

Affects this command: [find](#)

Related attribute: [find_inefficient_use](#) on page 104

find_inefficient_use

`find_inefficient_use {false | true}`

Default: false

Read-write root attribute. Controls whether a warning is issued if you use a potentially inefficient `find` syntax. By default, no messages are printed. If set to true, inefficient use of the `find` syntax will be flagged by the TUI-77 message.

Related Information

Affects this command: [find](#)

Related attribute: [find_inefficient_threshold](#) on page 104

find_help_ls_style

`find_help_ls_style {true | false}`

Default: true

Read-write root attribute. Prints the help for the `find` command in the same style as the UNIX `ls` command. In default 'ls' style, the entries are sorted down the columns.

Attribute Reference for Encounter RTL Compiler

General—Root Attributes

Example

With the `find_help_ls_style` attribute set to `true` (default), the help for the `find` command looks like this:

```
rc:/> find -h
  find: finds an object by type and name
Usage: ...
      actual_scan_chain          memory_spare_column_map_data
      actual_scan_segment         memory_spare_row
      analysis_view               memory_spare_row_map_address
      attribute                  message
      blackbox                   message_group
      ...
      ...
```

With the `find_help_ls_style` attribute set to `false`, the help looks like this:

```
rc:/> find -h
  find: finds an object by type and name
Usage: ...
      actual_scan_chain          actual_scan_segment
      analysis_view               attribute
      blackbox                   blockage
      boundary_scan_segment      bump
      ...
      ...
```

Related Information

Affects this command: [find](#)

information_level

`information_level integer`

Default: 1

Read-write `root` attribute. Controls the amount of information RTL Compiler produces when executing commands. Specify a number of 0 through 9. The higher the value, the more verbose the output. The extra verbosity can be useful when debugging problem designs.

Example

If you set the value of the `information_level` attribute to 3 before you run the `check_dft_rules` command, the output of the command will also list the valid scan cells it found in the technology library.

Related Information

Affects these commands: all commands

limited_access_feature

```
limited_acces_feature {{feature key}...}
```

Read-write root attribute. Specifies a list of sub-lists to enable limited access features. Each sublist contains a feature name and the corresponding key to access the feature.

Note: You need to contact Cadence to access a limited-access feature and get the required key.

Example

```
set_attribute limited_access_feature {{ieee_1801 nnnn}} /
```

limit_lbr_messages

```
limit_lbr_messages {true | false}
```

Default: true

Read-write root attribute. Controls the printing of the LBR messages. By default, each LBR message will be printed maximum twenty times for each library that is being read. If for any LBR message, the `max_print` attribute has a lower limit, this limit will take precedence. If you set this attribute to `false`, all LBR messages will be printed to the logfile.

Related Information

Related attribute: [max_print](#) on page 132

log_command_error

```
log_command_error {false | true}
```

Default: false

Read-write root attribute. Controls whether a failing command is reported in the logfile.

Related Information

Affects these commands: all commands

max_cpus_per_server

`max_cpus_per_server integer`

Read-write root attribute. Controls the maximum number of *active* CPUs RTL Compiler is allowed to use per server for super-threading and multi-threading.

Examples

- The following command enables 3 CPUs on the local host:

```
set_attribute max_cpus_per_server 3 /
```

- The following commands enable 5 CPUs on various servers.

```
set_attribute super_thread_servers {localhost linux33} /  
set_attribute max_cpus_per_server 5 /
```

Related Information

Reducing Runtime Using Super-threading in *Using Encounter RTL Compiler*

Affects this command: [synthesize](#)

Related attributes: [auto_super_thread](#) on page 696

[super_thread_servers](#) on page 127

max_super_thread_cache_size

`max_super_thread_cache_size integer`

Default: 1000 (MegaBytes)

Read-write root attribute. Specifies the maximum approximate size allowed for the super-thread cache directory.

Note: A value of zero indicates that there is no limit. This effectively disables purging any data from the cache.

Related Information

Reducing Runtime Using Super-Caching in *Using Encounter RTL Compiler*

Affects this command: [synthesize](#)

Related attributes: [super_thread_cache](#) on page 123

memory_usage

`memory_usage integer`

Read-only root attribute. Returns the current memory consumption (in megabytes) by RTL Compiler for this process. This can be useful in Tcl scripts.

Example

```
rc:/> get_attribute memory_usage /  
25.26
```

peak_memory

`peak_memory float`

Read-only root attribute. Returns the peak memory consumption (in Mbytes) for this process (either the main process in a non-superthreaded run, or the foreground process in a super-threaded run).

platform_wordsize

`platform_wordsize integer`

Read-only root attribute. Returns the word size of the program in bits. Value is 32 or 64.

print_error_info

`print_error_info {false | true}`

Default: false

Read-write root attribute. Controls whether the tool prints the `errorInfo` variable when a command fails. If this attribute is set to true, the tool prints out a stack of Tcl `errorInfo` for the current script.

Attribute Reference for Encounter RTL Compiler

General—Root Attributes

Example

```
rc:/> set_attr print_error_info true
      Setting attribute of root '/': 'print_error_info' = true
rc:/> proc report::timing::listify {} {} ;# overwrite existing proc so 'report
      timing' will fail
rc:/> report timing
      Warning : Possible timing problems have been detected in this design. [TIM-11]
      : The design is 'test2'.
=====
      Generated by:          Encounter(R) RTL Compiler 11.10
      Generated on:         Sep 19 2011  02:30:11 pm
      ...
=====
errorInfo: wrong # args: should be "listify"
      while executing
"listify $the_pin"
errorInfo: wrong # args: should be "listify"
      (procedure "report_pin_name" line 19)
      invoked from within
"report_pin_name $rsl $full_pin_names"
errorInfo: wrong # args: should be "listify"
      ("pin" arm line 3)
      invoked from within
"switch $field_name {
      "pin" {
          if {[llength $rsl]} {
              set rsl [report_pin_name $rsl $full_pin_names]
          }
      }
      "type" {
          if {[lile..."]
errorInfo: wrong # args: should be "listify"
      (procedure "field_value" line 15)
      invoked from within
"field_value fields $desired_column $full_pin_names domain2id $map_timing"
errorInfo: wrong # args: should be "listify"
      (procedure "format_line" line 32)
      invoked from within
"format_line $fil $line $desired_columns $cols_0 col_widths $full_pin_names
show_first_pin_hier $debug_proc shown_flags domain2id $map_timing"
errorInfo: ...
...
...
```

program_major_version

`program_major_version string`

Default: 14.2

Read-write root attribute. Specifies the major version of the program.

program_name

`program_name string`

Default: Encounter(r) RTL Compiler

Read-only root attribute. Returns the name of the program being run. This may be useful for generating headers in customized reports.

program_short_name

`program_short_name string`

Default: rc

Read-only root attribute. Returns the short name of the program being run. This name is used for the prompt, the command file, and so on.

program_version

`program_version string`

Read-only root attribute. Returns the version of the program being run. This may be useful for generating headers in customized reports.

Example

```
rc:/> get_att program_version /  
6.1
```

real_runtime

`real_runtime float`

Read-only root attribute. Returns the runtime (in seconds) which is computed as the sum of the cpu time of the foreground process and the cpu time spent in the longest thread amongst the super-thread servers.

Note: Since this attribute keeps track of total CPU seconds, it is very dependent on the processor and clock-speed. Faster processors will generally have lower run-times, all else being equal. Run-time is affected by machine-loading to the order of 10-15% because increased loading leads to more context-switching, which in turn contributes to more stalls and general inefficiency in the program's execution. Run-time is also affected more significantly by memory. Processes that start to page affect the CPU run-time.

report_library_message_summary

```
report_library_message_summary {true | false}
```

Default: true

Read-write root attribute. Enables printing of a summary of the messages issued during library parsing. Messages issued during processing of the library are not included.

Example

```
Loading library /.../lib/mylib.lib
Info   : An unsupported construct was detected in this library. [LBR-40]
        : /.../lib/mylib.lib:102:16: Construct 'output_voltage' is not supported.
        : Check to see if this construct is really needed for synthesis. Many
liberty constructs are not actually required.
Info   : An unsupported construct was detected in this library. [LBR-40]
        : /.../lib/mylib.lib:108:15: Construct 'input_voltage' is not supported.
Info   : Created nominal operating condition. [LBR-412]
        : Operating condition '_nominal_' was created for the PVT values (1.0000
00,
2.250000, 125.000000) in library '/.../lib/mylib.lib'.
        :The nominal operating condition represents either the nominal PVT values
if specified in the library source, or the default PVT values (1.0, 1.0, 1.0).

Message Summary for Library /.../lib/mylib.lib:
*****
An unsupported construct was detected in this library. [LBR-40]: 2
Created nominal operating condition. [LBR-412]: 1
*****
```

report_tcl_command_error

```
report_tcl_command_error {false | true}
```

Default: false

Read-write root attribute. Enables printing of RC messages for Tcl command errors, such as wrong number of arguments, or bad set command. This allows to catch these errors with the report messages. By default, Tcl command errors are not registered.

Attribute Reference for Encounter RTL Compiler

General—Root Attributes

Example

```
rc:/> set_attr report_tcl_command_error true /  
rc:/> set foo  
Error   : Tcl 'set' command has encountered an error. [TUI-7]  
       : can't read "foo": no such variable while executing "set foo"  
       : Check the syntax and rerun.  
can't read "foo": no such variable  
rc:/> set foo 1 2  
Error   : Tcl command has wrong number of arguments. [TUI-9]  
       : wrong # args: should be "set varName ?newValue?"  
       : Check the syntax and rerun.  
Error   : Tcl 'set' command has encountered an error. [TUI-7]  
       : wrong # args: should be "set varName ?newValue?" while executing "set foo  
1 2"  
wrong # args: should be "set varName ?newValue?"  
rc:/>
```

restore_history_file

```
restore_history_file {false | true}
```

Default: false

Read-write root attribute. Controls the restoration of a history file. When you enable this attribute, the true history is restored from the `~/.rc_history` file.

Example

Assume the `.rc_history` file contains:

```
set_attr library typical.lib  
load test.v  
elaborate  
q
```

Now restart RC and enable the `restore_history_file` attribute:

```
rc:/> set_attr restore_history_file true /  
      Setting attribute of root '/': 'restore_history_file' = true  
rc:/> history  
1  set_attr restore_history_file true /  
2  set_attr library typical.lib  
3  load test.v  
4  elaborate  
5  q  
6  history  
rc:/> !2  
set_attr library typical.lib /  
      Setting attribute of root '/': 'library' = typical.lib
```

Related Information

Related attribute: [save_history_file](#) on page 113

save_history_file

`save_history_file {false | true}`

Default: false

Read-write [root](#) attribute. Controls the creation of a history file. When you enable this attribute, the command history is saved to the `~/.rc_history` file.

Example

Assume the following commands were entered on the `rc:/>` prompt:

```
set_attr save_history_file true /  
set_attr library typical.lib  
load test.v  
elaborate  
q
```

In this case, the `.rc_history` file contains the following commands:

```
set_attr library typical.lib  
load test.v  
elaborate  
q
```

Related Information

Related attribute: [restore_history_file](#) on page 112

show_command_usage

```
show_command_usage {true | false}
```

Default: true

Read-write root attribute. Controls whether the command usage is printed when the command fails. Set this attribute to `false` if you do not want the command usage printed when a command fails.

Examples

- The following example shows the behavior when this attribute is set to `false`.

```
rc:/designs/top_3> set_attribute show_command_usage false /  
      Setting attribute of root '/': 'show_command_usage' = false  
rc:/designs/top_3> filter  
Error   : A required argument was not specified. [TUI-202] [parse_options]  
        : An argument of type '<string>' was not specified.  
Failed on filter
```

- When using the default attribute setting (`true`), the command usage is printed when the command fails.

```
rc:/designs/top_3> filter  
Error   : A required argument was not specified. [TUI-202] [parse_options]  
        : An argument of type '<string>' was not specified.  
        : Rerun the command specifying all required arguments.  
filter: filters objects based on attribute value  
  
Usage: filter [-special] [-invert] [-regexp] <string> <string> [<object>+]  
  [-special]:  
    attribute values contain special characters  
  [-invert]:  
    filters out objects that match the expression and returns those that do not  
  [-regexp]:  
    uses regular expression matching for the required value  
<string>:  
  the attribute name  
<string>:  
  the required value. A compound string (containing spaces) should be  
represented as a list (using double-quotes or braces).  
  [<object>+]:  
    the object(s) to filter  
Failed on filter
```

show_report_options

```
show_report_options {false | true}
```

Default: false

Read-write root attribute. Controls whether the report command options used to generate a report are printed in the report header. By default, the command options are not printed.

Example

The following report shows the command options:

```
rc:/> set_attribute show_report_options true /  
rc:/> report timing -lint  
=====  
Generated by:          Encounter(R) RTL Compiler 10.1.300  
Generated on:         Apr 06 2011 11:01:05 am  
Generated with:      report timing -lint  
Module:               mult_bit_muxed_add  
Technology library:   umc113u210t2_wc 1.0  
Operating conditions: _nominal_ (balanced_tree)  
Wireload mode:        enclosed  
Area mode:            timing library  
=====
```

source_verbose

```
source_verbose {false | true}
```

Default: false

Read-write root attribute. If set to true, each command in a file is printed as it is executed along with the file name from which it came, line number, current memory usage, and CPU time. All information is printed to standard output. Since each command is evaluated when it executes, setting this attribute to true can have some impact on the runtime and logfile sizes due to the amount of extra data being dumped.

Example

The following example shows the information printed when the `set_attr library` command of the `test3.g` command file is executed.

```
#At ./test3.g:3 4.21secs 25.26MB Fri Feb 20 04:45:28 PM PST 2009  
set_attr library /home/rchap/lib/tech/tsmc_25.lib
```

source_verbose_info

```
source_verbose_info {true | false}
```

Default: true

Read-write root attribute. Controls the information printed to standard output when the source_verbose attribute is set to true. By default, the tool writes out the command along with the file name from which it came, line number, current memory usage, and CPU time. If you set this attribute to false, only the command name will be printed.

Examples

Assuming that source_verbose is set to true, the next examples show the information printed when the set_attr library command of the test3.g command file is executed.

- Output printed when the source_verbose_info attribute is set to true:

```
#At ./test3.g:3 4.21secs 25.26MB Fri Feb 20 04:45:28 PM PST 2009  
set_attr library /home/rcap/lib/tech/tsmc_25.lib
```

- Output printed when the source_verbose_info attribute is set to false:

```
set_attr library /home/rcap/lib/tech/tsmc_25.lib
```

Related Information

Related attribute: [source_verbose](#) on page 115

source_suspend_on_error

```
source_suspend_on_error {false | true}
```

Default: false

Read-write root attribute. If set to true, the tool enters the suspend mode if an error occurs while sourcing the script. You can then interactively correct the error in the script and resume to run the script.

source_verbose_proc

```
source_verbose_proc {false | true}
```

Default: false

Read-write root attribute. Enables printing of the proc bodies in the script files to the command file. This attribute is only taken into account if the `source_verbose` root attribute was set to `true`. Because some of the procs can be very large, enabling this attribute can result in a huge command file.

Related Information

Affected by this attribute: [source_verbose](#) on page 115

startup_license

```
startup_license string
```

Read-only root attribute. Returns the name of the primary license that was used to start the tool.

Example

```
rc:/> get_attribute startup_license /  
RTL_Compiler_Ultra
```

Note: This information is also printed in the beginning of the log file:

```
Checking out license 'RTL_Compiler_Ultra'... (0 seconds elapsed)
```

statistics_db_file

`statistics_db_file file`

Default: `current_directory/run_id.stats_db`

Read-write `root` attribute. Specifies the name of the database file being written out to save the metrics data.

Example

```
set_attribute statistics_db_file ./output/sample.stats_db /
```

Related Information

[Tracking and Saving QoR Metrics in Using Encounter RTL Compiler](#)

Affects this command: [statistics write](#)

Related attributes: [statistics_db_runtime](#) on page 118

[statistics_log_data](#) on page 119

[statistics_enable_power_report](#) on page 119

[statistics_run_description](#) on page 120

[statistics_run_id](#) on page 121

statistics_db_runtime

`statistics_db_runtime float`

Read-only `root` attribute. Returns the elapsed runtime used to compute the statistics and write out the database file.

Related Information

[Tracking and Saving QoR Metrics in Using Encounter RTL Compiler](#)

Affected by this command: [statistics](#)

Related attributes: [statistics_db_file](#) on page 118

[statistics_enable_power_report](#) on page 119

[statistics_log_data](#) on page 119

[statistics_run_description](#) on page 120

[statistics_run_id](#) on page 121

statistics_enable_power_report

```
statistics_enable_power_report {false | true}
```

Default: false

Read-write root attribute. Controls whether to include the power metrics in the statistics information for the design.

Related Information

[Tracking and Saving QoR Metrics in Using Encounter RTL Compiler](#)

Affects this command: [statistics](#)

Related attributes: [statistics_db_file](#) on page 118

[statistics_db_runtime](#) on page 118

[statistics_log_data](#) on page 119

[statistics_run_description](#) on page 120

[statistics_run_id](#) on page 121

statistics_log_data

```
statistics_log_data {false | true}
```

Default: false

Read-write root attribute. Controls tracking and generation of the QoR metrics at each predefined (elaborate and synthesize) stage. By default, QoR metrics are not tracked or generated at the predefined stages.

Related Information

[Tracking and Saving QoR Metrics in Using Encounter RTL Compiler](#)

Affects this command: [statistics](#)

Related attributes: [statistics_db_file](#) on page 118

[statistics_db_runtime](#) on page 118

[statistics_enable_power_report](#) on page 119

[statistics_run_description](#) on page 120

[statistics_run_id](#) on page 121

statistics_run_description

`statistics_run_description string`

Default: " "

Read-write `root` attribute. Specifies the description for the run. This information is captured along with the QoR metrics which makes it easier to compare multiple runs.

Example

```
set_attribute statistics_run_description "Medium effort run with no path \
segmentation mode for max delay" /
```

Related Information

[Tracking and Saving QoR Metrics in Using Encounter RTL Compiler](#)

Affects these commands:

[statistics read](#)

[statistics run_stage_ids](#)

Related attributes:

[statistics_db_file](#) on page 118

[statistics_db_runtime](#) on page 118

[statistics_enable_power_report](#) on page 119

[statistics_log_data](#) on page 119

[statistics_run_id](#) on page 121

statistics_run_id

`statistics_run_id string`

Default: `design.date_time_stamp`

Read-write `root` attribute. Specifies the user-defined identification (ID) for the synthesis run.

Example

```
set_attribute statistics_run_id generic_med_map_high /
```

Related Information

[Tracking and Saving QoR Metrics in Using Encounter RTL Compiler](#)

Affects these commands:

[statistics report](#)

[statistics run_stage_ids](#)

Related attributes:

[statistics_db_file](#) on page 118

[statistics_db_runtime](#) on page 118

[statistics_enable_power_report](#) on page 119

[statistics_log_data](#) on page 119

[statistics_run_description](#) on page 120

stdout_log

`stdout_log log_file_name`

Default: `rc.log`

Read-write `root` attribute. Specifies the log file name for the current session. All information printed to standard out will be recorded in the specified log file. You can specify different log files multiple times during one session, thereby recording information that begins from a different part of the flow.

super_thread_batch_command

```
super_thread_batch_command {queue_command}
```

Default: bsub

Read-write root attribute. Specifies LSF or SGE commands to submit jobs to the queueing system for super-threading. Use this attribute in conjunction with the super_thread_kill_command attribute, which specifies LSF or SGE commands to remove jobs from the queueing system.

Examples

- The following example uses the SGE qsub command to submit the “RC_server” job to the “lnx-penny” queue and specifies that no output file should be created. The super_thread_kill_command attribute passes the qdel command to SGE to appropriately remove the job after it has finished.

```
rc:/> set_attribute super_thread_batch_command \
    {qsub -N RC_server -q lnx-penny -b y -j y -o /dev/null} /
rc:/> set_attribute super_thread_kill_command {qdel} /
```

- The following example uses the LSF bsub command to submit the “RC_server” job to the “lnx-penny” queue and specifies that no output file should be created. The super_thread_kill_command attribute passes the bkill command to LSF to appropriately remove the job after it has finished.

```
rc:/> set_attribute super_thread_batch_command \
    {bsub -q lnx-penny -o /dev/null -J RC_server} /
rc:/> set_attribute super_thread_kill_command {bkill} /
```



Caution

Do not use the bsub options -I, -Ip, or -Is. The super-threading behavior can become unpredictable with these LSF options.

Related Information

Reducing Runtime Using Super-threading in *Using Encounter RTL Compiler*

Affected by this command:

[rc -lsf_cpus -lsf_queue](#)

Related attributes:

[super_thread_debug_directory](#) on page 123

[super_thread_equivalent_licenses](#) on page 124

[super_thread_kill_command](#) on page 125

[super_thread_rsh_command](#) on page 126
[super_thread_servers](#) on page 127
[super_thread_status_command](#) on page 128

super_thread_cache

`super_thread_cache directory`

Read-write [root](#) attribute. Enables the caching feature in RTL Compiler and specifies the directory where the cache keys and corresponding mapped design data are cached to be used in a subsequent cache-enabled run. You must specify an existing directory to which you have read-write access.

Related Information

[Reducing Runtime Using Super-Caching](#) in *Using Encounter RTL Compiler*

Affects this command: [synthesize](#)

Related attribute: [max_super_thread_cache_size](#) on page 107

super_thread_debug_directory

`super_thread_debug_directory directory`

Read-write [root](#) attribute. Facilitates debugging of super-threading related issues and specifies the directory where super-thread data is to be saved for debugging purposes.

Related Information

[Reducing Runtime Using Super-threading](#) in *Using Encounter RTL Compiler*

Affects this command: [synthesize](#)

Related attributes: [super_thread_batch_command](#) on page 122
[super_thread_equivalent_licenses](#) on page 124
[super_thread_kill_command](#) on page 125
[super_thread_rsh_command](#) on page 126
[super_thread_servers](#) on page 127
[super_thread_status_command](#) on page 128

super_thread_equivalent_licenses

`super_thread_equivalent_licenses string`

Default: RTL_Compiler_CPU_Accel_Option RTL_Compiler_L
RTL_Compiler_Ultra RTL_Compiler_Physical Virtuoso_Digital_Implem
Virtuoso_Digital_Implem_XL C_to_Silicon_Compiler_L

Read-write root attribute. Specifies the order in which licenses can be checked out for super-thread servers.

Example

```
set_attr super_thread_equivalent_licenses "RTL_Compiler_L  
RTL_Compiler_CPU_Accel_Option RTL_Compiler_Ultra "
```

Related Information

Reducing Runtime Using Super-threading in *Using Encounter RTL Compiler*

Affects this command:

[synthesize](#)

Related attributes:

[super_thread_batch_command](#) on page 122

[super_thread_debug_directory](#) on page 123

[super_thread_kill_command](#) on page 125

[super_thread_rsh_command](#) on page 126

[super_thread_servers](#) on page 127

[super_thread_status_command](#) on page 128

super_thread_kill_command

```
super_thread_kill_command {queue_command}
```

Default: bkill -s 9

Read-write root attribute. Specifies LSF or SGE commands to remove jobs from the queueing system for super-threading. Use this attribute in conjunction with the `super_thread_batch_command` attribute, which specifies LSF or SGE commands to submit jobs to the queueing system.

Examples

- The following example uses the LSF `bsub` command to submit the “RC_server” job to the “lnx-penny” queue and specifies that no output file should be created. The `super_thread_kill_command` attribute passes the `bkill` command to LSF to appropriately remove the job after it has finished.

```
rc:/> set_attribute super_thread_batch_command \
    {bsub -q lnx-penny -o /dev/null -J RC_server} /
rc:/> set_attribute super_thread_kill_command {bkill} /
```

- The following example uses the SGE `qsub` command to submit the “RC_server” job to the “lnx-penny” queue and specifies that no output file should be created. The `super_thread_kill_command` attribute passes the `qdel` command to SGE to appropriately remove the job after it has finished.

```
rc:/> set_attribute super_thread_batch_command \
    {qsub -N RC_server -q lnx-penny -b y -j y -o /dev/null} /
rc:/> set_attribute super_thread_kill_command {qdel} /
```

Related Information

[Reducing Runtime Using Super-threading in Using Encounter RTL Compiler](#)

Related attributes:

[super thread batch command](#) on page 122
[super thread debug directory](#) on page 123
[super thread equivalent licenses](#) on page 124
[super thread rsh command](#) on page 126
[super thread servers](#) on page 127
[super thread status command](#) on page 128

super_thread_peak_memory

`super_thread_peak_memory real`

Read-only root attribute. Returns a conservative estimate of the total memory consumption (in Mbytes) by the current session. The value is the sum of the peak memory used by the foreground process and each of the super-threaded background processes. At any time, the required memory will always be less than the value returned by this attribute for both super-threading and non super-threading modes since the likelihood of the foreground process and background processes being at their peak memory consumption would be quite small.

Related Information

Related attribute: [peak_memory](#) on page 108

super_thread_rsh_command

`super_thread_rsh_command command`

Default: `rsh`

Read-write root attribute. Specifies the UNIX command to start a shell on another host.

For security reasons, some hosts do not allow you to use the `rsh` command (default) to connect to them, but they may allow you to use `ssh` or another command.

Related Information

[Reducing Runtime Using Super-threading](#) in *Using Encounter RTL Compiler*

Related attributes: [super_thread_batch_command](#) on page 122
[super_thread_debug_directory](#) on page 123
[super_thread_equivalent_licenses](#) on page 124
[super_thread_kill_command](#) on page 125
[super_thread_servers](#) on page 127
[super_thread_status_command](#) on page 128

super_thread_servers

```
super_thread_servers {machine_names [batch]}
```

Read-write root attribute. Specifies a list of machine names that should be used for super-threading. RTL Compiler supports super-thread servers of different platform types. Super-threading is the process of distributing work across multiple CPUs.

If you want to specify either the LSF or SGE queue managers, use the `batch` argument instead of any machine name.

Examples

- The following example illustrates RTL Compiler launching three processes on the current machine for super-threading.

```
rc:/> set_attribute super_thread_servers {localhost localhost localhost} /  
The runtime reduction using the super-threading feature is proportional to the number of  
CPUs provided for synthesis.
```

- The following example illustrates RTL Compiler launching two server processes on the machine called `linux33` and one process on `sun42` and one process on a queue manager.

```
rc:/> set_attribute super_thread_servers {linux33 linux33 sun42 batch} /
```

Related Information

[Reducing Runtime Using Super-threading](#) in *Using Encounter RTL Compiler*

Affected by this command: [rc -lsf_cpus](#)

Related attributes: [max_cpus_per_server](#) on page 107

[super_thread_batch_command](#) on page 122

[super_thread_debug_directory](#) on page 123

[super_thread_equivalent_licenses](#) on page 124

[super_thread_kill_command](#) on page 125

[super_thread_rsh_command](#) on page 126

[super_thread_status_command](#) on page 128

super_thread_status_command

```
super_thread_status_command {queue_command}
```

Read-write root attribute. Specifies the LSF or SGE command to check the status of the batch jobs in the queueing system for super-threading.

Example

- The following example uses the SGE qstat command to check the status of the jobs

```
set_attribute super_thread_status_command {qstat -j} /
```
- The following example uses the LSF bjobs command to check the status of the jobs in the queue.

```
set_attribute super_thread_status_command {bjobs -l} /
```

Related Information

[Super-threading in Using Encounter RTL Compiler](#)

Related attributes:

[super_thread_batch_command](#) on page 122
[super_thread_debug_directory](#) on page 123
[super_thread_equivalent_licenses](#) on page 124
[super_thread_kill_command](#) on page 125
[super_thread_rsh_command](#) on page 126
[super_thread_servers](#) on page 127

tcl_result_truncation_length

```
tcl_result_truncation_length integer
```

Default: 4096

Read-write root attribute. Limits the string length of the command results printed by the Tcl interpreter in RTL Compiler.

Design Attributes

Contain information about the specified design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design
```

or

```
set_attribute attribute_name attribute_value [find /des* -design name]
```

- To get a design attribute value, type

```
get_attribute attribute_name design
```

state

```
state {no_value |state }
```

Read-only `design` attribute. Returns the state of the specified design. The attribute can have the following values:

annotated	State of the design when physical information is annotated for some nets.
generic	State of the design after you perform <code>synthesize -to_generic</code> on an unmapped design.
mapped	State of the design when all cells are mapped is mapped.
partially_mapped	State of a design containing mapped and unmapped cells
unmapped	State of the design when it contains no mapped cells.

Note: When you read in a DEF file, the following values can be *appended* to the `annotated`, `mapped`, and `partially_mapped` state values:

_partially_placed	Indicates that the design is partially placed.
_placed	Indicates that the design is fully placed.
_unplaced	Indicates that the design is not placed.

Examples

- The following example shows that the design `test` is unmapped:

```
...
rc:/> read_hdl MOD69.v
rc:/> elaborate
rc:/> get_attribute state [find . -design test]
unmapped
```

- The following example shows that the design `top` is mapped:

```
...
synthesize -to_mapped
rc:/> get_attribute state [find . -design top]
mapped
```

Message Attributes

Contain information about a message in RTL Compiler. This includes the information that is printed, as well as the number of times the message has occurred or can be printed. Most message attributes are read only.

- To set a message attribute, type

```
set_attribute attribute_name attribute_value /messages/*/message_id
```

- To get a message attribute value, type

```
get_attribute attribute_name /messages/*/message_id
```

count

```
count integer
```

Read-only message attribute. Specifies the number of times the message has been issued. The value of this attribute is larger than or equal to the value of the print_count attribute.

help

```
help string
```

Read-only message attribute. Returns a more detailed explanation of the message or can include help to debug the problem.

help_always_visible

```
help_always_visible {false | true}
```

Default: false

Read-write message attribute. When enabled, the help for the message will be shown each time the message occurs. By default, the extended help is only shown the first time the message occurs.

id

```
id integer
```

Read-only message attribute. Returns the identification number of the message.

max_print

```
max_print {infinity | integer}
```

Default: infinity

Read-write message attribute. Specifies the maximum number of times a message can be printed to the logfile *and* the screen.

Related Information

Related attribute: [screen_max_print](#)

print_count

```
print_count integer
```

Read-only message attribute. Specifies the number of times the message has been printed to the logfile *and* the screen. The value of this attribute is smaller than or equal to the value of the max_print attribute, and is also determined by the value of the information_level attribute. The value of this attribute is smaller than or equal to the value of the count attribute.

Related Information

Related attribute: [screen_print_count](#)

priority

```
priority integer
```

Read-only message attribute. Returns the priority of the message. If the priority of a message is higher than the value of the information_level attribute, it will not be printed.

screen_max_print

```
screen_max_print {infinity | integer}
```

Default: infinity

Read-write message attribute. Specifies the maximum number of times this message can be printed to the screen. The max_print message attribute takes precedence if its value is smaller than the value of screen_max_print.

Related Information

Related attribute: [max_print](#)

screen_print_count

```
screen_print_count integer
```

Read-only message attribute. Specifies the number of times the message has been printed to the screen. The value of this attribute is smaller than or equal to the value of the screen_max_print attribute, and is also determined by the value of the information_level attribute. The value of this attribute is smaller than or equal to the value of the count attribute.

Related Information

Related attribute: [print_count](#)

severity

```
severity {Info | Error | Warning}
```

Read-write message attribute. Returns the severity of the message. You can upgrade the severity of a particular message. For example, you can change the severity of a message from Warning to Error, but you cannot change the severity from Error to Info.

Example

The following example upgrades the severity of the DFM-200 message from Warning to Error:

```
rc:/messages/DFM> get_attribute severity DFM-200
Warning

rc:/messages/DFM> set_attribute severity Error DFM-200
Setting attribute of message 'DFM-200': 'severity' = Error
```

truncate

```
truncate {true | false}
```

Default: true

Read-write message attribute. Limits messages in RTL Compiler to 4000 characters. All characters after the 4000 character limit are truncated. To remove this limit, set the attribute to `false`. However, this may dramatically increase the size of the log file.

type

```
type string
```

Read-only message attribute. Returns a brief explanation for the message.

Attribute Attributes

Contain information about the attributes. These attributes are read-only attributes, so you cannot set their values.

- To get an attribute attribute value, type

```
get_attribute attribute_name object_type/attribute
```

additional_help

additional_help string

Read-only attribute attribute. Returns the additional help for the specified attribute.

Note: Most attributes have no additional help.

Note: For user-defined attributes, the attribute value corresponds to the value set with the `-more_help_string` option of the `define_attribute` command.

Example

The following example shows the additional help for the `encounter_executable` root attribute.

```
rc:/messages> get_attribute additional_help root/encounter_executable
The default search order is the 1) ENOUNTER environment variable, 2) PATH
environment variable and 3) CDN_SYNTH_ROOT environment variable.
```

Related Information

Related command: [define_attribute](#)

category

category string

Read-only attribute attribute. Returns the category that the specified attribute belongs to. Categories group attributes that perform similar functions. For example, `elab` indicates that the attribute is used during elaboration. All categories starting with `lp_` indicate that these attributes are used for low power.

Note: For user-defined attributes, the attribute value corresponds to the value set with the `-category` option of the `define_attribute` command.

Example

The following example indicates that the `logic_abstract` design attribute belongs to the `elab` category.

```
rc:/designs> get_attribute category design/logic_abstract  
elab
```

Related Information

Related command: [define_attribute](#)

check_function

`check_function string`

Read-only attribute attribute. Returns the name of the Tcl proc that ensures that the defined attribute is valid.

Note: This attribute applies only to user-defined attributes and its value corresponds to the value set with the `-check_function` option of the `define_attribute` command.

Example

Assume you defined the `test_check` root attribute:

```
define_attribute test_check -obj_type root -data_type integer -category test \  
-help_string "test check function" -check_function check_fxn
```

The following example returns the name of the `check_function`.

```
rc:/> get_attr check_function root/test_check  
check_fxn
```

Related Information

Related command: [define_attribute](#)

compute_function

`compute_function string`

Read-only attribute attribute. Returns the name of the Tcl proc that computes the attribute value.

Note: This attribute applies only to user-defined attributes and its value corresponds to the value set with the `-compute_function` option of the `define_attribute` command.

Example

Assume you defined the `test_compute` root attribute:

```
define_attribute test_compute -obj_type root -data_type integer -category test \
    -help_string "test compute function" -compute_function compute_fxn
```

The following example returns the name of the `check_function`.

```
rc:/> get_attr compute_function root/test_compute
compute_fxn
```

Related Information

Related command: [define_attribute](#)

computed

`computed {false | true}`

Default: false

Read-only attribute attribute. Indicates whether the value of the specified attribute is computed.

Note: For user-defined attributes, the attribute value is true if the `-computed_function` option of the `define_attribute` command was set.

Example

The following example indicates that the value of the `area` design attribute is computed.

```
rc:/designs> get_attribute computed design/area
true
```

Related Information

Related command: [define_attribute](#)

data_type

`data_type string`

Read-only attribute attribute. Returns the data type of the value of the specified attribute. The data type can be boolean, fixed point, floating point number, integer, or string.

Note: For user-defined attributes, the value corresponds to the value set with the `-data_type` option of the `define_attribute` command.

Example

The following example indicates that the data type of the `area` design attribute is a float with double precision.

```
rc:/designs> get_attribute data_type design/area  
double
```

Related Information

Related command: [define_attribute](#)

default_value

`default_value string`

Read-only attribute attribute. Returns the default value of the specified attribute.

Note: For user-defined attributes, the value corresponds to the value set with the `-default_value` option of the `define_attribute` command.

Example

The following example indicates that the `area` design attribute has no default.

```
rc:/designs> get_attribute default_value design/area  
no_value
```

Related Information

Related command: [define_attribute](#)

help

```
help string
```

Read-only attribute attribute. Returns the help string you would see if you used the `get_attribute -h` command for the specified attribute.

Note: For user-defined attributes, the value corresponds to the value set with the `-help_string` option of the `define_attribute` command.

Example

The following command returns the help for the `async_clear_phase` libpin attribute.

```
rc:/designs> get_attribute help libpin/async_clear_phase  
Sequential input asynchronous clear phase of libpin.
```

Related Information

Related command: [define_attribute](#)

set_function

```
set_function string
```

Read-only attribute attribute. Returns the name of the Tcl proc that allows to override (set) a user-defined value.

Note: This attribute applies only to user-defined attributes and its value corresponds to the value set with the `-set_function` option of the `define_attribute` command.

Example

Assume you defined the `test_set` root attribute:

```
define_attribute test_set -obj_type root -data_type integer \  
-category test -help_string "test set function" -set_function set_fxn
```

The following example returns the name of the `set_function`.

```
rc:/> get_attr set_function root/test_set  
set_fxn
```

Related Information

Related command: [define_attribute](#)

settable

```
settable {false | true}
```

Default: false

Read-only attribute attribute. Indicates whether the value of the specified attribute can be set with the `set_attribute` command.

Example

The following example shows that you cannot set the value of the `async_clear_phase` `libpin` attribute.

```
rc:/designs> get_attribute settable libpin/async_clear_phase  
false
```

units

```
units string
```

Read-only attribute attribute. Returns the units of the value of the specified attribute.

Note: Most attributes have no units.

Example

The following example indicates that the height of floorplan is given in microns.

```
rc:/messages> get_attr units design/fplan_height  
microns
```

GUI

Root Attributes

- [gui_auto_update](#) on page 142
- [gui_enabled](#) on page 142
- [gui_hv_phys_threshold](#) on page 142
- [gui_hv_threshold](#) on page 143
- [gui_sv_threshold](#) on page 143
- [gui_sv_update](#) on page 143
- [gui_visible](#) on page 144

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

gui_auto_update

```
gui_auto_update {true | false}
```

Default: true

Read-write `root` attribute. Indicates whether the GUI should be automatically updated.

gui_enabled

```
gui_enabled {false | true}
```

Read-only `root` attribute. Indicates whether the tool was started in GUI mode.

Related Information

Set by this command: [rc](#)

gui_hv_phys_threshold

```
gui_hv_phys_threshold integer
```

Default: 10

Read-write `root` attribute. Specifies the number of instances that a hierarchical instance should have to be highlighted in the Physical Viewer when you select the *Highlight Physical* command. If the number of instances in a hierarchical instance is below the threshold, the hierarchical instance is not highlighted.

gui_hv_threshold

`gui_hv_threshold integer`

Default: 50

Read-write `root` attribute. Sets the threshold for the number of objects that is listed for each hierarchy level in the hierarchy viewer. If the number of objects for a given hierarchy level exceeds the threshold, the list is truncated.

gui_sv_threshold

`gui_sv_threshold integer`

Default: 2000

Read-write `root` attribute. Sets the threshold for the number of instances that can be displayed in the current hierarchy level. If the number of instances in the current hierarchy level exceeds the threshold, the display mode for the Schematic Viewer is set to manual mode. This implies that the value of the `gui_sv_update` attribute is set to manual, even if the attribute was set to auto.

gui_sv_update

`gui_sv_update {manual | auto}`

Default: manual

Read-write `root` attribute. Controls the display mode for the Schematic Viewer.

If the attribute is set to `auto`, you can display the schematic of an instance by double-clicking

- The left or middle mouse button on the instance in the Hierarchy Viewer
- The left mouse button on the instance in the Schematic Viewer

If the attribute is set to `manual`, you can display the schematic of an instance by selecting the instance in

- The Hierarchy Viewer and selecting the *Open in – Schematic Viewer (main)* command from the context menu
- The main Schematic Viewer and selecting the *Open in – Schematic Viewer (new)* command from the context menu

Related Information

Affected by this attribute: [gui_sv_threshold](#) on page 143

gui_visible

`gui_visible {false | true}`

Default: false

Read-only [root](#) attribute. Indicates whether the GUI is currently visible.

By default, the tool is started in GUI mode (unless you started with the `-nogui` option). However, the GUI is only visible by entering the [gui_raise](#) or [gui_show](#) command.

Before you use the [gui_raise](#) or [gui_show](#) commands, this attribute will return `false`. After you use either of these commands, the attribute returns `true`.

ChipWare

hdl_arch Attributes

- [location](#) on page 148

hdl_bind Attributes

- [avoid](#) on page 149
- [constraint](#) on page 149
- [operator](#) on page 149
- [param_association](#) on page 150
- [pin_association](#) on page 150
- [priority](#) on page 150
- [unbound_oper_pin](#) on page 151

hdl_comp Attributes

- [avoid](#) on page 152
- [designware_compatibility](#) on page 153
- [location](#) on page 153
- [obsolete](#) on page 153
- [parameters](#) on page 154
- [pins](#) on page 154
- [report_as_datapath](#) on page 154
- [sim_model](#) on page 155

hdl impl Attributes

- [avoid](#) on page 156
- [language](#) on page 156
- [legal](#) on page 156
- [location](#) on page 157
- [obsolete](#) on page 157
- [pre_elab_script](#) on page 158
- [preserve_techelts](#) on page 159
- [priority](#) on page 159
- [technology](#) on page 159

hdl inst Attributes

- [preferred_impl](#) on page 160

hdl label Attributes

- [preferred_comp](#) on page 162
- [preferred_impl](#) on page 163

hdl lib Attributes

- [avoid](#) on page 164

hdl oper Attributes

- [signed](#) on page 165

hdl param Attribute

- [current_value](#) on page 166
- [hdl_parameter](#) on page 166
- [formula](#) on page 167

hdl_pin Attributes

- [bit_width](#) on page 168
- [direction](#) on page 168
- [permutable_group](#) on page 168
- [signed](#) on page 168

Subdesign Attributes

- [candidate_impls](#) on page 170
- [selected_impl](#) on page 171
- [speed_grade](#) on page 171
- [sub_arch](#) on page 172
- [user_speed_grade](#) on page 172

hdl_arch Attributes

Contain information about user-defined modules (or entities).

- To set an hdl_arch attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_arch name]
```

- To get a an hdl_arch attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_arch name]
```

location

`location pathname`

Read-only hdl_arch attribute. Returns the name and physical location of the source file that contains the specified entity (in VHDL) or module (in Verilog).

Note: This attribute is supported only in the RTL flow.

Example

The following example returns the source Verilog file for module `top`:

```
rc:/> get_attribute location /hdl_libraries/default/architectures/top
```

Related Information

Related attributes: (hdl_comp) location on page 153

 (hdl_impl) location on page 157

hdl_bind Attributes

Contain information about binding within the ChipWare Developer framework.

- To set an hdl_bind attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_bind name]
```

- To get a an hdl_bind attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_bind name]
```

avoid

```
avoid {false | true}
```

Default: false

Read-write hdl_bind attribute. Determines whether the specified binding should be used during elaboration.

Related Information

Related attributes:

(hdl_comp) avoid on page 152
(hdl_impl) avoid on page 156
(hdl_lib) avoid on page 164
(libcell) avoid on page 190

constraint

```
constraint constraint_setting
```

Read-write hdl_bind attribute. Specifies the constraint setting, which is a set of conditions that must be satisfied to make the specified binding valid during elaboration.

operator

```
operator operator_name
```

Read-only hdl_bind attribute. Returns the name of the synthetic operator to which the specified binding applies.

param_association

`param_association string`

Read-write `hdl_bind` attribute. Specifies the method to compute values for parameters of the component. The parameter values can be obtained either from

- Input pins of the synthetic operator that are driven by constant values in the HDL subprogram
- Constant values

pin_association

`pin_association string`

Read-write `hdl_bind` attribute. Specifies how pins of the specified component are to be mapped. They can be mapped either through:

- Pins of the synthetic operator
- Constant values

priority

`priority integer`

Read-write `hdl_bind` attribute. Specifies an integer representing the priority of the binding among all the valid bindings of the specified synthetic operator. The highest value indicates the highest priority.

Related Information

Related attributes: [\(hdl_impl\) priority on page 159](#)

unbound_oper_pin

`unbound_oper_pin unbound_setting`

Read-write `hdl_bind` attribute. Specifies the unbound setting. The unbound setting is a set of constant values that can be given to the input pins of the synthetic operator. The input pins cannot already be mapped to signals in an HDL subprogram that specifies the synthetic operator through the `map_to_operator` pragma.

hdl_comp Attributes

Contain information about ChipWare components.

- To set an hdl_comp attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_comp name]
```

- To get a an hdl_comp attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_comp name]
```

avoid

```
avoid {false | true}
```

Default: false

Read-write hdl_comp attribute. Determines whether a particular ChipWare component from the specified library should be used during elaboration.

Example

The following example specifies that the CW_add component should not be used during elaboration:

```
rc:/> set_attribute avoid true /hdl_libraries/CW/components/CW_add
```

Related Information

Related attributes:	(hdl_bind) avoid on page 149
	(hdl_impl) avoid on page 156
	(hdl_lib) avoid on page 164
	(libcell) avoid on page 190

designware_compatibility

```
designware_compatibility {false | true}
```

Read-write `hd1_comp` attribute. Indicates whether the component is compatible with an existing DesignWare component. When `false`, the component has no corresponding DesignWare counterpart. When `true`, the component is compatible with an existing DesignWare component. When you use such a component the tool prints message CDFG-820 to the log file to point out that while the features and functions are compatible they cannot be guaranteed to be exactly implementation-equivalent. It is your responsibility to verify if the specific Cadence implementation matches your requirements.

location

location *pathname*

Read-write hdl_comp attribute. Specifies the physical location of the source file that contains the VHDL entity declaration for the specified component.

Note: This attribute is supported only in the RTL flow.

Related Information

Related attributes: (hdl_arch) [location](#) on page 148
(hdl_Impl) [location](#) on page 157

obsolete

`obsolete {false | true}`

Default: false

Read-write `hdl_comp` attribute. Indicates whether the specified ChipWare component will be obsolete. If the attribute returns a value of `true`, you should replace the component with a comparable one that will not be obsoleted.

Related Information

Related attribute: (hdl impl) obsolete on page 157

parameters

`parameters string`

Read-only `hdl_comp` attribute. Returns an ordered list of all parameters of the specified ChipWare component.

Note: Do not confuse this attribute with the parameters branch of vdir objects attached to the `hdl_comp` object. Under that branch, each parameter is represented by its own `hdl_param` object.

Related Information

Related attribute: [\(hdl_arch\) parameters](#) on page 330

pins

`pins pin_list`

Read-only `hdl_comp` attribute. Returns an ordered list of all pins of the specified ChipWare component.

Note: Do not confuse this attribute with the pins branch of vdir objects attached to this `hdl_comp` object. Under that branch, each pin is represented by its own `hdl_pin` object.

Related Information

Related attribute: [\(hdl_arch\) parameters](#) on page 330

report_as_datapath

`report_as_datapath {false | true}`

Default: `false`

Read-write `hdl_comp` attribute. When set to `true`, the ChipWare component represented by this `hdl_comp` attribute is considered a datapath component, and it will be included in the datapath report generated by the `report datapath` command.

If this attribute is set to `false`, then this ChipWare component will not be included in the datapath report.

Regarding ChipWare components, this attribute is set to `true` by default for datapath components and set to `false` by default for other components.

sim_model

```
sim_model {{hdl_format list_of_unix_paths}...}
```

Read-write hdl_comp attribute. Specifies the UNIX location of the simulation model for the specified ChipWare component. This attribute takes a Tcl list of Tcl lists: each sub-list represents a simulation model of the ChipWare component and a pair of strings in the following format:

```
{hdl_format list_of_unix_paths}
```

The possible values for *hdl_format* are:

- v1995 (for Verilog-1995 simulation models)
- v2001 (for Verilog-2001 simulation models)
- sv (for SystemVerilog simulation models)
- vhdl1987 (for VHDL-1987 simulation models)
- vhdl1993 (for VHDL-1993 simulation models)

The *list_of_paths* is a UNIX path pointing to the simulation model. If it is a relative path, the *hdl_search_path* attribute can help identify its actual location.

Examples

If the simulation model is not hierarchical, the *sim_model* attribute values can look like the following example:

```
{ { v1995 $path/CW_complete.v } \
{ v2001 $path/CW_complete.v } \
{ vhdl1987 $path/CW_complete.vhd } \
{ vhdl1993 $path/CW_complete.vhd } }
```

If the simulation model is hierarchical, the *sim_model* attribute values can look like the following example:

```
{ { v1995 { $path/CW_top.v $path/CW_leaf.v } } \
{ v2001 { $path/CW_top.v $path/CW_leaf.v } } \
{ vhdl1987 { $path/CW_top.vhd $path/CW_leaf.vhd } } \
{ vhdl1993 { $path/CW_top.vhd $path/CW_leaf.vhd } } }
```

Related Information

Affects this command: [elaborate](#)

hdl_impl Attributes

Contain information about implementation within the ChipWare Developer framework.

- To set an hdl_impl attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_impl name]
```

- To get a an hdl_impl attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_impl name]
```

avoid

```
avoid {false | true}
```

Default: false

Read-write hdl_impl attribute. Specifies whether a particular architecture of a specific ChipWare component should be used during elaboration.

Related Information

Related attributes:	(hdl_bind) avoid on page 149
	(hdl_comp) avoid on page 152
	(hdl_lib) avoid on page 164
	(libcell) avoid on page 190

language

```
language language_version
```

Read-write hdl_impl attribute. Returns the HDL language version in which the RTL code for the specified architecture was written.

legal

```
legal formula
```

Read-write hdl_impl attribute. Specifies a formula, in Tcl, to determine the legality of the specified implementation. The criteria is usually based on the bit-width of input/output signals.

location

location pathname

Read-write `hdl_impl` attribute. Specifies the physical location of the source file containing the RTL code of the specified component implementation. If the source file is Verilog, it specifies the location of the entire synthesis model. If the source file is in VHDL, it specifies the location of the VHDL architecture.

Note: This attribute is supported only in the RTL flow.

Related Information

Related attributes: (hdl_arch) [location](#) on page 148
(hdl_comp) [location](#) on page 153

obsolete

`obsolete {false | true}`

Read-write `hdl_impl` attribute. Indicates whether the implementation (architecture) of the specified ChipWare component will be obsoleted. If this attribute returns a value of `true`, you should replace the implementation with a comparable one that will not be obsoleted.

Related Information

Related attribute: [\(hdl_comp\) obsolete](#) on page 153

pre_elab_script

`pre_elab_script {UNIX_path}`

Read-write `hdl_impl` attribute. Specifies the UNIX path that contains the pre-elaboration script. Each CWD synthesis model can be accompanied by a "pre-elaboration script". When this synthesis model is to be used to implement something, this script is sourced after its HDL code is parsed, but before its HDL code is elaborated (hence the name).

A pre-elaboration script is exercised on an `hdl_arch` object.

Example

The following example specifies the preferred component and implementations of the HDL operators and CWD instantiation in a module:

```

assert [string equal [pwd] /hdl_libraries/default/architectures/my_mult.booth]
set_attribute preferred_impl cla instances/INST_W*
set_attribute preferred_impl rpl instances/INST_R*
set_attribute preferred_comp CW_add labels/LABEL_R*
set_attribute preferred_impl rpl labels/LABEL_R*
if [expr [get_attribute current_value parameters/A_width] > 1] {
    if [expr [get_attribute current_value parameters/B_width] > 1] {
        set_attribute preferred_comp CW_add labels/LABEL_A*
        set_attribute preferred_impl cla labels/LABEL_A*
    }
}

```

This example assumes the HDL code of the synthesis model of this implementation has:

- Instance names starting with INST_W
 - Instance names starting with INST_R
 - Label names starting with LABEL_R
 - Label names starting with LABEL_A

Related Information

Related attributes: [\(hdl_inst\) preferred_impl](#) on page 160
[\(hdl_label\) preferred_impl](#) on page 163
[preferred_comp](#) on page 162

preserve_techelts

```
preserve_techelts {delete_ok | size_delete_ok | false }
```

Default: delete_ok

Read-write hdl_impl attribute. Determines how to optimize the technology cells that are explicitly instantiated in the synthesis model of the specified ChipWare implementation.

delete_ok	Allows the technology cells to be deleted during optimization, but does not allow resizing, renaming, or remapping them.
false	Allows changes to all the technology cells in the synthesis model during optimization.
size_delete_ok	Allows resizing or deleting of the technology cells during optimization, but not renaming or remapping them.

priority

```
priority integer
```

Read-write hdl_impl attribute. Specifies an integer representing the priority of the implementation among all the valid implementations of the specified component. The highest value indicates the highest priority.

Related Information

Related attributes: [\(hdl_bind\) priority](#) on page 150

technology

```
technology library_name
```

Read-write hdl_impl attribute. Specifies the name of a technology library, if the specified architecture (implementation) is technology-specific. The value should be a null string if it is technology-neutral.

hdl_inst Attributes

Contain information about HDL instances.

- To set an hdl_inst attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_inst name]
```

- To get a an hdl_inst attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_inst name]
```

preferred_impl

```
preferred_impl implementation_name [hdl_inst_pathname]
```

Read-write `hdl_inst` attribute. Specifies a preferred implementation for the specified CWD component instance. When RTL Compiler is choosing an implementation for a particular CWD component, it applies a rigorous selection process to select the best implementation. This attribute allows you to explicitly communicate to RTL Compiler a preference for a particular implementation for a particular CWD component.

Note: This attribute needs to be set before the `elaborate` command is issued.

As the name of this attribute indicates, it is a preference. The specified implementation must pass all of the following criteria to be conclusively honored:

- Its `legal` attribute is evaluated to `true`
- Its `priority` attribute is greater than 0
- Its `avoid` attribute is false
- Its `technology` attribute is either empty or consistent with the current library setting.

If any of these criteria fail for the specified implementation the following actions occur:

- A warning message is issued to explain the failure
- The preference is ignored
- The default implementation selection mechanism is used

Example

In the following example, the CWD component, CW_my_comp, has two implementations, i1 and i2. The RTL code that instantiates this component with an instance named u1:

```
module my_top (...);  
  ...  
  CW_my_comp #(...) u1 (...);  
  ...  
endmodule
```

To force RTL Compiler to use the i2 implementation for this instance of CW_my_comp, use the following commands before issuing the elaborate command.

```
rc:/> set_attribute preferred_impl i2 \  
[find [find / -hdl_arch my_top] -hdl_inst u1]
```

Alternatively, you can cd into the directory:

```
rc:/hdl_libraries/default/architectures/my_top/instances> ls  
./      u1      u2  
rc:/hdl_libraries/default/architectures/my_top/instances> set_attribute \  
preferred_impl i2 u1
```

Related Information

Affects this attribute: [selected_impl](#) on page 171

Related attributes: [candidate_impls](#) on page 170

[pre_elab_script](#) on page 158

(hdl_label) [preferred_impl](#) on page 163

hdl_label Attributes

Contain information about HDL labels.

- To set an hdl_label attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_label name]
```

- To get a an hdl_label attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_label name]
```

preferred_comp

```
preferred_comp component_name
```

Read-write hdl_label attribute. Used in conjunction with the attribute preferred_impl on page 163. The preferred_comp attribute specifies a preferred component for the HDL operator annotated by a label in the RTL code. For example, the following operation in the RTL code is label L1:

```
p <= a + b; -- pragma label L1
```

During elaboration, if the specified component is available, RTL Compiler uses it to implement the HDL operator. Otherwise, the component will be ignored and RTL Compiler will choose the best component.

Related Information

Affects this attribute: [selected_impl](#) on page 171

Related attributes: [candidate_impls](#) on page 170

[pre_elab_script](#) on page 158

(hdl_label) [preferred_impl](#) on page 163

preferred_impl

`preferred_impl implementation_name`

Read-write `hdl_label` attribute. Used in conjunction with the attribute [preferred_comp](#) on page 162. The `preferred_impl` attribute specifies a preferred implementation for the HDL operator annotated by a label in the RTL code. For example, the following operation in the RTL code is label L1:

```
p <= a + b; -- pragma label L1
```

During elaboration, if the specified implementation is available, RTL Compiler uses it to implement the HDL operator. Otherwise, the implementation will be ignored and RTL Compiler will choose the best implementation.

Related Information

Affects this attribute: [selected_impl](#) on page 171

Related attributes: [candidate_impls](#) on page 170

[pre_elab_script](#) on page 158

[preferred_comp](#) on page 162

(hdl_inst) [preferred_impl](#) on page 160

hdl_lib Attributes

Contain information about the HDL libraries.

- To set an hdl_lib attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_lib name]
```

- To get a an hdl_lib attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_lib name]
```

avoid

```
avoid {false | true}
```

Default: false

Read-write hdl_lib attribute. Determines whether ChipWare components from the specified library should be used during elaboration.

Related Information

Related attributes:	(hdl_comp) avoid on page 152
	(hdl_impl) avoid on page 156
	(hdl_lib) avoid on page 164
	(libcell) avoid on page 190

hdl_oper Attributes

Contain information about synthetic operators within the ChipWare Developer framework.

- To get a an `hdl_oper` attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_oper name]
```

signed

`signed {true | false}`

Read-only `hdl_oper` attribute. Returns whether the specified synthetic operator is a signed operator.

Related Information

Related attributes: (hdl_pin) [signed](#) on page 168

hdl_param Attribute

Contain information about the HDL parameters used inside the ChipWare component.

- To set an hdl_param attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_param name]
```

- To get a an hdl_param attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_param name]
```

current_value

`current_value integer`

Read-write hdl_param attribute. Specifies the value currently assigned to this parameter.

You can use this attribute in pre-elaboration scripts that are attached to `hdl_impl` objects via the `pre_elab_script` attribute.

Related Information

Related command:

[hdl_create parameter](#)

hdl_parameter

`hdl_parameter {true | false}`

Read-only hdl_param attribute. Returns whether the specified parameter is visible within the ChipWare component. If the specified parameter was created with the `hdl_create parameter` command without its `-hdl_invisible` option, the default value of this attribute will be `false`. If the specified parameter was created with the `-hdl_invisible` option, this attribute value becomes `true`. This attribute is valid on all parameters (`hdl_param` objects), not just those created by the `hdl_create parameter` command.

Related Information

Affected by this command:

[hdl_create parameter](#)

formula

formula *string*

Read-write hdl_param attribute. Specifies a formula, in Tcl, to compute the value of a parameter.

hdl_pin Attributes

Contain information about the pins of the ChipWare component.

- To set an hdl_pin attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_pin name]
```

- To get a an hdl_pin attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_pin name]
```

bit_width

bit_width formula

Read-write hdl_pin attribute. Specifies a formula, in Tcl, to compute the bit-width of the specified pin. The values are usually based on the parameters of the component.

direction

direction {input | output | inout}

Read-only hdl_pin attribute. Returns the direction of the specified pin.

permutable_group

permutable_group group_name

Read-write hdl_pin attribute. Specifies the name of the permutation group to which this pin belongs. The value is the null string if no group is specified.

signed

signed {true | false}

Read-write hdl_pin attribute. Determines whether the input or output pin of a synthetic operator is a signed or unsigned pin. By default, all inputs and outputs of a synthetic operator have the same signedness as the operator itself. Usually, you do not need to set this attribute. However, if a particular pin has a different signedness than the operator, you must set this attribute for that pin.

Example

- In the following example the DECODE_OP operator is created as an unsigned operator. By default, its pins A and Z will also be unsigned:

```
hdl_create_operator DECODE_OP -unsigned  
cd /hdl_libraries/synthetic/operators/DECODE_OP  
hdl_create_pin A -input  
hdl_create_pin Z -output
```

- In the following example the DECREMENT_UNS_CI_OP operator is created as an unsigned operator. However, you want the output pin to be signed:

```
hdl_create_operator DECREMENT_UNS_CI_OP -unsigned  
cd /hdl_libraries/synthetic/operators/DECREMENT_UNS_CI_OP  
hdl_create_pin A -input  
hdl_create_pin CI -input  
hdl_create_pin Z -output  
set_attr -quiet signed true pins/Z
```

Related Information

Related attributes: [\(hdl_oper\) signed](#) on page 165

Subdesign Attributes

Contain information about the subdesigns in the specified design. Subdesigns correspond to Verilog modules or VHDL entities instantiated in the top-level Verilog module or top-level VHDL entity.

- To set a subdesign attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subdesign name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subdesign name]
```

Note: These attributes are located at `/designs/design/subdesigns`

candidate_impls

```
candidate_impls {{impl_directory_1 bind_name_1}
                {impl_directory_2 bind_name_2} ...}
```

Read-only subdesign attribute. Returns all the implementations available for a particular subdesign. The returned value is a Tcl list of a Tcl list, where `impl_directory` represents the implementation directory and `bind_name` represents the binding name. Each list item represents a possible implementation, which is a valid binding to a particular instance.

Related Information

Related attributes: [selected_impl on page 171](#)

[\(hdl_inst\) preferred_impl on page 160](#)

[\(hdl_label\) preferred_impl on page 163](#)

selected_impl

```
selected_impl {impl_directory bind_name}
```

Read-only subdesign attribute. Returns the implementation of the subdesign at any stage in the synthesis flow. The returned value is a Tcl list, where *impl_directory* (implementation directory) can be the name of the hdl_lib, hdl_comp, or hdl_impl object. *bind_name* (binding name) is the name of the hdl_bind object.

Related Information

Affected by these attributes ([hdl_inst](#)) preferred_impl on page 160

 ([hdl_label](#)) preferred_impl on page 163

Related attribute: [candidate_impls](#) on page 170

speed_grade

```
speed_grade {very_slow | slow | medium| fast | very_fast | timing_driven}
```

Read-only subdesign attribute. Specifies the speed of an internal RC component, such as a datapath component. A component can have several implementations with different speeds. RTL Compiler automatically determines which implementation to choose to meet timing and area requirements.

Related Information

Affects this command: [synthesize](#)

Affected by this attribute: [user_speed_grade](#) on page 172

sub_arch

`sub_arch string`

Read-only subdesign attribute. Returns the partial product encoding scheme that the tool used for a particular multiplier component.

Example

The following command returns the sub-architecture of a particular multiplier component:

```
rc:/> get_attribute sub_arch [find /designs* -subdesign name]
```

Related Information

Controlling Sub-Architecture Selection *Datapath Synthesis in Encounter RTL Compiler*.

Affects this command: [synthesize](#)

Affected by this attribute: [user_sub_arch](#) on page 911

user_speed_grade

`user_speed_grade {very_slow | slow | medium | fast | very_fast}`

Read-write subdesign attribute. Allows you to choose a fixed implementation of an internal RTL Compiler component, such as a datapath component. A component can have several implementations with different speeds. RTL Compiler automatically determines which implementation to choose to meet timing and area requirements, but this attribute allows you to choose a different implementation.

Note: Set this attribute only on datapath subdesigns inferred by RTL Compiler.

Related Information

Controlling Architecture Selection in *Datapath Synthesis in Encounter RTL Compiler*

Affects this command: [synthesize](#)

Affects this attribute: [speed_grade](#) on page 171

Library

Library Attributes

- [cap scale in fF](#) on page 180
- [default power rail](#) on page 180
- [default wireload](#) on page 180
- [files](#) on page 181
- [input threshold pct fall](#) on page 181
- [input threshold pct rise](#) on page 182
- [leakage power scale in nw](#) on page 182
- [liberty attributes](#) on page 182
- [output threshold pct fall](#) on page 183
- [output threshold pct rise](#) on page 183
- [power rails](#) on page 184
- [slew derate from library](#) on page 184
- [slew lower threshold pct fall](#) on page 185
- [slew lower threshold pct rise](#) on page 185
- [slew upper threshold pct fall](#) on page 185
- [slew upper threshold pct rise](#) on page 186
- [time scale in ps](#) on page 186
- [usable comb cells](#) on page 186
- [usable seq cells](#) on page 187
- [usable timing models](#) on page 187

- [version](#) on page 187

Libcell Attributes

- [adder](#) on page 188
- [area](#) on page 188
- [area_multiplier](#) on page 189
- [async_clear](#) on page 189
- [async_preset](#) on page 189
- [avoid](#) on page 190
- [buffer](#) on page 190
- [cell_bitwidth](#) on page 191
- [cell_delay_multiplier](#) on page 191
- [cell_internal_power](#) on page 192
- [cell_leakage_power](#) on page 192
- [cell_min_delay_multiplier](#) on page 193
- [clock](#) on page 193
- [clock_gating_integrated_cell](#) on page 194
- [combinational](#) on page 194
- [congestion_avoid](#) on page 194
- [constraint_multiplier](#) on page 195
- [data_pins](#) on page 195
- [direction](#) on page 195
- [flop](#) on page 196
- [ground_direction](#) on page 196
- [height](#) on page 197
- [instance_probability](#) on page 197
- [inverter](#) on page 197

Attribute Reference for Encounter RTL Compiler

Library—List

- [is_always_on](#) on page 198
- [is_clock_gating_cell](#) on page 198
- [is_isolation_cell](#) on page 198
- [is_level_shifter](#) on page 198
- [latch](#) on page 199
- [latch_enable](#) on page 199
- [liberty_attributes](#) on page 199
- [liberty_level_shifter_type](#) on page 200
- [master_slave_flop](#) on page 200
- [master_slave_lssd_flop](#) on page 200
- [max_ground_input_voltage](#) on page 200
- [max_ground_output_voltage](#) on page 201
- [max_input_voltage](#) on page 201
- [max_output_voltage](#) on page 202
- [min_ground_input_voltage](#) on page 202
- [min_ground_output_voltage](#) on page 203
- [min_input_voltage](#) on page 203
- [min_output_voltage](#) on page 204
- [non_seq_setup_arc](#) on page 204
- [pad](#) on page 204
- [power_gating_cell](#) on page 205
- [power_gating_cell_type](#) on page 205
- [preserve](#) on page 206
- [scan_enable](#) on page 206
- [scan_in](#) on page 207
- [sequential](#) on page 207
- [sync_clear](#) on page 207

Attribute Reference for Encounter RTL Compiler

Library—List

- [sync_enable](#) on page 207
- [sync_preset](#) on page 208
- [systematic_probability](#) on page 208
- [timing_model](#) on page 208
- [timing_model_reason](#) on page 209
- [tristate](#) on page 209
- [unusable_reason](#) on page 209
- [usable](#) on page 210
- [user_defined](#) on page 210
- [width](#) on page 211

Libpin Attributes

- [async_clear_phase](#) on page 212
- [async_preset_phase](#) on page 212
- [capacitance](#) on page 213
- [clock_gate_clock_pin](#) on page 213
- [clock_gate_enable_pin](#) on page 213
- [clock_gate_obs_pin](#) on page 213
- [clock_gate_out_pin](#) on page 214
- [clock_gate_reset_pin](#) on page 214
- [clock_gate_test_pin](#) on page 214
- [clock_phase](#) on page 214
- [driver_type](#) on page 215
- [fall_capacitance_range](#) on page 215
- [fanout_load](#) on page 216
- [function](#) on page 216
- [generated_clock](#) on page 216

Attribute Reference for Encounter RTL Compiler

Library—List

- [ground](#) on page 217
- [higher_drive](#) on page 217
- [incoming_timing_arcs](#) on page 217
- [input](#) on page 217
- [internal](#) on page 217
- [is_always_on](#) on page 218
- [is_iq_function](#) on page 218
- [is_iqn_function](#) on page 218
- [isolation_cell_enable_pin](#) on page 218
- [latch_enable_phase](#) on page 218
- [level_shifter_enable_pin](#) on page 219
- [liberty_attributes](#) on page 219
- [lower_drive](#) on page 219
- [max_capacitance](#) on page 220
- [max_fanout](#) on page 220
- [max_transition](#) on page 220
- [min_capacitance](#) on page 221
- [min_fanout](#) on page 221
- [min_transition](#) on page 221
- [non_seq_setup_arc](#) on page 222
- [outgoing_timing_arcs](#) on page 222
- [output](#) on page 222
- [pad](#) on page 222
- [polarity](#) on page 223
- [power](#) on page 223
- [power_gating_pin_class](#) on page 223
- [power_gating_pin_phase](#) on page 224

Attribute Reference for Encounter RTL Compiler

Library—List

- [q_pin_of_d_pin](#) on page 224
- [rise_capacitance_range](#) on page 225
- [scan_enable_phase](#) on page 225
- [scan_in_phase](#) on page 225
- [signal_level](#) on page 225
- [sync_clear_phase](#) on page 226
- [sync_enable_phase](#) on page 226
- [sync_preset_phase](#) on page 226
- [tristate](#) on page 226
- [use](#) on page 227
- [user_defined](#) on page 227
- [user_function](#) on page 227
- [x_offset](#) on page 228
- [y_offset](#) on page 228

Libarc Attributes

- [enabled](#) on page 229
- [from_pin](#) on page 229
- [liberty_attributes](#) on page 229
- [real_enabled](#) on page 230
- [type](#) on page 230

Seq Function Attributes

- [d_function](#) on page 231

Operating Conditions Attributes

- [liberty_attributes](#) on page 232
- [process](#) on page 232

Attribute Reference for Encounter RTL Compiler

Library—List

- [temperature](#) on page 233
- [tree_type](#) on page 233
- [voltage](#) on page 233

Wireload Attributes

- [fanout_cap](#) on page 234
- [liberty_attributes](#) on page 235

Library Attributes

Contain information about the specified technology library. Most of these attributes are read only.

- To set a library attribute, type

```
set_attribute attribute_name attribute_value /libraries/library
```

or

```
set_attribute attribute_name attribute_value [find /lib* -library library]
```

- To get a library attribute value, type

```
get_attribute attribute_name [find /lib* -library library]
```

cap_scale_in_fF

`cap_scale_in_fF float`

Read-only library attribute. Returns the scaling factor used to compute any capacitance value in the library. All capacitance values in RTL Compiler are expressed in femtofarads. Resolution is 1/10.

Related Information

Affects this attribute: [library](#) on page 274

default_power_rail

`default_power_rail string`

Read-only library attribute. Returns the attribute value of the Liberty `default_power_rail` attribute.

default_wireload

`default_wireload string`

Read-write library attribute. Specifies the default wire-load model for a library. This model can be specified in the library, but you can override it.

files

`files string`

Read-only library attribute. Returns the full pathname of the specified library.

Example

- The following example specifies the `tutorial.lbr` to be used and then uses the `files` attribute to return the full path name of `tutorial.lbr`.

```
rc:/>set_attribute library tutorial.lbr
rc:/>get_attribute files tutorial
/home/usr/lib/tutorial.lbr
```

- The following example appends the second library to the first and then uses the `files` attribute to return the full path name of both libraries.

```
rc:/>set_attribute library {{tutorial.lbr append_factors.lib}}
rc:/>get_attribute files tutorial
/home/usr/lib/tutorial.lbr /home/usr/lib/append_factors.lib
```

Related Information

Affected by this attribute [library](#) on page 274

input_threshold_pct_fall

`input_threshold_pct_fall float`

Read-only library attribute. Returns the default value of the threshold point on an input pin signal falling from 1 to 0. If this attribute is not specified in the library, the value defaults to 50.0.

Related Information

Affects these commands:

[report cell_delay_calculation](#)
[report slew_calculation](#)
[report timing](#)

input_threshold_pct_rise

`input_threshold_pct_rise float`

Read-only library attribute. Returns the default value of the threshold point on an input pin signal rising from 0 to 1. If this attribute is not specified in the library, the value defaults to 50.0.

Related Information

Affects these commands:

[report cell_delay_calculation](#)
[report slew_calculation](#)
[report timing](#)

leakage_power_scale_in_nW

`leakage_power_scale_in_nW float`

Read-only library attribute. Returns the scaling factor used to compute the cell leakage power in the library. This attribute is determined by the value of the `leakage_power_unit` attribute defined in the Liberty library.

If the `leakage_power_unit` in the Liberty library is 10 pW, then this value is 0.01.

Related Information

Affects these commands:

[report gates](#)
[report power](#)

Affects this attribute:

[cell_leakage_power](#) on page 192

liberty_attributes

`liberty_attributes string`

Read-only library attribute. Returns a list of Liberty attributes and values that were specified at the library level.

Related Information

Related attributes:	(libcell) liberty_attributes on page 199 (libpin) liberty_attributes on page 219 (libarc) liberty_attributes on page 229 (operating conditions) liberty_attributes on page 232 (wireload) liberty_attributes on page 235
---------------------	--

output_threshold_pct_fall

`output_threshold_pct_fall float`

Read-only [library](#) attribute. Returns the default value of the threshold point on an output pin signal falling from 1 to 0. If this attribute is not specified in the library, the value defaults to 50.0.

Related Information

Affects these commands:	report cell_delay_calculation report slew_calculation report timing
-------------------------	---

output_threshold_pct_rise

`output_threshold_pct_rise float`

Read-only [library](#) attribute. Returns the default value of the threshold point on an output pin signal rising from 0 to 1. If this attribute is not specified in the library, the value defaults to 50.0.

Related Information

Affects these commands:	report cell_delay_calculation report slew_calculation report timing
-------------------------	---

power_rails

`power_rails Tcl_list`

Read-only library attribute. Returns a Tcl list of Tcl lists. Each Tcl list contains the power supply name and the corresponding voltage. The number of Tcl lists corresponds to the number of `power_rail` attributes in the `power_supply` group in the .lib file.

Example

```
rc:/libraries/> get_att power_rails mylib  
{VDDH 1.08} {VDDL 0.7} {VSS 0}
```

Related Information

[Leakage Power Specification](#) in *Library Guide for Encounter RTL Compiler*

[Internal Power Specification](#) in *Library Guide for Encounter RTL Compiler*

Affects these commands: [report gates](#)

[report power](#)

Affects this attribute: [cell_leakage_power](#) on page 192

slew_derate_from_library

`slew_derate_from_library float`

Read-only library attribute. Returns how the transition times need to be derated to match the transition times between the characterization trip points. If this attribute is not specified in the library, the value defaults to 1.0.

Related Information

Affects these commands: [report cell_delay_calculation](#)

[report slew_calculation](#)

[report timing](#)

slew_lower_threshold_pct_fall

`slew_lower_threshold_pct_fall float`

Read-only library attribute. Returns the default value of the lower threshold point used for modeling the delay of a pin falling from 1 to 0. If this attribute is not specified in the library, the value defaults to 20.0.

Related Information

Affects these commands:

[report cell_delay_calculation](#)
[report slew_calculation](#)
[report timing](#)

slew_lower_threshold_pct_rise

`slew_lower_threshold_pct_rise float`

Read-only library attribute. Returns the default value of the lower threshold point used for modeling the delay of a pin rising from 0 to 1. If this attribute is not specified in the library, the value defaults to 20.0.

Related Information

Affects these commands:

[report cell_delay_calculation](#)
[report slew_calculation](#)
[report timing](#)

slew_upper_threshold_pct_fall

`slew_upper_threshold_pct_fall float`

Read-only library attribute. Returns the default value of the upper threshold point used for modeling the delay of a pin falling from 1 to 0. If this attribute is not specified in the library, the value defaults to 80.0.

Related Information

Affects these commands:

[report cell delay calculation](#)
[report slew calculation](#)
[report timing](#)

slew_upper_threshold_pct_rise

`slew_upper_threshold_pct_rise float`

Read-only [library](#) attribute. Returns the default value of the upper threshold point used for modeling the delay of a pin rising from 0 to 1. If this attribute is not specified in the library, the value defaults to 80.0.

Related Information

Affects these commands:

[report cell delay calculation](#)
[report slew calculation](#)
[report timing](#)

time_scale_in_ps

`time_scale_in_ps integer`

Read-only [library](#) attribute. Returns the scaling factor used to compute any timing value in the library. All timing values in RTL Compiler are expressed in picoseconds.

Related Information

Affects this attribute:

[library](#) on page 274

usable_comb_cells

`usable_comb_cells integer`

Read-only [library](#) attribute. Returns the number of usable combinational cells in the library.

usable_seq_cells

`usable_seq_cells integer`

Read-only library attribute. Returns the number of usable sequential cells in the library.

usable_timing_models

`usable_timing_models integer`

Read-only `library` attribute. Returns the number of library cells of which RTL Compiler can understand the timing behavior, but not the combinational or sequential logic function. RTL Compiler will never map to these cells, but if you instantiate these cells, RTL Compiler can analyze their timing correctly. A RAM is the most common example.

version

`version string`

Read-only `library` attribute. Returns the version string for the library. The library creator supplies this string, therefore there is no convention for the version of a library.

Libcell Attributes

Contain information about a cell in the specified technology library.

- To set a libcell attribute, type

```
set_attribute attribute_name attribute_value \
[find /lib*/library -libcell cell]
```

- To get a libcell attribute value, type

```
get_attribute attribute_name [find /lib*/library -libcell cell]
```

adder

```
adder {true |false}
```

Read-only libcell attribute. Indicates if the libcell is a 1-bit half adder or full adder.

area

```
area float
```

Read-write libcell attribute. Specifies the area of the cell in the technology library. You can only overwrite this area for synthesis when the `interconnect_mode` design attribute is set to `wireload`.

Related Information

Affects these commands:

[report area](#)

[report gates](#)

Affects this attribute:

(design) [cell_area](#) on page 932

area_multiplier

`area_multiplier float`

Default: 1.0

Read-write libcell attribute. Specifies the area scaling factor to use for this cell. Increasing the multiplier can prevent the use of this cell during synthesis, while specifying a number smaller than the default, can favor the use of the cell. This attribute is applied to the cell area in both the Liberty and LEF cell libraries that are read.

Related Information

Affects these commands: [report area](#)
[report gates](#)
[synthesize](#)

Affects this attribute: (design) [cell_area](#) on page 932

async_clear

`async_clear libpins`

Read-only libcell attribute. Returns the full path to the library pin(s) that corresponds to the asynchronous clear pin of this library cell.

The attribute can return a collection of libpins in case of a multibit libcell with multiple asynchronous clear pins.

Note: An empty string indicates that this library cell is either not a sequential cell, or that the sequential cell has no asynchronous clear pin.

async_preset

`async_preset libpins`

Read-only libcell attribute. Returns the full path to the library pin that corresponds to the asynchronous preset pin of this library cell.

The attribute can return a collection of libpins in case of a multibit libcell with multiple asynchronous preset pins.

Note: An empty string indicates that this library cell is either not a sequential cell, or that the sequential cell has no asynchronous preset pin.

avoid

```
avoid {false | true}
```

Default: false

Read-write libcell attribute. When set to true, prevents a library cell from being used by the technology mapper. If certain library cells are not desired, setting the `avoid` attribute on them will cause them not to be selected during mapping.

Note: This attribute is automatically set to true when you set the `preserve` attribute on this libcell to true. Similarly, if you set the `avoid` attribute to false, the `preserve` attribute will implicitly be set to false. This action can override the `dont_touch` value specified in the Liberty file for the libcell.

Related Information

Affects this command: [synthesize](#)

Affected by this attribute: [usable](#) on page 210

Related attribute: [preserve](#) on page 206

buffer

```
buffer {true | false}
```

Read-only libcell attribute. Indicates if the libcell is a buffer.

Related Information

Related attribute: [\(instance\) buffer](#) on page 946

cell_bitwidth

`cell_bitwidth integer`

Read-only libcell attribute. Specifies the width of the cell in terms of number of unit components.

Related Information

Related attributes:	multibit cells from different busses on page 798 use multibit cells on page 826 use multibit combo cells on page 827 use multibit seq and tristate cells on page 828
---------------------	---

cell_delay_multiplier

`cell_delay_multiplier float`

Default: 1.0

Read-write libcell attribute. Specifies the scaling factor for all the delay arcs of this libcell.

This attribute can also be set by the following SDC command:

```
dc::set_timing_derate -late -data -cell_delay libcell_name
```

Example

The following command scales all delay arcs of cell DFFHQX1 by 4.3.

```
set_attr cell_delay_multiplier 4.3 [find /lib*/* -libcell DFFHQX1]
```

Related Information

Affects these commands:	report cell_delay_calculation report slew_calculation report timing
-------------------------	---

Related attributes:	cell_delay_multiplier on page 562
---------------------	---

cell_internal_power

`cell_internal_power {no_value | float}`

Default: no_value

Read-write libcell attribute. Specifies the internal power of the cell. The unit of the power value is determined by the value of the lp_power_unit attribute.

By default, the internal cell power is derived from the power tables in the library. When you overwrite the internal cell power, the tool issues a warning message.

Example

The following example sets the internal cell power of cell AND2A to 10.1. If you did not change the value of lp_power_unit attribute, the units are in nW.

```
rc:/> set_att cell_internal_power 10.1 [find / -libcell AND2A]
Warning : Overwrote internal power characterized in the .lib. [LBR-89]
          : Overwrote the internal power of library-cell
'/libraries/cg/libcells/AND2A' with the user defined value '10.100000'.
          : The user defined value will be used for power analysis.
```

Related Information

Affects these commands:

[report gates](#)

[report power](#)

[synthesize](#)

cell_leakage_power

`cell_leakage_power {no_value | float}`

Default: no_value

Read-write libcell attribute. Specifies the leakage power of the libcell. The unit of the power value is determined by the value of the leakage_power_scale_in_nw attribute.

Example

If the value of leakage_power_scale_in_nw is 0.01 and the value of cell_leakage_power is 1000, the leakage power of this cell is 10 nW.

Related Information

Affects these commands:

[report gates](#)
[report power](#)
[synthesize](#)

cell_min_delay_multiplier

`cell_min_delay_multiplier float`

Default: 1.0

Read-write [libcell](#) attribute. Scales the minimum delay of the libcell by the specified value. This scaled delay is used to compute the input delay to the from_pin during data-to-data checking.

This attribute can also be set by the following SDC command:

`dc::set_timing_derate -early -data -cell_delay libcell`

Related Information

[Setting Data-to-Data Checks](#) in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affects these commands:

[report cell_delay_calculation](#)
[report slew_calculation](#)
[report timing](#)

Related attribute:

[cell_min_delay_multiplier](#) on page 563

clock

`clock path_list`

Read-only [libcell](#) attribute. Returns the path to the library pins that correspond to the clock pins of this library cell.

For timing models, it returns the path to the pin which has the clock-constraints.

Note: An empty string indicates that this library cell is not a flip-flop cell.

clock_gating_integrated_cell

`clock_gating_integrated_cell string`

Read-only `libcell` attribute. Describes the functionality of the integrated clock-gating cell: type of sequential cell, whether the cell is appropriate for positive or negative-edge triggered registers, whether the test control logic is located before or after the latch or flip-flop, or does not exist, and whether the cell contains observability logic or not.

Note: An empty string indicates that the cell is not an integrated clock-gating cell.

Related Information

[Clock Gating Cell Specification](#) in the *Library Guide for Encounter RTL Compiler*.

[Using Attributes to Select an Integrated Clock-Gating Cell in the Technology Library in Low Power in Encounter RTL Compiler](#).

Affects this command: [synthesize](#)

Related attributes: [lp_clock_gating_add_obs_port](#) on page 1349

[lp_clock_gating_add_reset](#) on page 1350

[lp_clock_gating_control_point](#) on page 1355

[lp_clock_gating_style](#) on page 1359

combinational

`combinational {false | true}`

Read-only `libcell` attribute. Indicates if the libcell is a combinational cell.

congestion_avoid

`congestion_avoid {false | true}`

Default: `false`

Read-write `libcell` attribute. Indicates whether this libcell should be avoided in congested regions and replaced with other available libcells in the library.

Related Information

Affects this command: [synthesize -to_placed](#)

constraint_multiplier

`constraint_multiplier float`

Default: 1.0

Read-write libcell attribute. Specifies the scaling factor for all the constraint arcs (setup, removal, recovery) of this libcell.

Example

The following command scales the setup, removal and recovery arcs of cell DFFHQX1 by 4.3.

```
set_attr constraint_multiplier 4.3 [find /lib*/* -libcell DFFHQX1]
```

Related Information

Affects these commands:

[report cell_delay_calculation](#)
[report slew_calculation](#)
[report timing](#)

data_pins

`data_pins libpin_list`

Read-only libcell attribute. Returns the data pins that are used in the `next_state` statement of the sequential cell.

Example

Assume libcell `mycell` has the following `next_state` function:

```
next_state : "((SE&SD)|((!SE)&((D1&S)|(D2&(!S)))))"
```

The attribute will return:

```
rc:/> get_attr data_pins mycell
/libraries/mylib/libcells/mycell/D1 /libraries/mylib/libcells/mycell/D2
/libraries/mylib/libcells/mycell/S
```

direction

`direction {up | down | bidir}`

Read-write libcell attribute. Specifies the direction of the level shifter.

Attribute Reference for Encounter RTL Compiler

Library—Libcell Attributes

- `up` indicates that the level shifter connects from a lower voltage domain to a higher voltage domain.
- `down` indicates that the level shifter connects from a higher voltage domain to a lower voltage domain.
- `bidir` indicates that the direction of the level shifter can be up or down.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

flop

`flop {true | false}`

Read-only [libcell](#) attribute. Indicates if the libcell is a flip-flop.

Related Information

Related attribute: (instance) [flop](#) on page 948

ground_direction

`ground_direction {up | down | bidir}`

Read-write [libcell](#) attribute. Specifies the direction of the ground level shifter.

- `up` indicates that the level shifter connects from a lower voltage domain to a higher voltage domain.
- `down` indicates that the level shifter connects from a higher voltage domain to a lower voltage domain.
- `bidir` indicates that the direction of the level shifter can be up or down.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

height

`height float`

Read-only libcell attribute. Returns the height, in microns, of the object based on the information from the physical library.

Related Information

Related attributes:	(instance) height on page 411
	(instance) width on page 413
	(libcell) width on page 211

instance_probability

`instance_probability float`

Default: 0.0

Read-write libcell attribute. Specifies the probability that one instance of the cell will fail. The probability is specified as a floating point number (with double precision) between 0.0 and 1.0.

Related Information

Set by this command:	read_dfm
Related attribute:	systematic_probability on page 208

inverter

`inverter {true | false}`

Read-only libcell attribute. Indicates if the libcell is an inverter.

Related Information

Related attribute:	(instance) inverter on page 949
--------------------	---

is_always_on

```
is_always_on {false | true}
```

Default: false

Read-write libcell attribute. Indicates if the libcell is an always-on cell.

is_clock_gating_cell

```
is_clock_gating_cell {true | false}
```

Read-only libcell attribute. Indicates if the libcell is a clock-gating cell. This is only used for simple combinational clock-gating elements such as an AND and OR gates.

Note: RTL Compiler cannot use these cells for clock-gating mapping based on this attribute. For more information, refer to Using a Simple Combinational Gate in the *Library Guide for Encounter RTL Compiler*.

is_isolation_cell

```
is_isolation_cell {false | true}
```

Default: false

Read-write libcell attribute. Indicates if the libcell is an isolation cell. Isolation cells are used in n designs with switchable power domains.

Related Information

Related attribute: [isolation_cell_enable_pin](#) on page 218

is_level_shifter

```
is_level_shifter {false | true}
```

Default: false

Read-write libcell attribute. Indicates if the libcell is a level-shifter cell. Level shifters are used in MSV designs.

Related Information

Related attribute: [level_shifter_enable_pin](#) on page 219

latch

`latch {true | false}`

Read-only [libcell](#) attribute. Indicates if the libcell is a latch.

Related Information

Related attribute: [\(instance\) latch](#) on page 949

latch_enable

`latch_enable string`

Read-only [libcell](#) attribute. Returns the path to the library pin that corresponds to the latch enable pin of this library cell.

Note: An empty string indicates that this library cell is not a latch cell.

liberty_attributes

`liberty_attributes string`

Read-only [libcell](#) attribute. Returns a list of Liberty attributes and values that were specified for this cell in the library.

Related Information

Related attributes: [\(library\) liberty_attributes](#) on page 182
[\(libpin\) liberty_attributes](#) on page 219
[\(libarc\) liberty_attributes](#) on page 229
[\(operating conditions\) liberty_attributes](#) on page 232
[\(wireload\) liberty_attributes](#) on page 235

liberty_level_shifter_type

```
liberty_level_shifter_type {LH | HL | HL_LH}
```

Read-only libcell attribute. Returns the supported voltage conversion by the level shifter cell. Valid values for a level shifter are:

- LH—Low to High
- HL—High to Low
- HL_LH—High to Low and Low to High.

Note: The attribute value can be `null` if this library cell is not a level shifter, or if the level shifter does not have the `level_shifter_type` libcell attribute in the Liberty library. In the latter case, the default type for the level shifter is `HL_LH`.

master_slave_flop

```
master_slave_flop {false | true}
```

Read-only libcell attribute. Indicates if the libcell is a master-slave flip-flop.

master_slave_lssd_flop

```
master_slave_lssd_flop {false | true}
```

Read-only libcell attribute. Indicates if the libcell is a master-slave LSSD flip-flop.

max_ground_input_voltage

```
max_ground_input_voltage float
```

Default: 0.0

Read-write libcell attribute. Specifies the maximum voltage for the input (source) ground supply voltage that can be handled by this level shifter cell.

Note: This attribute only applies to ground level shifters.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (level_shifter_group) [max_ground_input_voltage](#) on page 1432

(level_shifter_group) [min_ground_input_voltage](#) on page 1434

(libcell) [min_ground_input_voltage](#) on page 202

max_ground_output_voltage

`max_ground_output_voltage float`

Default: 0.0

Read-write [libcell](#) attribute. Specifies the maximum voltage for the output (destination) ground supply voltage that can be handled by this level shifter.

Note: This attribute only applies to ground level shifters.

Related Information

Affected by this command: [read_power_intent](#)

Related attribute: (level_shifter_group) [max_ground_output_voltage](#) on page 1433

(level_shifter_group) [min_ground_output_voltage](#) on page 1434

(libcell) [min_ground_output_voltage](#) on page 203

max_input_voltage

`max_input_voltage float`

Default: 0.0

Read-write [libcell](#) attribute. Specifies the upper bound of the voltage range (in volt) that can be handled by the level shifter for the source domain.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

Related attribute: (level_shifter_group) [max_input_voltage](#) on page 1433

max_output_voltage

`max_output_voltage float`

Default: 0.0

Read-write libcell attribute. Specifies the upper bound of the voltage range (in volt) that can be handled by the level shifter for the destination domain.

Related Information

Affected by this command:	read_power_intent
Affected by this attribute:	(library_domain) library on page 1427
Related attribute:	(level_shifter_group) max_output_voltage on page 1433

min_ground_input_voltage

`min_ground_input_voltage float`

Default: 0.0

Read-write libcell attribute. Specifies the minimum voltage for the input (source) ground supply voltage that can be handled by this level shifter.

Note: This attribute only applies to ground level shifters.

Related Information

Affected by this command:	read_power_intent
Related attributes:	(level_shifter_group) min_ground_input_voltage on page 1434
	(level_shifter_group) max_ground_input_voltage on page 1432
	(libcell) max_ground_input_voltage on page 200

min_ground_output_voltage

`min_ground_output_voltage float`

Default: 0.0

Read-write libcell attribute. Specifies the minimum voltage for the output (destination) ground supply voltage that can be handled by this level shifter.

Note: This attribute only applies to ground level shifters.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (level_shifter_group) [min_ground_output_voltage](#) on page 1434

(level_shifter_group) [max_ground_output_voltage](#) on page 1433

(libcell) [max_ground_output_voltage](#) on page 201

min_input_voltage

`min_input_voltage float`

Default: 0.0

Read-write libcell attribute. Specifies the lower bound of the voltage range (in volt) that can be handled by the level shifter for the source domain.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

Related attribute: (level_shifter_group) [min_input_voltage](#) on page 1435

min_output_voltage

`min_output_voltage float`

Default: 0.0

Read-write libcell attribute. Specifies the lower bound of the voltage range (in volt) that can be handled by the level shifter for the destination domain.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

Related attribute: (level_shifter_group) [min_output_voltage](#) on page 1435

non_seq_setup_arc

`non_seq_setup_arc {false | true}`

Read-only libcell attribute. Specifies whether the libcell has an enabled non_seq_setup timing arc.

Related Information

Related attribute: (libpin) [non_seq_setup_arc](#) on page 222

pad

`pad {true | false}`

Read-only libcell attribute. Indicates if the libcell is a pad cell.

The attribute value will be set to `true` if the cell has a corresponding cell in the LEF library defined as CLASS PAD.

Related Information

Related attribute: [pad](#) on page 222

power_gating_cell

`power_gating_cell {false | true}`

Default: false

Read-write libcell attribute. Indicates if the cell is configured to support state-retention power gating (SRPG).

Related Information

Related attributes: [power_gating_cell_type](#) on page 205
 [power_gating_pin_class](#) on page 223
 [power_gating_pin_phase](#) on page 224

power_gating_cell_type

`power_gating_cell_type string`

Read-write libcell attribute. Identifies the type of a state-retention cell. By default, the type corresponds to the value of the `power_gating_cell` Liberty cell attribute.

Related Information

Related attributes: [power_gating_cell](#) on page 205
 [power_gating_pin_class](#) on page 223
 [power_gating_pin_phase](#) on page 224

preserve

`preserve {false | true}`

Default: false

Read-write libcell attribute. Preserves instances of this cell from optimization.

By default, RTL Compiler performs optimizations that can result in logic changes to any object in the design. You can prevent any logic changes on instances of this cell while still allowing mapping optimizations in the surrounding logic.

Note: Setting this attribute to `true`, causes the tool to automatically set the `avoid` attribute on this libcell to `true`, which prevents the tool from using the cell during mapping or optimization. If the libcell was already instantiated in the design before the `preserve` libcell attribute was set to `true`, the instances would remain in the design.

Related Information

Affects this command:	synthesize
Affects this attribute:	avoid on page 190
Related attributes:	(design) preserve on page 838 (instance) preserve on page 855 (net) preserve on page 876 (subdesign) preserve on page 909

scan_enable

`scan_enable libpin_list`

Read-only libcell attribute. Returns the path(s) to the library pin(s) that correspond(s) to the scan enable pin(s) of this library cell.

Note: An empty string indicates that this library cell is not a scan flip-flop.

scan_in

`scan_in string`

Read-only libcell attribute. Returns the path to the library pin that corresponds to the scan data input pin of this library cell.

Note: An empty string indicates that this library cell is not a scan flip-flop.

sequential

`sequential {true | false}`

Read-only libcell attribute. Indicates if the libcell is a sequential logic circuit.

Related Information

Related attribute: (instance) [sequential](#) on page 952

sync_clear

`sync_clear string`

Read-only libcell attribute. Returns the path to the library pin that corresponds to the synchronous clear pin of this library cell.

Note: An empty string indicates that this library cell either is not a flip-flop cell, or has no synchronous clear pin.

Related Information

Related attribute: [use_nextstate_type_only_to_assign_sync_ctrls](#) on page 829

sync_enable

`sync_enable string`

Read-only libcell attribute. Returns the path to the library pin that corresponds to the synchronous enable pin of this library cell.

Note: An empty string indicates that this library cell either is not a flip-flop cell, or has no synchronous enable pin.

sync_preset

`sync_preset string`

Read-only libcell attribute. Returns the path to the library pin that corresponds to the synchronous preset pin of this library cell.

Note: An empty string indicates that this library cell either is not a flip-flop cell, or has no synchronous preset pin.

systematic_probability

`systematic_probability float`

Default: 0.0

Read-write libcell attribute. Specifies the probability of a systematic failure of the cell, that is, the probability that one or more instances of the cell will fail. The probability is specified as a floating point number (with double precision) between 0.0 and 1.0.

Related Information

Set by this command: [read_dfm](#)

Related attribute: [instance_probability](#) on page 197

timing_model

`timing_model {true | false}`

Read-only libcell attribute. Indicates if the internal functionality of the libcell is unknown.

Related Information

[Cells Identified as Timing Models in Using Encounter RTL Compiler](#)

Related attributes: [\(instance\) timing_model](#) on page 955

[timing_model_reason](#) on page 209

timing_model_reason

`timing_model_reason string`

Read-only libcell attribute. Returns the reason why the library cell is considered a timing model.

Related Information

Cells Identified as Timing Models in *Using Encounter RTL Compiler*

Related attributes: (instance) [timing_model](#) on page 955
(libcell) [timing_model](#) on page 208

tristate

tristate {true | false}

Read-only `libcell` attribute. Indicates if the libcell has at least one tristate output.

Related Information

Related attribute: (instance) [tristate](#) on page 955
(libpin) [tristate](#) on page 226

unable_reason

`unusable_reason string`

Read-only libcell attribute. Returns the reason why the library cell is considered unusable.

Related Information

Cells Identified as Unusable in Using *Encounter* RTL Compiler

Related attribute: [usable on page 210](#)

usable

```
usable {true | false}
```

Read-only libcell attribute. Indicates if the libcell can be used during mapping or incremental optimization.

If the tool cannot infer the libcell, or the libcell cannot be used during mapping or incremental optimization, this attribute will be set to `false`.

Even when the `avoid` attribute can indicate that a libcell can be used for mapping or optimization, the tool can consider the libcell not to be usable, for example because the function or timing is too complex.

Related Information

Cells Identified as Unusable in *Using Encounter RTL Compiler*

Affects this attribute: [avoid](#) on page 190

user_defined

```
user_defined string
```

Read-write libcell attribute. Provided to make Tcl scripting easier. The specified string contains the user-defined attribute name and attribute value.

Related Information

Related attributes:

(design) user_defined on page 1474
(instance) user_defined on page 1475
(libpin) user_defined on page 227
(pin) user_defined on page 1476
(net) user_defined on page 1478
(port) user_defined on page 1479
(subdesign) user_defined on page 1480
(subport) user_defined on page 1481

width

`width float`

Read-only libcell attribute. Returns the width, in microns, of the cell based on the information from the physical library.

Related Information

Related attributes:

- (instance) height on page 411
- (instance) width on page 413
- (libcell) width on page 211

Libpin Attributes

Contain information about a pin of the specified technology library cell.

- To set a libpin attribute, type

```
set_attribute attribute_name attribute_value \
[find /lib*/library -libpin libcell/pin]
```

- To get a libpin attribute value, type

```
get_attribute attribute_name [find /lib*/library -libpin libcell/pin]
```

async_clear_phase

```
async_clear_phase {active_high | active_low | none}
```

Read-only libpin attribute. Returns the active phase of the asynchronous clear pin of a sequential cell.

Note: The value `none` indicates that the pin is not an asynchronous clear pin.

async_preset_phase

```
async_preset_phase {active_high | active_low | none}
```

Read-only libpin attribute. Returns the active phase of the asynchronous preset pin of a sequential cell.

Note: The value `none` indicates that this pin is not an asynchronous preset pin.

bundle

```
bundle string_list
```

Read-only libpin attribute. Returns for every pin that is part of a bundle, a string of the following format:

```
bundle bundle_name: bundle_member bundle_member ...
```

For multi-bit cells, all pins that have similar timing or functionality are grouped in a bundle. The attribute value for all bundle members will be the same.

Note: This attribute has no value for single-bit cells.

capacitance

`capacitance string`

Read-write libpin attribute. Specifies the rise and fall capacitive load of the pin in femtofarads.

clock_gate_clock_pin

`clock_gate_clock_pin {true | false}`

Read-only libpin attribute. Indicates if this pin is a clock pin on a clock-gating cell.

Related Information

Affects this command: [synthesize](#)

clock_gate_enable_pin

`clock_gate_enable_pin {true | false}`

Read-only libpin attribute. Indicates if this pin is an enable pin on a clock-gating cell.

Related Information

Affects this command: [synthesize](#)

clock_gate_obs_pin

`clock_gate_obs_pin {true | false}`

Read-only libpin attribute. Indicates if this pin is an observable pin on a clock-gating cell.

Related Information

Affects this command: [synthesize](#)

clock_gate_out_pin

`clock_gate_out_pin {true | false}`

Read-only libpin attribute. Indicates if this pin is an output pin on a clock-gating cell.

Related Information

Affects this command: [synthesize](#)

clock_gate_reset_pin

`clock_gate_reset_pin {true | false}`

Read-only libpin attribute. Indicates if this pin is an asynchronous reset pin on a clock-gating cell.

Related Information

Affects this command: [synthesize](#)

clock_gate_test_pin

`clock_gate_test_pin {true | false}`

Read-only libpin attribute. Indicates if this pin is a test pin on a clock-gating cell.

Related Information

Affects this command: [synthesize](#)

clock_phase

`clock_phase {active_high | active_low | none}`

Read-only libpin attribute. Returns the active phase of the clock pin of a sequential cell

Note: The value `none` indicates that this pin is not a clock pin of a sequential cell.

driver_type

```
driver_type {bus_hold | open_drain | open_source | pull_up | pull_down  
| resistive | resistive_0 | resistive_1}
```

Read-only libpin attribute. Returns the driver type of the output or inout pin.

If the pin is an inout pin, the attribute can have a driver type for the input and output. In this case the following applies:

- if `pull_up` or `pull_down` is returned with `open_drain`, the `pull_up` or `pull_down` value will be applied to the input, while the `open_drain` will be applied to the output
- if the value returned is `bus_hold`, it will be applied to the input and output
- if the value returned is not `bus_hold`, it will be applied to the output.

For output pins only one of the possible values can be returned.

Note: This attribute has no value for input pins.

Related Information

Related command: [report dft violations](#)

fall_capacitance_range

```
fall_capacitance_range min_cap max_cap
```

Read-only libpin attribute. Returns a range of values for the pin capacitance during a fall transition.

Example

```
rc:/> get_attr fall_capacitance_range [find /lib*/* -libpin HD65_LS_MSFPRBX3/A]  
0.00142144 0.00143265
```

Related Information

Related attribute: [rise_capacitance_range](#) on page 225

fanout_load

`fanout_load float`

Read-only libpin attribute. Specifies the internal fanout of the input pin. Resolution is 1/1000. Typical units are standard loads or pin count.

function

`function string`

Read-only libpin attribute. Specifies the value (Boolean expression) of an output pin as a function of the cell's input or inout pins.

Example

```
rc:/libraries/tutorial/libcells> get_att function nor2/Y  
(A + B)'
```

generated_clock

`generated_clock string`

Read-only libpin attribute. Returns the Liberty attributes and corresponding values that were specified in the `generated_clock` group for the libcell to which this pin belongs.

The following Liberty attributes *can* be specified in a `generated_clock` group:

- `clock_pin`
- `master_pin`
- `divided_by`
- `multiplied_by`
- `invert`
- `duty_cycle`
- `edges`
- `shifts`

ground

```
ground {false | true}
```

Read-only libpin attribute. Indicates if the libpin is a ground pin.

higher_drive

```
higher_drive string
```

Read-only libpin attribute. Returns the path to the library pin of a cell that has the same functionality, but has a higher drive strength. If your library contains several cells with the same functionality, this attribute points to the pin with the next higher drive strength. The attribute value on the pin with the highest drive strength is an empty string.

Related Information

Related attribute: [lower_drive](#) on page 219

incoming_timing_arcs

```
incoming_timing_arcs string
```

Read-only libpin attribute. Returns a list of incoming timing arcs.

input

```
input {true | false}
```

Read-only libpin attribute. Indicates if this pin is an input pin.

internal

```
internal {true | false}
```

Read-only libpin attribute. Indicates if this pin is an internal pin.

is_always_on

```
is_always_on {false | true}
```

Default: false

Read-write libpin attribute. Indicates if this pin is an always-on pin on an always-on cell.

Related Information

Set by this command: [read_power_intent](#)

is_iq_function

```
is_iq_function {true | false}
```

Read-only libpin attribute. Indicates if this pin is an IQ (noninverting output) pin.

is_iqn_function

```
is_iqn_function {true | false}
```

Read-only libpin attribute. Indicates if this pin is an IQN (inverting output) pin.

isolation_cell_enable_pin

```
isolation_cell_enable_pin {false | true}
```

Default: false

Read-write libpin attribute. Indicates if this pin is the enable pin of an isolation cell.

Related Information

Related attribute: [is_isolation_cell on page 198](#)

latch_enable_phase

```
latch_enable_phase {active_high | active_low | none}
```

Read-only libpin attribute. Returns the active phase of the latch enable pin of a sequential cell.

Note: The value `none` indicates that this pin is not a latch input enable.

level_shifter_enable_pin

`level_shifter_enable_pin {false | true}`

Default: false

Read-write [libpin](#) attribute. Indicates if this pin is the enable pin of a level shifter cell.

Related Information

Related attribute: [is_level_shifter](#) on page 198

liberty_attributes

`liberty_attributes string`

Read-only [libpin](#) attribute. Returns a list of Liberty attributes and values that were specified for this pin in the library.

Related Information

Related attributes: [\(library\) liberty_attributes](#) on page 182
[\(libcell\) liberty_attributes](#) on page 199
[\(libarc\) liberty_attributes](#) on page 229
[\(operating conditions\) liberty_attributes](#) on page 232
[\(wireload\) liberty_attributes](#) on page 235

lower_drive

`lower_drive string`

Read-only [libpin](#) attribute. Returns the path to the library pin of a cell that has the same functionality, but has a lower drive strength. If your library contains several cells with the same functionality, this attribute points to the pin with the next lower drive strength. The attribute value on the pin with the lowest drive strength is an empty string.

Related Information

Related attribute: [higher_drive](#) on page 217

max_capacitance

`max_capacitance float`

Read-write libpin attribute. Specifies the maximum capacitance in femtofarads that an output pin can drive.

Note: This attribute has no value for input pins.

Related Information

Affected by this attribute: [ignore_library_drc on page 549](#)

Related attributes: (design) [max_capacitance on page 557](#)
(port) [max_capacitance on page 651](#)

max_fanout

`max_fanout float`

Read-write libpin attribute. Specifies the maximum fanout that an output pin of the library cell can drive.

Note: This attribute has no value for input pins.

Related Information

Affected by this attribute: [ignore_library_drc on page 549](#)

Related attributes: (design) [max_fanout on page 558](#)
(port) [max_fanout on page 652](#)

max_transition

`max_transition float`

Read-write libpin attribute. Specifies the maximum acceptable transition time on the library pin.. This attribute applies to input and output pins.

Related Information

Affected by this attribute: [ignore_library_drc on page 549](#)

Related attributes: (design) [max_transition](#) on page 559
(port) [max_transition](#) on page 653

min_capacitance

`min_capacitance float`

Read-write [libpin](#) attribute. Specifies the minimum capacitance in femtofarads that an output pin can drive.

Minimum DRC values are used to ensure the predictability of the timing by ensuring it falls within the characterization range, and to ensure that high drive cells are only used when needed.

Note: This attribute has no value for input pins.

min_fanout

`min_fanout float`

Read-write [libpin](#) attribute. Specifies the minimum fanout that an output pin of the library cell can drive.

Minimum DRC values are used to ensure the predictability of the timing by ensuring it falls within the characterization range, and to ensure that high drive cells are only used when needed.

Note: This attribute has no value for input pins.

min_transition

`min_transition float`

Read-write [libpin](#) attribute. Specifies the minimum acceptable transition time on the library pin. This attribute applies to input and output pins.

Minimum DRC values are used to ensure the predictability of the timing by ensuring it falls within the characterization range, and to ensure that high drive cells are only used when needed.

non_seq_setup_arc

`non_seq_setup_arc {false | true}`

Read-only libpin attribute. Specifies whether the pin has an enabled `non_seq_setup` timing arc.

Related Information

Related attribute: (libcell) [non_seq_setup_arc](#) on page 204

outgoing_timing_arcs

`outgoing_timing_arcs string`

Read-only libpin attribute. Returns a list of outgoing timing arcs.

output

`output {true | false}`

Read-only libpin attribute. Indicates if this pin is an output.

pad

`pad {true | false}`

Read-only libpin attribute. Indicates if the libpin is a pad pin.

Related Information

Related attribute: [pad](#) on page 204

polarity

`polarity {high | low}`

Default: high

Read-write libpin attribute. Specifies the polarity of the test or reset pin of an integrated clock-gating cell.

Related Information

Related attributes: [lp_insert_clock_gating](#) on page 1343

[lp_clock_gating_control_point](#) on page 1355

power

`power {false | true}`

Read-only libpin attribute. Indicates if the libpin is a power pin.

power_gating_pin_class

`power_gating_pin_class string`

Read-write libpin attribute. Specifies the class of the power gating pin of a state-retention cell. The value is of the form `power_pin_x`, where `x` can take an integer value from 1 through 5.

Note: The value corresponds to the first part of the `power_gating_pin` pin Liberty attribute value.

Related Information

Related attributes: [power_gating_cell](#) on page 205

Related attributes: [power_gating_cell_type](#) on page 205

[power_gating_pin_phase](#) on page 224

power_gating_pin_phase

`power_gating_pin_phase {none | active_low | active_high}`

Default: none

Read-write libpin attribute. Specifies the active phase of the pin of a state-retention cell.

- `active_low` indicates that an active low signal (0) applied to this pin puts the gate in sleep mode.
- `active_high` indicates that an active high signal (1) applied to this pin puts the gate into sleep mode.

The value of this attribute is the opposite of the value of the `power_gating_pin` pin Liberty attribute.

Note: The value `none` indicates that the pin is not a power gating pin.

Related Information

Related attributes:

[power_gating_cell](#) on page 205

[power_gating_cell_type](#) on page 205

[power_gating_pin_class](#) on page 223

q_pin_of_d_pin

`q_pin_of_d_pin libpin`

Read-only libpin attribute. Returns the output pin that corresponds to the data pin of the sequential cell. Use this for sequential cells with multiple data pins and output pins.

Example

Assume libcell `mycell` has the following data pins: D1, D2, S

To find the corresponding output (Q) pin for data pin D1, use the following command:

```
rc:/> get_attr q_pin_of_d_pin mycell/D1  
/libraries/mylib/libcells/mycell/Q
```

rise_capacitance_range

```
rise_capacitance_range min_cap max_cap
```

Read-only libpin attribute. Returns a range of values for the pin capacitance during a rise transition.

Example

```
rc:/> get_attr rise_capacitance_range [find /lib*/* -libpin HD65_LS_MSFPBX3/A]  
0.00138321 0.00145612
```

Related Information

Related attribute: [fall_capacitance_range](#) on page 215

scan_enable_phase

```
scan_enable_phase {active_high | active_low | none}
```

Read-only libpin attribute. Returns the active phase of the scan enable pin of a scan flip-flop.

Note: The value `none` indicates that the pin is not a scan enable pin.

scan_in_phase

```
scan_in_phase {active_high | active_low | none}
```

Read-only libpin attribute. Returns the active phase of the scan data input pin of a scan flip-flop.

The value `none` indicates that the pin is not a scan data input pin.

signal_level

```
signal_level string
```

Read-only libpin attribute. Returns the name of the power supply that the pin is connected to. This information is defined in the .lib file through the `input_signal_level` (`output_signal_level`) attribute for an input (output) pin.

sync_clear_phase

```
sync_clear_phase {active_high | active_low | none}
```

Read-only libpin attribute. Returns the active phase of the synchronous clear pin of a sequential cell.

Note: The value `none` indicates that the pin is not a synchronous clear pin.

sync_enable_phase

```
sync_enable_phase {active_high | active_low | none}
```

Read-only libpin attribute. Returns the active phase of the synchronous enable pin of a sequential cell.

The value `none` indicates that the pin is not a synchronous enable pin.

sync_preset_phase

```
sync_preset_phase {active_high | active_low | none}
```

Read-only libpin attribute. Returns the active phase of the synchronous preset pin of a sequential cell.

Note: The value `none` indicates that this pin is not an synchronous preset pin.

tristate

```
tristate {true | false}
```

Read-only libpin attribute. Indicates if this pin is a tristate output.

Related Information

Related attributes: (instance) [tristate](#) on page 955

(libcell) [tristate](#) on page 209

use

```
use {signal | analog | clock | ground | power}
```

Read-only libpin attribute. Returns the use of the pin. If the cell was read from a library in Liberty format, the use will be signal. If a LEF library is read later, the value of this attribute might change if the pin's use was defined through the USE statement in the LEF library.

Related Information

Related attributes: [lef_library](#) on page 272
 [library](#) on page 274

user_defined

```
user_defined string
```

Read-write libpin attribute. Provided to make Tcl scripting easier. The specified string contains the user-defined attribute name and attribute value.

Related Information

Related attributes: (design) [user_defined](#) on page 1474
 (instance) [user_defined](#) on page 1475
 (libcell) [user_defined](#) on page 210
 (pin) [user_defined](#) on page 1476
 (net) [user_defined](#) on page 1478
 (port) [user_defined](#) on page 1479
 (subdesign) [user_defined](#) on page 1480
 (subport) [user_defined](#) on page 1481

user_function

```
user_function string
```

Read-write libpin attribute. Specifies the user-defined function for the cell on the output pin.

Attribute Reference for Encounter RTL Compiler

Library—Libpin Attributes

x_offset

`x_offset float`

Read-only `libpin` attribute. Specifies the x-offset (in microns) of the corresponding LEF cell.

y_offset

`y_offset float`

Read-only `libpin` attribute. Specifies the y-offset (in microns) of the corresponding LEF cell.

Libarc Attributes

Contain information about a timing path between two pins of the specified technology library cell.

- To set a libarc attribute, type

```
set_attribute attribute_name attribute_value \
[find /lib*/library -libarc libcell/output_pin/input_pin*]
```

- To get a libarc attribute value, type

```
get_attribute attribute_name
[find /lib*/library -libarc libcell/output_pin/input_pin*]
```

enabled

`enabled {true | false}`

Default: true

Read-write libarc attribute. Generally, all timing arcs (libarcs) defined in the library are valid. Using this attribute you can disable one or more timing arcs of a particular cell in the design, or re-enable a previously disabled arc.

Related Information

[Disabling Timing Arcs in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: [real_enabled](#) on page 230

from_pin

`from_pin path`

Read-only libarc attribute. Specifies the path to the pin where this timing path originated.

liberty_attributes

`liberty_attributes string`

Read-only libarc attribute. Returns a list of Liberty attributes and values that were specified for this timing arc in the library.

Related Information

Related attributes:	(library) liberty_attributes on page 182
	(libcell) liberty_attributes on page 199
	(libpin) liberty_attributes on page 219
	(operating conditions) liberty_attributes on page 232
	(wireload) liberty_attributes on page 235

real_enabled

```
real_enabled {true | false}
```

Read-only [libarc](#) attribute. Represents the enabling of the timing arc internally.

Even when the `enabled` attribute indicates that the timing arc is enabled, the tool can consider it otherwise.

Related Information

Affects this attribute:	enabled on page 229
-------------------------	-------------------------------------

type

```
type { borrow_arc | clear_arc | clock_edge_arc | combo_arc | hold_arc  
| nonseq_hold_arc | nonseq_setup_arc | preset_arc | recovery_arc | removal_arc  
| setup_arc | tristate_disable_arc | tristate_enable_arc }
```

Read-only [libarc](#) attribute. Represents the timing arc type.

Seq Function Attributes

Contain information about the function of the specified sequential cell.

- To get a seq_function attribute value, type

```
get_attribute attribute_name [find /lib*/library -seq_function libcell/*]
```

d_function

d_function string

Read-only seq_function attribute. Returns the function of the sequential cell.

Example

The following example returns the function of sequential cell SEDFFTRX1:

```
rc:/> get_att d_function [find /lib/* -seq_function SEDFFTRX1/*]  
(SE SI) + (SE' ((E RN D) + (E' RN IQ)))
```

Operating Conditions Attributes

Contain information about the operating conditions for which the specified technology library is characterized.

- To set an `operating_condition` attribute, type

```
set_attribute attribute_name attribute_value \
[find /lib*/library -operating_condition condition]
```

- To get an `operating_condition` attribute value, type

```
get_attribute attribute_name \
[find /lib*/library -operating_condition condition]
```

liberty_attributes

`liberty_attributes string`

Read-only `operating_condition` attribute. Returns a list of Liberty attributes and values that were specified for the operating condition in the library.

Related Information

Related attributes:	(library) liberty_attributes on page 182
	(libcell) liberty_attributes on page 199
	(libpin) liberty_attributes on page 219
	(libarc) liberty_attributes on page 229
	(wireload) liberty_attributes on page 235

process

`process string`

Read-write `operating_condition` attribute. Specifies the process value.

This attribute is useful when tracking down timing discrepancies between different tools. A common source of timing discrepancies is caused by differences in the technology library files.

temperature

`temperature string`

Read-write operating condition attribute. Specifies the operating temperature.

This attribute is useful when tracking down timing discrepancies between different tools. A common source of timing discrepancies is caused by differences in the technology library files.

tree_type

`tree_type {balanced_tree | best_case_tree | worst_case_tree}`

Read-write operating condition attribute. Specifies the wire delay estimation method.

This attribute is useful when tracking down timing discrepancies between different tools. A common source of timing discrepancies is caused by differences in the technology library files.

voltage

`voltage string`

Read-write operating condition attribute. Specifies the operating voltage.

This attribute is useful when tracking down timing discrepancies between different tools. A common source of timing discrepancies is caused by differences in the technology library files.

If the voltage is missing, RTL Compiler defaults to the nominal voltage. If the nominal voltage value is missing, RTL Compiler deaaults to 1.0V.

Wireload Attributes

Contain information about the available wire-load models in the technology library.

- To set a wireload attribute, type

```
set_attribute attribute_name attribute_value \
[find /lib*/library -wireload model]
```

- To get a wireload attribute value, type

```
get_attribute attribute_name [find /lib*/library -wireload model]
```

fanout_cap

`fanout_cap string`

Read-write wireload attribute. Sets the capacitance per fanout for the wire-load model.

Example

```
rc:/libraries/my_lib/wireload_models> set_att fanout_cap {{3 128} {5 543}} 30x30
  Setting attribute of wireload 30x30: 'fanout_cap' = {{3 128.00}{5 543.00}}
rc:/libraries/my_lib/wireload_models> ls -a -l 30x30
/libraries/tsmc_25/wireload_models/30x30 (wireload)
  All attributes:
    fanout_cap = {{3 128.00}{5 543.00}}
    liberty_attributes = area 0 capacitance 1 resistance 0 slope 0.782
  Additional information:

  NOTE: lu=library units
  fanout      :       1       2       3       4       5       6
  cap(fF)     :     42.7     85.3   128.0   335.5   543.0  1325.0
  res(kOhm)   :      0.00      0.00      0.00      0.00      0.00      0.00
  area(lu)    :      0.00      0.00      0.00      0.00      0.00      0.00
```

Related Information

Affects these commands:

[synthesize](#)

[report area](#)

[report design_rules](#)

[report gates](#)

[report summary](#)

[report timing](#)

liberty_attributes

`liberty_attributes string`

Read-only wireload attribute. Returns a list of Liberty attributes and values that were specified for the wireload_model in the library.

Related Information

Related attributes:	(library) liberty_attributes on page 182
	(libcell) liberty_attributes on page 199
	(libpin) liberty_attributes on page 219
	(libarc) liberty_attributes on page 229
	(operating conditions) liberty_attributes on page 232

Attribute Reference for Encounter RTL Compiler

Library—Wireload Attributes

Input and Output

Root Attributes

- [bit blasted port style](#) on page 243
- [ccd executable](#) on page 244
- [command log](#) on page 244
- [detailed sdc messages](#) on page 244
- [encounter executable](#) on page 247
- [error on lib lef pin inconsistency](#) on page 247
- [ets executable](#) on page 248
- [group generate portname from netname](#) on page 248
- [group instance suffix](#) on page 250
- [hdl allow inout const port connect](#) on page 250
- [hdl allow instance name conflict](#) on page 251
- [hdl ignore pragma names](#) on page 252
- [hdl keep first module definition](#) on page 252
- [hdl language](#) on page 252
- [hdl link from any lib](#) on page 253
- [hdl max memory address range](#) on page 253
- [hdl nc compatible module linking](#) on page 253
- [hdl primitive input multibit](#) on page 255
- [hdl report case info](#) on page 256
- [hdl search path](#) on page 256

Attribute Reference for Encounter RTL Compiler

Input and Output—List

- [hdl track filename row col](#) on page 257
- [hdl use current dir before hdl search path](#) on page 258
- [hdl verilog defines](#) on page 259
- [hdl vhdl assign width mismatch](#) on page 260
- [hdl vhdl case](#) on page 261
- [hdl vhdl environment](#) on page 261
- [hdl vhdl lrm compliance](#) on page 262
- [hdl vhdl preferred architecture](#) on page 262
- [hdl vhdl range opto](#) on page 263
- [hdl vhdl read version](#) on page 264
- [hdl zero replicate is null](#) on page 264
- [input assert one cold pragma](#) on page 265
- [input assert one hot pragma](#) on page 265
- [input asynchro reset blk pragma](#) on page 265
- [input asynchro reset pragma](#) on page 266
- [input case cover pragma](#) on page 267
- [input case decode pragma](#) on page 267
- [input map to mux pragma](#) on page 268
- [input pragma keyword](#) on page 268
- [input synchro enable blk pragma](#) on page 269
- [input synchro enable pragma](#) on page 269
- [input synchro reset blk pragma](#) on page 269
- [input synchro reset pragma](#) on page 270
- [inst prefix](#) on page 271
- [lbr respect async controls priority](#) on page 271
- [lec executable](#) on page 271
- [lef library](#) on page 272

Attribute Reference for Encounter RTL Compiler

Input and Output—List

- [lib_search_path](#) on page 273
- [library](#) on page 274
- [nc_protect_version](#) on page 275
- [path](#) on page 275
- [rc_verification_directory](#) on page 276
- [script_begin](#) on page 278
- [script_end](#) on page 278
- [script_search_path](#) on page 279
- [supportAppendingLibs](#) on page 280
- [supportMultiSeqElements](#) on page 280
- [synthesisOffCommand](#) on page 281
- [synthesisOnCommand](#) on page 281
- [ungroupSeparator](#) on page 282
- [usePowerGroundPinFromlef](#) on page 282
- [wccdThresholdPercentage](#) on page 282
- [wcdcClockDomCombPropagation](#) on page 283
- [wclplibStatetable](#) on page 284
- [wlecAddNoblackBoxRetimeSubdesign](#) on page 284
- [wlecAnalyzeAbort](#) on page 285
- [wlecAnalyzeSetup](#) on page 286
- [wlecAutoAnalyze](#) on page 286
- [wlecCompareThreads](#) on page 287
- [wlecCutPoint](#) on page 288
- [wlecHierCompThreshold](#) on page 288
- [wlecLibStatetable](#) on page 289
- [wlecParallelThreads](#) on page 289
- [wlecSetCdnSynthRoot](#) on page 290

Attribute Reference for Encounter RTL Compiler

Input and Output—List

- [wlec uniquify](#) on page 290
- [wlec use lec model](#) on page 291
- [write sv port wrapper](#) on page 292
- [write vlog bit blast bus connections](#) on page 293
- [write vlog bit blast constants](#) on page 294
- [write vlog bit blast mapped ports](#) on page 296
- [write vlog bit blast tech cell](#) on page 298
- [write vlog convert onebit vector to scalar](#) on page 301
- [write vlog declare wires](#) on page 302
- [write vlog empty module for black box](#) on page 304
- [write vlog empty module for logic abstract](#) on page 305
- [write vlog generic gate define](#) on page 307
- [write vlog line wrap limit](#) on page 308
- [write vlog no negative index](#) on page 309
- [write vlog preserve net name](#) on page 310
- [write vlog top module first](#) on page 311
- [write vlog unconnected port style](#) on page 312
- [write vlog wor wand](#) on page 314

Design Attributes

- [arch filename](#) on page 317
- [embedded script](#) on page 318
- [entity filename](#) on page 318
- [hdl_all_filelist](#) on page 319
- [hdl_config_name](#) on page 320
- [hdl_filelist](#) on page 321
- [hdl_user_name](#) on page 323

Attribute Reference for Encounter RTL Compiler

Input and Output—List

- [hdl_v2001](#) on page 323
- [protected](#) on page 324
- [wcdc_synchronizer_type](#) on page 324

Instance Attributes

- [hdl_v2001](#) on page 326
- [write_vlog_port_association_style](#) on page 327

Pin Attributes

- [hdl_v2001](#) on page 328

Pgpin Attributes

- [hdl_v2001](#) on page 329

hdl_arch Attributes

- [encrypted](#) on page 330
- [parameters](#) on page 330
- [pins](#) on page 331
- [start_source_line](#) on page 331
- [structural](#) on page 332
- [verilog_macros](#) on page 332

hdl_config Attributes

- [entity](#) on page 333
- [location](#) on page 333

hdl_inst Attributes

- [component](#) on page 334

Attribute Reference for Encounter RTL Compiler

Input and Output—List

hdl_pack Attributes

- [default_location](#) on page 335
- [location](#) on page 335

Port Attributes

- [hdl_v2001](#) on page 336

Subdesign Attributes

- [arch_filename](#) on page 337
- [embedded_script](#) on page 338
- [entity_filename](#) on page 338
- [hdl_all_filelist](#) on page 339
- [hdl_config_name](#) on page 340
- [hdl_filelist](#) on page 341
- [hdl_user_name](#) on page 342
- [hdl_v2001](#) on page 342
- [protected](#) on page 343
- [write_vlog_empty_module_for_subdesign](#) on page 344

Support Attributes

- [hdl_v2001](#) on page 346

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

bit_blasted_port_style

`bit_blasted_port_style` *string*

Default: %s_%d

Read-write `root` attribute. Specifies the naming style to be used if mapped ports are bit blasted. The attribute value needs to contain elements %s and %d, in that order.

The following rules apply:

- An port name always starts with the base name (represented by %s)
- The suffix is appended to the base name to form the port name, according to the format specified in the attribute value.
- %d represents the bit information

Example

```
rc:/> set_attr bit_blasted_port_style %d_%s /  
Error   : The naming style is not legal. [TUI-261] [set_attribute]  
        : The given value was '%d_%s'  
        : The string must contain substrings '%s' and '%d' in that order.  
rc:/> set_attribute bit_blasted_port_style %s\[%d\]  
Setting attribute of root '/': 'bit_blasted_port_style' = %s[%d]
```

Related Information

[Handling Bit Blasted Port Styles in Using Encounter RTL Compiler](#)

Affects these commands:

[edit_netlist bitblast all_ports](#)

[write_hdl](#)

ccd_executable

`ccd_executable path`

Read-write root attribute. Specifies the Encounter® Conformal® Constraint Designer (CCD) executable that should be used for the `generate_constraints` and `validate_constraints` commands.

Example

The following example takes the CCD executable from the specified directory:

```
rc:/> set_attribute ccd_executable /path/conformal/ccd07.10/bin/ccd /
```

Related Information

Affects these commands:

[generate_constraints](#)

[validate_constraints](#)

command_log

`command_log string`

Default: `rc.cmd`

Read-write root attribute. Specifies the output file to which to write all commands executed in the session.

detailed_sdc_messages

`detailed_sdc_messages {false | true}`

Default: `false`

Read-write root attribute. Controls the preciseness of the line numbers reported in the messages for failing SDC commands when reading in an SDC file.

Set this attribute to `true` to get an exact pointer to the line number of the failing SDC command during `read_sdc`. This is especially useful when the SDC command is embedded within control structures (if, foreach, while etc) or within a TCL procedure.

By default, the warning and error messages simply point to the line number of the *control* structure (if,foreach,while) enclosing the SDC command or to the line number of the procedure call.

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

Examples

For the examples below, the lines in the sample SDC file (test.sdc) below are numbered:

```
1.set temp 1
2.if {$temp} {
3.
4.
5.
6.
7.
8.
9.set_false_path -from [get_cells ffff]
10.
11.
12.
13.}
```

- The following messages are given with the default behavior:

```
rc:/> read_sdc test.sdc
```

```
Warning : Could not find requested search value. [SDC-208] [get_cells]
          : The 'get_cells' command on line '2' of the SDC file 'test.sdc' cannot fi
nd any cells named 'ffff'.
          : Use the 'cd' and 'ls' commands to browse the virtual directories to find
the object because the specified name and/or location does not exist.
Error   : A required object parameter could not be found. [TUI-61] [parse_options]
          : An object of type 'clock|port|instance|pin' named '' could not be found.
          : Check to make sure that the object exists and is of the correct type.
The 'what_is' command can be used to determine the type of an object.
```

```
Usage: set_false_path [-rise] [-fall] [-from <clock|port|instance|pin>+]
                     [-rise_from <clock|port|instance|pin>+]
                     [-fall_from <clock|port|instance|pin>+]
                     [-to <clock|port|instance|pin>+] [-rise_to <clock|port|instance|pin>+]
                     [-fall_to <clock|port|instance|pin>+] ...
...
Error   : Could not interpret SDC command. [SDC-202] [read_sdc]
          : The 'read_sdc' command encountered an error while processing this command
on line '2' of the SDC file 'test.sdc': if {$temp} {
```

```
set_false_path -from [get_cells ffff]
```

```
.
          : The 'read_sdc' command encountered a problem while trying to evaluate an
SDC command. This SDC command will be added to the Tcl variable
$::dc::sdc_failed_commands.
```

```
Statistics for commands executed by read_sdc:
```

"get_cells"	- successful	0	, failed	1	(runtime 0.00)
"set_false_path"	- successful	0	, failed	1	(runtime 0.00)

```
Warning : Total failed commands during read_sdc are 2
```

```
Warning : One or more commands failed when these constraints were applied.[SDC-209]
          : The 'read_sdc' command encountered a problem while processing commands.
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

```
: You can examine the failed commands or save them to a file by querying  
the Tcl variable $::dc::sdc_failed_commands.  
Total runtime 0.0
```

- The following messages are given when you set detailed_sdc_messages to true:

```
rc:/> read_sdc test.sdc

-- Message on line '9' of the SDC file 'test.sdc'  
-- The command present in the file is : 'get_cells ffff'  
-- The command passed to the interpreter is : ':::dc:::get_cells ffff'  
-- Message type: Error  
Warning : Could not find requested search value. [SDC-208] [get_cells]  
: The 'get_cells' command on line '9' of the SDC file 'test.sdc' cannot  
find any cells named 'ffff'.  
-- End of messages for line '9'

-- Message on line '9' of the SDC file 'test.sdc'  
-- The command present in the file is : 'set_false_path -from [get_cells ffff]'  
-- The command passed to the interpreter is : ':::dc:::set_false_path -from {}'  
-- Message type: Error  
Error : A required object parameter could not be found. [TUI-61] [parse_options]  
: An object of type 'clock|port|instance|pin' named '' could not be found.

Usage: set_false_path [-rise] [-fall] [-from <clock|port|instance|pin>+]  
[-rise_from <clock|port|instance|pin>+]  
[-fall_from <clock|port|instance|pin>+]  
[-to <clock|port|instance|pin>+] [-rise_to <clock|port|instance|pin>+]  
[-fall_to <clock|port|instance|pin>+] ...  
...  
-- End of messages for line '9'

Error : Could not interpret SDC command. [SDC-202] [read_sdc]  
: The 'read_sdc' command encountered an error while processing this command  
from line '2' to line '13' of the SDC file 'test.sdc'  
: The command is:  
if {$temp} {  
  
set_false_path -from [get_cells ffff]  
  
}  
##### M1 - End of error Messages from line '2' to line '13' of test.sdc  
  
Statistics for commands executed by read_sdc:  
"get_cells"           - successful      0 , failed      1 (runtime 0.00)  
"set_false_path"       - successful      0 , failed      1 (runtime 0.00)  
Warning : Total failed commands during read_sdc are 2  
Warning : One or more commands failed when these constraints were applied. [SDC-209]  
: The 'read_sdc' command encountered a problem while processing commands.  
Total runtime 0.0
```

Related Information

Related command: [read_sdc](#)

encounter_executable

`encounter_executable path`

Default: default_search_order

Read-write [root](#) attribute. Specifies the Encounter® executable that should be used for the `synthesize -to_placed` command. It overrides the default search order, which first examines the `ENCOUNTER` environment variable, next the `PATH` environment variable, then the `CDS_SYNTH_ROOT` environment variable. Includes checking to assure that the specified executable is reachable.

Example

The following example takes the Encounter executable from the specified directory:

```
rc:/> set_attribute encounter_executable /path/soce/bin/encounter /
```

Related Information

[Setup for Physical Flow in Design with Physical in Encounter RTL Compiler](#)

Affects these commands: [synthesize -to_placed](#)

Related attribute: [phys_checkout_encounter_license](#) on page 379

error_on_lib_lef_pin_inconsistency

`error_on_lib_lef_pin_inconsistency {false | true}`

Default: false

Read-write [root](#) attribute. If enabled, the tool will issue an error message if a library pin of a logical cell is not found in the corresponding physical cell.

Note: You must set this attribute before reading the libraries.

Related Information

Affects this attribute: [library](#) on page 274

ets_executable

`ets_executable path`

Default: default_search_order

Read-write root attribute. Specifies the Encounter® Timing System executable that should be used for the validate_timing command. If this attribute is not set, it examines the PATH environment variable. If no executable is found, the command issues an error message.

Related Information

Affects these commands: [validate_timing](#)

group_generate_portname_from_netname

`group_generate_portname_from_netname {false | true}`

Default: false

Read-write root attribute. Determines whether the port names of a grouped module should be generated based on the names of the nets connected to these ports.

By default, the port name might be generated arbitrarily. Usually the port name of a new group is generated based on the names of the instance and pin to which the port is connected.

Example

Consider the following RTL.

```
module test (en,in, out, clk);
    input [4:0] in;
    input clk,en;
    output [4:0] out;
    reg [4:0] out;
    always @(posedge clk)
        begin
            if(en)
                out <= in;
            else
                out <= out;
        end
endmodule
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

Now consider the following script:

```
set_attr group_generate_portname_from_netname true
set_attr library tutorial.lbr
set_attr lp_insert_clock_gating true /
read_hdl test.v
elab
synth -to_map
cd /designs/test/instances_seq
edit_netlist group -group_name my_module [list out_reg[2] out_reg[3] ]
write_hdl
```

In the following netlist, the port names of module `my_module` are based on the net names (`in`, `rc_gclk`, and `out`) connected to these ports.

```
module RC(CG_MOD_AUTO_test(ck_in, enable, test, ck_out);
...
endmodule

module my_module(in, rc_gclk, out);
    input [1:0] in;
    input rc_gclk;
    output [1:0] out;
    wire [1:0] in;
    wire rc_gclk;
    wire [1:0] out;
    fflopd \out_reg[2] (.CK (rc_gclk), .D (in[0]), .Q (out[0]));
    fflopd \out_reg[3] (.CK (rc_gclk), .D (in[1]), .Q (out[1]));
endmodule

module test(en, in, out, clk);
    input en, clk;
    input [4:0] in;
    output [4:0] out;
    wire en, clk;
    wire [4:0] in;
    wire [4:0] out;
    wire rc_gclk;
    RC(CG_MOD_AUTO_test RC(CG_HIER_INST1(.ck_in (clk), .enable (en),
        .test (1'b0), .ck_out (rc_gclk));
    my_module my_modul ei(.in (in[3:2]), .rc_gclk (rc_gclk), .out (out[3:2]));
    fflopd \out_reg[4] (.CK (rc_gclk), .D (in[4]), .Q (out[4]));
    fflopd \out_reg[1] (.CK (rc_gclk), .D (in[1]), .Q (out[1]));
    fflopd \out_reg[0] (.CK (rc_gclk), .D (in[0]), .Q (out[0]));
endmodule
```

If you use the default value of the attribute, the port names may be arbitrary. In this example, the clock port of the group is based on the names of the instance and the pin that the port is connected to.

```
module my_module(in, RC(CG_HIER_INST1_ck_out, out);
    input [1:0] in;
    input RC(CG_HIER_INST1_ck_out;
    output [1:0] out;
    wire [1:0] in;
    wire RC(CG_HIER_INST1_ck_out;
    wire [1:0] out;
    fflopd \out_reg[2] (.CK (RC(CG_HIER_INST1_ck_out), .D (in[0]), .Q (out[0]));
    fflopd \out_reg[3] (.CK (RC(CG_HIER_INST1_ck_out), .D (in[1]), .Q (out[1]));
endmodule
```

Related information

Affects this command: [edit netlist group](#)

group_instance_suffix

`group_instance_suffix string`

Default: i

Read-write root attribute. Controls the suffix added to the instance name of a new group hierarchy.

Example

The following command prevents a suffix to be added.

```
set_attribute group_instance_suffix "" /
```

Related information

Affects this command: [edit netlist group](#)

hdl_allow inout const_port_connect

`hdl_allow inout const_port_connect {false | true}`

Default: false

Read-write root attribute. When set to false, issues an error message if an output or inout pin of an instantiated submodule is connected to a constant value.

Note: This attribute is supported only in the RTL flow.

Example

Consider the following RTL:

```
module top (enable,CLK, CLK1, CLK2, in1, in2, out1, out2, inout_1, inout_2);
input     in1, in2 ;
input enable ;
inout inout_1, inout_2;
input CLK, CLK1, CLK2 ;
output    out1, out2 ;

test test1(.enable(enable),.CLK(CLK),.CLK1(CLK1), .CLK2(CLK2), .in1(in1),
.in2(in2), .out1(out1), .out2(out2), .inout_1(1'b1));
endmodule
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

```
module test (enable, CLK, CLK1, CLK2, in1, in2, out1, out2, inout_1);
  input    in1, in2 ;
  input enable ;
  inout inout_1;
  input CLK, CLK1, CLK2 ;
  output   out1, out2 ;
  reg      out1, out2 ;
  wire CLK3;
  wire en;

  assign en = enable ? in1&in2&enable : in1|in2|enable ;
  assign CLK3 = en ? CLK1 : CLK2 ;
  always @ ( posedge CLK3 )
    out1 <= en ? in1 & in2 : out1 ;
  always @ ( posedge CLK )
    out2 <= en ? in1 | in2 : out2 ;
endmodule
```

Now consider the following script:

```
set_attr library typical.lib
set_attribute hdl_allow inout const_port_connect false /
read_hdl test.v
elab top
```

When you use the default value of the attribute (`false`) as in the script above, you will get the following error:

```
Error    : Cannot associate a constant to an output or inout port. [CDFG-413]
[elaborate]
          : Port 'inout_1' of module 'test' in file 'test.v' on line 10.
          : Use 'set_attribute hdl_allow inout const_port_connect true /' to allow
connection of a constant to an inout port.
```

Related Information

[HDL-Related Attributes in HDL Modeling in Encounter RTL Compiler](#)

hdl_allow_instance_name_conflict

`hdl_allow_instance_name_conflict {false | true}`

Default: `false`

Read-write root attribute. Controls whether the input Verilog file can have a variable (reg or wire) and an instance with the same name. By default, this results in an error. Set this attribute to `true` to allow this situation.

Related Information

Affects these commands:

[read_hdl](#)

[read_netlist](#)

hdl_ignore_pragma_names

`hdl_ignore_pragma_names string`

Default: coverage

Read-write root attribute. Specifies a Tcl list of one or more pragma names that must be ignored when reading in Verilog or VHDL.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

hdl_keep_first_module_definition

`hdl_keep_first_module_definition {false | true}`

Default: false

Read-write root attribute. Controls whether the first seen or last seen definition for a given module name is used. When encountering multiple definitions for the same module name, RTL Compiler keeps by default the last seen definition while ignoring other definitions for the same module name. To force RTL Compiler to retain the first seen (instead of the last seen) definition for a given module name, set the `hdl_keep_first_module_definition` attribute to true.

Related Information

Affects these commands: [read_hdl](#)

hdl_language

`hdl_language {v2001 | v1995 | vhdl | sv}`

Default: v2001

Read-write root attribute. Specifies the default hdl language mode assumed when you run the `read_hdl` command without specifying the language mode.

Note: This attribute is supported only in the RTL flow.

Related Information

[HDL-Related Attributes](#) in *HDL Modeling in Encounter RTL Compiler*

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_link_from_any_lib

`hdl_link_from_any_lib {true | false}`

Default: true

Read-write [root](#) attribute. When set to `true`, links to a unique definition from any library. This is useful when a module is instantiated and there is no specification in the instantiation as to which definition it should be linked to. If there are multiple definitions, then the instantiated module is not linked.

hdl_max_memory_address_range

`hdl_max_memory_address_range integer`

Default: 16384

Read-write [root](#) attribute. Specifies the maximum range of indexed memory access (read or write) that can be elaborated and synthesized.

Related Information

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_nc_compatible_module_linking

`hdl_nc_compatible_module_linking {true | false}`

Default: true

Read-write [root](#) attribute. When set to `true`, implements Native Compiler (NC) compatible binding rules for linking module or entity instantiations with corresponding definitions.

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

To link a module instantiation and choose a definition that should be linked, the following rules describe the order in which a search is applied to find a module to link.

- For a Verilog instance, searches:
 - The entity in the work default library.
 - All libraries if the `hdl_nc_compatible_module_linking` attribute is set to true.
- For a VHDL instance, searches:
 - The specified library to find a module definition if the library name is specified as part of the instantiation, for example:

```
instance_name: entity lib_name.entity_name
```

If the library is not found, then the search is not attempted across any other library and a message is issued stating that the specified binding was not found.
 - The library in which the component was compiled.
 - A design unit made visible with a USE clause given to the architecture instantiating the component.
 - A design unit made visible with a USE clause given to the entity of the architecture instantiating the component.
 - A design unit in RTL Compiler's default library.
 - A design unit made visible with a LIBRARY clause given to the architecture instantiating the component.
 - A design unit made visible with a LIBRARY clause given to the entity of the architecture instantiating the component.

With these rules in place, if there is more than one match found for a particular search rule, then the following describes the search order:

- For a Verilog instance, searches only for a case-sensitive match.
- For a VHDL instance, searches:
 - A VHDL specific binding. A matching VHDL module that was read in first will be picked up for linking.
 - A verilog module whose name and case matches that used in the component declaration.
 - A verilog module with a matching name that is all lowercase.
 - A verilog module with a matching name in any case.

When multiple module definitions exist for an instantiation, setting the attribute to `false` allows the tool to search in all libraries for a module definition. The module resolution for the instance may or may not be successful depending on finding the exact match. If there are multiple matches, linking resolution may not happen. When set to `false`, the tool does not follow any particular set of rules when performing linking resolution.



It is recommended to set the attribute to `true`, especially when multiple definitions exist for the same module.

Related Information

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_primitive_input_multibit

`hdl_primitive_input_multibit {false | true}`

Default: `false`

Read-write `root` attribute. Controls whether a Verilog primitive gate instantiated in RTL can have a multibit output pin connected to the primitive. For example, gates like NOT, bufif0, bufif1, notif0, notif1, and tran expect a single bit output. However, sometimes multibit outputs are connected in the RTL to these primitives.

Set this attribute to `true` to allow the LSB bit of a multibit output to be connected to the input of these primitives. By default, RTL with multibit signals connected to the input would not be allowed.

Related Information

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_report_case_info

`hdl_report_case_info {false | true}`

Default: false

Read-write root attribute. When set to `true`, reports information about how the `elaborate` command infers parallel-case or full-case for every “multi-way decision statement” (like a `case` statement). The reporting is module-by-module, with one report for each module.

The reports can help users understand whether both:

- RTL Compiler honored the `full_case` or `parallel_case` pragma
- RTL Compiler automatically inferred a full or parallel case

Related Information

[hdl_report_case_info in HDL Modeling in Encounter RTL Compiler](#)

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_search_path

`hdl_search_path Tcl_list`

Default: { . }

Read-write root attribute. Specifies a list of UNIX directories that RTL Compiler should search for files associated with the `read_hdl` command. The behavior is similar to the search path in UNIX.

In Verilog, this attribute directs the search of Verilog files specified with the `read_hdl` command and `include files specified in Verilog code.

In VHDL, this attribute directs the search of VHDL files specified with the `read_hdl` command.

Note: The “~” is supported.

This attribute is supported in the RTL flow and the structural flow.

Example

```
set_attr hdl_search_path { .../src .../incl }
```

Related Information

[Specifying HDL Library Search Paths](#) in *Using Encounter RTL Compiler*

Affects these commands: [elaborate](#)
[read_hdl](#)
[read_netlist](#)

hdl_track_filename_row_col

`hdl_track_filename_row_col {false | true}`

Default: false

Read-write [root](#) attribute. Enables or disables file, row, and column information tracking. When set to false, all the file, row, and column information is deleted. Set this attribute to true to enable file, row, column information before using the elaborate command.

This attribute enables RTL Compiler to keep track of filenames, line numbers, and column numbers for all instances before optimization, and use this information in selected reports and messages.

Enabling this attribute can have an impact on the runtime.

Note: This attribute is supported in the RTL flow and the structural flow.

Related Information

[Running the DFT Rule Checker](#) in *Design for Test in Encounter RTL Compiler*

Affects these commands: [elaborate](#)
[read_hdl](#)
[read_netlist -netlist](#)
[read_netlist](#)

hdl_use_current_dir_before_hdl_search_path

```
hdl_use_current_dir_before_hdl_search_path {false | true}
```

Default: false

Read-write root attribute. Controls the order of the directories from where the include files must be read. By default, the tool uses the directories specified with the `hdl_search_path` attribute to search for include files. When this attribute is set to `true`, the tool uses the directory of the source file (containing the include file) before the `hdl search path`.

Examples

Consider the following directory structure:

```
TOP.v
define.v
D1/
    A.v
    define.v
D2/
    B.v
    define.v
D3/
    C.v
    define.v
```

Files `A.v`, `B.v`, and `C.v`, each include a `define.v` file. The content of the `define.v` files in the three directories is different.

Assume you run the following script

```
set_attr library tutorial.lib /
set_attr hdl_search_path {./D1 ./D2 ./D3} /
read_hdl A.v B.v C.v TOP.v
```

- Using the default setting (`false`) of the attribute.

The tool will use the directories specified with the `hdl_search_path` attribute. In this case, the `define.v` file will be picked from the `./D1` directory.

- Using the `true` setting of the `hdl_use_current_dir_before_hdl_search_path` attribute.

In this case, you will see that the tool uses the `define.v` file from the directory of the source file (that contains the include file).

```
Reading Verilog file './D1/A.v'
Reading Verilog file './D1/define.v'
Reading Verilog file './D2/B.v'
Reading Verilog file './D2/define.v'
Reading Verilog file './D3/C.v'
Reading Verilog file './D3/define.v'
Reading Verilog file './TOP.v'
```

Related Information

Affects these commands: [elaborate](#)
[read_hdl](#)
[read_netlist -netlist](#)

hdl_verilogDefines

`hdl_verilogDefines macro_definition_list`

Default: SYNTHESIS

Read-write `root` attribute. Specifies a list of global macro definitions that must be applied to each subsequent `read_hdl` (except with `-vhdl`) and `read_netlist` command. This includes calls to `read_hdl` made during a call to `elaborate -libpath`.

Each macro definition has the form `id` or `id=expr`.

Example

The following commands are equivalent:

- `set_attr hdl_verilogDefines "A B=4 C"`
`read_hdl ...`
- `read_hdl -define "A B=4 C" ...`
- `read_hdl -define A -define B=4 -define C ...`

Related Information

Affects these commands: [elaborate -libpath](#)
[read_hdl](#)
[read_netlist](#)

hdl_vhdl_assign_width_mismatch

```
hdl_vhdl_assign_width_mismatch {false | true}
```

Default: false

Read-write root attribute. Controls whether to allow an assignment in VHDL when the left-hand side (lhs) and right-hand side (rhs) have the same array type but mismatching width. When set to true, the rhs gets truncated or extended to the width of the lhs. By default, VHDL does not allow such assignments.

Example

Consider the following RTL. Ports `Mult_A` and `Z` have the same array type but have incompatible bit widths.

```
library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
entity mmnumeric1 is
    generic (wmA : integer := 2;
             wZ   : integer := 5);
    port ( Mult_A  : in signed (wmA-1 downto 0);
           Z       : out signed (wZ-1 downto 0) );
end;
architecture rtl of mmnumeric1 is
begin
    Z <= Mult_A;
end;
```

With the default setting of `hdl_vhdl_assign_width_mismatch` (false), RTL Compiler issues error message CDFG-283 during elaboration.

When you set `hdl_vhdl_assign_width_mismatch` to true, the RTL will successfully elaborate, but RTL Compiler issues warning message CDFG-239.

Related Information

[VHDL-Specific Attributes in HDL Modeling in Encounter RTL Compiler](#)

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_vhdl_case

```
hdl_vhdl_case {original | lower | upper}
```

Default: original

Read-write root attribute. Stores VHDL identifiers and operators in lowercase, uppercase, or the case given in the source file.

Related Information

[VHDL-Specific Attributes in HDL Modeling in Encounter RTL Compiler](#)

[Using Case Sensitivity in Verilog/VHDL Mixed-Language Designs in Using Encounter RTL Compiler](#)

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_vhdl_environment

```
hdl_vhdl_environment {common | synergy}
```

Default: common

Read-write root attribute. Specifies the selection of the predefined arithmetic libraries.

Related Information

[VHDL-Specific Attributes in HDL Modeling in Encounter RTL Compiler](#)

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_vhdl_lrm_compliance

```
hdl_vhdl_lrm_compliance {false | true}
```

Default: false

Read-write root attribute. When set to true, the `read_hdl` command enforces a strict interpretation of the VHDL LRM. This variable lets you verify that your VHDL code is compliant with the LRM, such that it is likely to work on other VHDL tools.

When set to false the following features are allowed:

- Treats a concatenation as a locally static expression, which allows constructs such as the following:

```
case expr is
    when "001" & '1' => ...
```

- Initializes an interface object with a function call.

- Lets you use name *X* in the definition of a different *X*:

```
constant c : integer := 3;
function f (c : integer := c) is ...
```

- In VHDL-1987, treats a function call with globally static arguments as a globally static expression. For instance:

```
function f(x : integer) return integer;
...
generic (y : integer := f(3));
```

Related Information

[VHDL-Specific Attributes in HDL Modeling in Encounter RTL Compiler](#)

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_vhdl_preferred_architecture

```
hdl_vhdl_preferred_architecture string
```

Read-write root attribute. Sets the name of preferred architecture to use with an entity when there are multiple architectures.

Related Information

VHDL-Specific Attributes in *HDL Modeling in Encounter RTL Compiler*

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_vhdl_range_opto

`hdl_vhdl_range_opto {true | false}`

Default: true

Read-write root attribute. Controls the optimization of the widths of arithmetic operators (+, *, -) that have ranged integer arguments in VHDL.

Example

Consider the following RTL:

```
entity rngopt is port(
    in1 : in integer range 0 to 12;
    in2 : in integer range 0 to 3;
    output : out integer);
end;

architecture rtl of rngopt is
begin
    output <= in1 + in2;
end;
```

- When you use the default setting (true), RTL Compiler infers an unsigned adder with an output width of 4 bits during elaboration, because the result will be in the range of 0 to 15 (12+3).
- When you set `hdl_vhdl_range_opto` to false, RTL Compiler infers an unsigned adder with an output width of 5 bits during elaboration.

Related Information

VHDL-Specific Attributes in *HDL Modeling in Encounter RTL Compiler*

Affects this command: [elaborate](#)

hdl_vhdl_read_version

```
hdl_vhdl_read_version {1993 | 1987 | 2008}
```

Default: 1993

Read-write root attribute. Specifies the VHDL version to be used when reading VHDL designs. If you change the `hdl_vhdl_read_version` read version at any time, RTL Compiler automatically re-maps the IEEE and STD VHDL libraries to the correct directory in the software release. In this way the correct versions of standard packages in these libraries are picked up.

Related Information

VHDL-Specific Attributes in *HDL Modeling in Encounter RTL Compiler*

Affects these commands: elaborate
 read_hdl

hdl_zero_replicate_is_null

```
hdl_zero_replicate_is_null {true | false}
```

Default: true

Read-write root attribute. Controls whether a zero-length replication is treated as a null expression having zero width. This matches the behavior of Encounter® Conformal® Equivalence Checking.

Example

Consider the following expression.

```
{ {0{8'hff}}, 3'b111 }
```

If the attribute is set to `false`, this expression evaluates to `4'b0111`.

If the attribute is set to `true`, the expression evaluates to `3'b111`.

Related Information

Affects these commands: elaborate
 read_hdl

input_assert_one_coldPragma

`input_assert_one_coldPragma string`

Default: assert_one_cold one_cold

Read-write root attribute. Carries a Tcl list of one or more names, each being treated as a one_cold pragma when reading in Verilog or VHDL.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

input_assert_one_hotPragma

`input_assert_one_hotPragma string`

Default: assert_one_hot one_hot

Read-write root attribute. Carries a Tcl list of one or more names, each being treated as a one_hot pragma when reading in Verilog or VHDL.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

input_asynchro_reset_blkPragma

`input_asynchro_reset_blkPragma string`

Default: async_set_reset_local asynchro_reset_blk

Read-write root attribute. Carries a Tcl list of one or more names, each being treated as a asynchro_reset_blk pragma when reading in Verilog or VHDL.

Specifies the equivalent used in input pragmas for the RTL Compiler asynchro_reset_blk pragma.

```
//cadence asynchro_reset_blk <namedBlock> [{rst1 rst2}]
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

Any `if` conditions in the specified block containing `rst1` and `rst2` are considered as asynchronous reset condition inside the asynchronous `if-then-else` decoding code.

Note: This attribute is supported only in the RTL flow.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

input_asynchro_reset_pragma

`input_asynchro_reset_pragma string`

Default: `async_set_reset asynchro_reset`

Read-write `root` attribute. Carries a Tcl list of one or more names, each being treated as a `asynchro_reset` pragma when reading in Verilog or VHDL.

Specifies the equivalent used in input pragmas for the RTL Compiler `asynchro_reset` pragma.

`//cadence asynchro_reset [{rst1 rst2}]`

Any `if` conditions containing `rst1` and `rst2` are considered as asynchronous reset conditions inside the asynchronous `if-then-else` decoding code.

Note: This attribute is supported only in the RTL flow.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

input_case_coverPragma

`input_case_coverPragma string`

Default: full_case

Read-write `root` attribute. Carries a Tcl list of one or more names, each being treated as a `full_case` pragma when reading in Verilog or VHDL.

Specifies the equivalent used in input pragmas for the RTL Compiler `case_logic cover` pragma.

Note: This attribute is supported only in the RTL flow.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

input_case_decodePragma

`input_case_decodePragma string`

Default: parallel_case

Read-write `root` attribute. Carries a Tcl list of one or more names, each being treated as a `parallel_case` pragma when reading in Verilog or VHDL.

Specifies the equivalent used in input pragmas for the RTL Compiler `case_logic no_priority` pragma.

Note: This attribute is supported only in the RTL flow.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

input_map_to_mux_pragma

`input_map_to_mux_pragma string`

Default: `map_to_mux infer_mux`

Read-write `root` attribute. Carries a Tcl list of one or more names, each being treated as a `map_to_mux` pragma when reading in Verilog or VHDL.

Specifies the equivalent used in input pragmas for the RTL Compiler `map_to_mux` pragma.

Note: This attribute is supported only in the RTL flow.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

Related attribute [map_to_mux](#) on page 848

input_pragma_keyword

`input_pragma_keyword string`

Default: `cadence synopsys ambit pragma synthesis`

Read-write `root` attribute. Carries a Tcl list of one or more names, each being treated as the keyword indicating a pragma when reading in Verilog or VHDL.

A pragma is a comment whose first word is a pragma keyword specified in this attribute.

You cannot remove the `cadence` keyword from the list.

Note: This attribute is supported in the RTL flow and the structural flow.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects these commands: [read_hdl](#)

[read_netlist](#)

input_synchro_enable_blk_pragma

`input_synchro_enable_blk_pragma string`

Default: `synchro_enable_blk`

Read-write `root` attribute. Carries a Tcl list of one or more names, each being treated as a `synchro_enable_blk` pragma when reading in Verilog or VHDL.

Specifies the `synchro_reset_blk` pragma name in input pragmas.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

input_synchro_enable_pragma

`input_synchro_enable_pragma string`

Default: `synchro_enable`

Read-write `root` attribute. Carries a Tcl list of one or more names, each being treated as a `synchro_enable` pragma when reading in Verilog or VHDL.

Specifies the synchro enable command in input pragmas.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

[read_netlist](#)

input_synchro_reset_blk_pragma

`input_synchro_reset_blk_pragma string`

Default: `sync_set_reset_local synchro_reset_blk`

Read-write `root` attribute. Carries a Tcl list of one or more names, each being treated as a `synchro_reset_blk` pragma when reading in Verilog or VHDL.

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

Specifies the equivalent used in input pragmas for the RTL Compiler sync_set_reset_local pragma.

```
//cadence sync_set_reset_local ,namedBlock "comma-separated_list_of_signals"
```

Note: This attribute is supported only in the RTL flow.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

input_synchro_reset_pragma

```
input_synchro_reset_pragma string
```

Default: sync_set_reset "comma_separated_list_of_signals"

Read-write [root](#) attribute. Carries a Tcl list of one or more names, each being treated as a synchro_reset_blk pragma when reading in Verilog or VHDL.

Specifies the equivalent used in input pragmas for the RTL Compiler synchro_reset pragma.

```
//cadence sync_set_reset [{rst1 rst2}]
```

Any if conditions containing `rst1` and `rst2` are considered as synchronous reset conditions inside the synchronous if-then-else decoding code.

Note: This attribute is supported only in the RTL flow.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)

inst_prefix

`inst_prefix string`

Default: g

Read-write root attribute. Specifies the prefix to be used together with an integer to generate new instance names. For example, g4044.

Each gate is named by constructing a string from the instance prefix and an integer that allows RTL Compiler to construct unique instance names. This attribute lets you change the naming scheme for gates.

Related Information

Affects these commands:

[elaborate](#)

[synthesize](#)

lbr_respect_async_controls_priority

`lbr_respect_async_controls_priority {true | false}`

Default: true

Read-write root attribute. Instructs the library parser to consider the asynchronous control pins priority description of sequential libcells from the liberty files. This enables the implicit implementation of the priority logic by the sequential library cells during mapping. Set this attribute to `false` before reading the library if you want the priority logic to be implemented explicitly by combinational gates.

Related Information

Affects this attribute

[library](#) on page 274

lec_executable

`lec_executable path`

Read-write root attribute. Specifies the Encounter® Conformal® Low Power (CLP) executable that should be used for the `verify_power_structure` and `check_cpf` commands.

Example

The following example takes the CLP executable from the specified directory:

```
rc:/> set_attribute lec_executable /path/conformal/lec07.10/bin/lec /
```

Related Information

[Specifying the Conformal Executable](#) in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects these commands: [check_cpf](#)

[verify_power_structure](#)

lef_add_logical_pins

```
lef_add_logical_pins {true|false}
```

Default: true

Read-write root attribute. Controls the addition of LEF logical pins (pins other than ANALOG, GROUND or POWER) that are not part of the libcell definition in the Liberty file. By default these pins are added to the libcell definition.

lef_library

```
lef_library lef_library
```

Read-write root attribute. Specifies the LEF libraries for technology mapping. Specifically, RTL Compiler will use the wire resistance, capacitance, area, and site size information in the LEF library. The capacitance table file can contain the same information but in all cases, RTL Compiler will use the latest specification. You must specify the LEF libraries before the capacitance table file for the best synthesis results.

Cells that are found in the timing library but not in the LEF library will be marked as avoid. That is, the avoid attribute will be set to true. on those cells that are in the .lib file but not in the LEF file.

When you read in LEF libraries, the cell area defined in the LEF libraries will be used instead of the cell area specified in the timing library (.lib). The timing library area will be used if the physical libraries do not contain any cell definitions.

Note: Reading in LEF libraries, sets the `interconnect_mode` root attribute to ple.

RTL Compiler supports LEF 5.3 and above.

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

Note: You must import all LEF libraries, not just the technology LEF. You must specify the technology LEF file first.

Example

The following example specifies two LEF libraries, `tech.lef` and `cell.lef`:

```
rc:/> set_attribute lef_library {tech.lef cell.lef}
```

The following example differs from the above example: it replaces the existing LEF file because it specifies the files separately with two `set_attribute` commands as opposed to a Tcl list with one `set_attribute` command.

```
rc:/> set_attribute lef_library tech.lef
rc:/> set_attribute lef_library cell.lef
```

Related Information

[Reading the LEF Libraries in Design with Physical in Encounter RTL Compiler](#)

Related attributes:

- [avoid](#) on page 190
- [cap_table_file](#) on page 364
- [interconnect_mode](#) on page 377
- [scale_of_cap_per_unit_length](#) on page 384
- [scale_of_res_per_unit_length](#) on page 385

lib_search_path

`lib_search_path` *Tcl_list*

Default: { . }

Read-write [root](#) attribute. Specifies a list of UNIX directories that RTL Compiler should search to locate the technology libraries and LEF libraries.

Note: The “~” is supported.

Related Information

[Specifying Explicit Search Paths in Using Encounter RTL Compiler](#)

Affects these attributes:

- [library](#) on page 274
- [lef_library](#) on page 272

library

```
library {{lib [lib]...} [{lib [lib]}]...}
```

Read-write root attribute. Sets the target library for technology mapping.

You must specify this attribute on root if you have a single voltage design.

You can specify a single library or a Tcl list of library lists. Each library list is also a Tcl list. Each library must be found in the library search path (specified through the `lib_search_path` attribute). The first library in each list is considered the master library to which the content of the other libraries in that list is appended.

The `library` attribute also supports libraries that have been compressed with GNU Zip (.gz extension).

Note: The information in the appended libraries overwrites the corresponding information in the master library. However, RTL Compiler fails on loading the libraries if the delay models in the appended libraries differ from the delay models in the master library.

Examples

- To read independent libraries, type the following command:

```
set_attribute library {a.lib b.lib c.lib x.lbr y.lib} /
```

This command creates five libraries in the `/libraries` directory.

- To load one library and append some others, type the following command:

```
set_attribute library {{a.lib b.lib c.lib x.lbr y.lib}} /
```

This command loads the first library and appends the contents of the next libraries to the first one. This command creates one library (`a.lib`) in the `/libraries` directory.

- To load several libraries and append some, type the following command:

```
set_attribute library {{a.lib b.lib} {c.lib} {x.lbr y.lib}}
```

This command loads `a.lib`, appends `b.lib` to `a.lib`, then loads `c.lib`, followed by `x.lbr` and finally appends `y.lib` to `x.lbr`. This command creates three libraries in the `/libraries` directory: `a.lib`, `c.lib`, and `x.lbr`.

- To load a GZIP compressed library, type the following command:

```
set_attribute library big_15_lv.lib.gz
```

Related Information

Setting the Target Technology Library in *Using Encounter RTL Compiler*

Affected by this attribute: [lib_search_path](#) on page 273

Related attribute: [supportAppendingLibs](#) on page 280

nc_protect_version

`nc_protect_version string`

Read-only [root](#) attribute. Indicates which release of the `ncprotect` utility is supported by RTL Compiler.

Currently, the release of the `ncprotect` utility supported by RTL Compiler is IUS 8.2-s015.

Note: The `ncprotect` utility is a utility of the Cadence® NC-Verilog Simulator and of the Cadence® NC-VHDL Simulator. Both simulators are part of the IUS release.

Related Information

IP Protection in *Using Encounter RTL Compiler*

path

`path string`

Default: `. /libraries/* /libraries/library_domains/*/*
/libraries/library_domains/* /designs/*
/designs/* /timing/clock_domains/*
/designs/* /dft/test_clock_domains/* /designs/* /modes
/designs/* /modes/* /clock_domains/* /mmmc_designs_spec/*
/designs/* /mmmc/*`

Read-write [root](#) attribute. Specifies the search paths for implicit finds. During an implicit find, RTL Compiler will perform an exhaustive search by recursively searching the specified paths.

Example

In the following example, RTL Compiler recursively searches the paths specified with the `path` attribute and sets a false path between all clock objects named `clk1` and `clk2`.

```
rc:/> dc::set_false_path -from clk1 -to clk2
```

In this case, RTL Compiler interprets the names `clk1` and `clk2` to be clock names because the inherent object search order of the SDC command `set_false_path` is clocks, ports, instances, pins. If there were no clocks named `clk1` or `clk2`, RTL Compiler would have interpreted the names to have been port names.

Related Information

[Specifying Implicit Search Paths](#) in *Using Encounter RTL Compiler*

rc_create_verification_directory

```
rc_create_verification_directory {true | false}
```

Default: true

Read-write root attribute. Controls whether to create the directory to which the verification files are written.

Related Information

Affects these commands:

[elaborate](#)

[synthesize](#)

rc_verification_directory

```
rc_verification_directory string
```

Default: fv/%s

Read-write root attribute. Specifies the format to use to name the directory to which the verification files are written.

The attribute value can contain an arbitrary string, zero or one %s representing the design name, and zero or one %d representing the number of the session.

- If the value contains neither a %s or a %d, each elaboration run writes the files in the same directory, overwriting the existing ones.

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

- If the value contains a %d, each elaboration run creates a new directory.
- If the value contains a %s but no %d, a new directory is created only if elaboration is given a different top design.

By default, the verification directory will be overwritten each time you run a new RTL Compiler session.

The following files will be written to this directory:

- dp_info.v—file with datapath information
- g1.v.gz—post-mapped intermediate netlist
- rtl_to_g1.do—do file for comparison between RTL and post-mapped intermediate netlist

Note: If you want to change the attribute setting, do so before you elaborate the design.

Examples

- Assume your design name is test, the name of the directory will be fv/test.
- The following setting can be used to prevent overwriting the directory for each new RTL Compiler session.

```
set_attribute rc_verification_directory fv/%s_%d /
```

Related Information

Affects these commands:

[elaborate](#)

[synthesize](#)

script_begin

`script_begin string`

Default: dc_script_begin dc_tcl_script_begin script_begin

Read-write root attribute. Specifies the keyword in the RTL netlist that indicates the beginning of a script included in the netlist.

Note: This attribute is supported only in the RTL flow.

Related Information

Supported Synopsys Pragmas in HDL Modeling in Encounter RTL Compiler

Affects this command: [read_hdl](#)

Affects this attribute: [hdl_auto_exec_sdc_scripts](#) on page 723

Related attribute: [script_end](#) on page 278

script_end

`script_end string`

Default: dc_script_end dc_tcl_script_end script_end

Read-write root attribute. Specifies the keyword in the RTL netlist that indicates the end of a script included in the netlist.

Related Information

Supported Synopsys Pragmas in HDL Modeling in Encounter RTL Compiler

Affects this command: [read_hdl](#)

Affects this attribute: [hdl_auto_exec_sdc_scripts](#) on page 723

Related attribute: [script_end](#) on page 278

script_search_path

`script_search_path string`

Default: \$CDN_SYNTH_ROOT/lib/etc

Read-write root attribute. Specifies the search path for script files. The “~” is supported.

Note: This attribute affects the `include` and the `source` commands.

Examples

- The following example sets the script path to /home/monicas/RC/scripts and then uses the `include` command to run the `example.g` script in the directory:

```
rc:/> set_attribute script_search_path /home/monicas/RC/scripts
Setting attribute of root /: 'script_search_path' =  /home/monicas/RC/scripts
rc:/> include example.g
```

- The following example sets the script search path to /home/monicas/RC/scripts and then tries to use the Tcl `source` command to load the `custom.tcl` file:

```
rc:/> set_attribute script_search_path /home/monicas/RC/scripts
Setting attribute of root /: 'script_search_path' =  /home/monicas/RC/scripts
rc:/> source custom.tcl
couldn't read file "custom.tcl": no such file or directory
```

The unsuccessful result of this command is because the attribute only affects the `include` command and not the Tcl `source` command. This intended behavior allows RTL Compiler to predictably honor the `source` command whenever it is embedded in the Tcl code that is loaded into, or specified within, the tool.

Related Information

[Specifying Explicit Search Paths in Using Encounter RTL Compiler](#)

Affects this command: [include](#)

supportAppendingLibs

```
supportAppendingLibs {false | true}
```

Default: false

Read-write root attribute. Controls whether you can append two or more libraries that are defined in different .lib files and that have the same library name and the same PVT conditions but contain different libcells.

Example

Assume you have libraries lib1.lib, lib2.lib and lib3.lib. The three library files have the same internal library name and PVT conditions but contain different libcells. To append the libcells of lib2.lib and lib3.lib to lib1.lib, enter the following:

```
set_attribute supportAppendingLibs true /  
set_attribute library {{lib1.lib lib2.lib lib3.lib }} /
```

These commands create one library (lib1.lib) in the /libraries directory.

Related Information

Affects this attribute: [library](#) on page 274

supportMultiSeqElements

```
supportMultiSeqElements {false | true}
```

Default: false

Read-write root attribute. Controls the conversion of a state retention cell with multiple sequential elements (for example, multiple ff or latch groups) into a single ff or latch function.

Any outgoing timing arcs on the power gating pin of the state-retention cell are enabled by default.

Note: You must set this attribute before you read the libraries.

Related Information

Affects this attribute: [library](#) on page 274

synthesis_off_command

`synthesis_off_command string`

Default: `translate_off synthesis_off`

Read-write `root` attribute. Specifies the pragma that is used to indicate the beginning of non-synthesizable constructs in the RTL source code or in the generated generic netlist.

Note: This attribute is supported in the RTL flow and the structural flow.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)
 [read_netlist](#)

Related attributes: [synthesis_on_command](#) on page 281
 [input_pragma_keyword](#) on page 268

synthesis_on_command

`synthesis_on_command string`

Default: `translate_on synthesis_on`

Read-write `root` attribute. Specifies the pragma that is used to indicate the end of non synthesizable constructs in the RTL source code or in the generated generic netlist.

Note: This attribute is supported in the RTL flow and the structural flow.

Related Information

[Specifying Synthesis Pragma Keywords in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [read_hdl](#)
 [read_netlist](#)

Related attributes: [synthesis_off_command](#) on page 281
 [input_pragma_keyword](#) on page 268

ungroup_separator

`ungroup_separator string`

Default: `_`

Read-write `root` attribute. Specifies the string used to separate the hierarchical names of instances that are ungrouped (flattened) after elaboration.

Example

The following example uses the slash (" / ") to separate ungrouped instances:

```
rc:/> set_attribute ungroup_separator /
```

use_power_ground_pin_from_lef

`use_power_ground_pin_from_lef {true | false}`

Default: `true`

Read-write `root` attribute. Specifies whether to ignore the inconsistency in the `use` definition of power and ground pins between .lib and LEF libraries. By default, the definition from LEF is used when an inconsistency is detected.

Note: When this attribute is enabled, you must read in the LEF libraries before reading the design information.

wccd_threshold_percentage

`wccd_threshold_percentage float`

Default: `0.800`

Read-write `root` attribute. Specifies the threshold percentage in the dofiles for the false path generation flow and the directed false path generation flow. This value instructs the Encounter® Conformal® Constraint Designer tool to consider only those paths with logic length that is greater than the maximum logic length of all paths in the design multiplied by the specified value. You can specify a real number between 0 and 1 for the threshold percentage.

Related Information

[False Path Generation](#) in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

[Directed False Path Generation](#) in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects this command	<u>generate constraints -dfpgen</u> <u>generate constraints -fpgen</u> <u>write do ccd generate -dfpgen</u> <u>write do ccd generate -dpigen</u>
----------------------	---

wcdc_clock_dom_comb_propagation

wcdc_clock_dom_comb_propagation {logic_nophase | exact_phase | either_phase | wire | logic_phase}

Default: logic_nophase

Read-write [root](#) attribute. Controls clock domain propagation through combinational elements in the Encounter® Conformal® Extended Checks software.

Example

The following command sets this attribute to exact_phase.

```
set_attribute wcdc_clock_dom_comb_propagation exact_phase /
```

This attribute setting causes the following clock domain rule:

```
set clock_domain rule -derived -exact_phase -extract buffer
```

Related Information

[Interfacing with Encounter Conformal Extended Checks](#) in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects this command	<u>write do verify cdc</u>
----------------------	--

wclp_lib_statetable

```
wclp_lib_statetable {true | false}
```

Default: true

Read-write root attribute. Affects the `read library -liberty` command in the generated dofile. If set to `true`, that command is equipped with a `-statetable` option. If set to `false`, that command will not have `-statetable`.

Related Information

[Enabling Support for Liberty State Tables in Interfacing between Encounter RTL Compiler and Encounter Conformal](#)

Affects this command

[check_cpf](#)
[verify_power_structure](#)
[write_do_clp](#)

wlec_add_noblock_box_retime_subdesign

```
wlec_add_noblock_box_retime_subdesign {true | false}
```

Default: true

Read-write root attribute. Controls whether to add the Conformal LEC command that excludes the specified retimed modules from the hierarchical dofile script generation. Specify this attribute before writing out the intermediate netlist.

The default value is the recommended value for retiming verification.

Example

Assume that your synthesis script contains the following commands:

```
set_attribute retime true {submod1 submod2}
set_attribute wlec_add_noblock_box_retime_subdesign true /
```

As a results, the following commands are added in the dofile generated by the `write_do_lec` command:

```
add noblock box submod1 -submodules -golden
add noblock box submod1 -submodules -revised
add noblock box submod2 -submodules -golden
add noblock box submod2 -submodules -revised
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

Related Information

[Interfacing with Encounter Conformal Logical Equivalence Checker in Interfacing between Encounter RTL Compiler and Encounter Conformal](#)

Affects this command [write do lec](#)

wlec_analyze_abort

wlec_analyze_abort {false | true}

Default: false

Read-write root attribute. Adds the Conformal LEC command that analyzes the abort points to the dofile. The command works on the abort points in the design and tries to resolve them by adding partition points, increasing the compare effort, and more. Specify this attribute before writing out the intermediate netlist.

If the attribute is set to `true` in a hierarchical compare, the following commands are added in the dofile:

```
write hier_compare dofile <dofile name> -noexact_pin_match -constraint -usage  
-replace -run_hier -prepend_string  
"remodel -seq_constant -repeat"  
run hier_compare <dofile name> -analyze_abort
```

If the attribute is set to `true` in a *flat* compare, the following commands are added in the dofile:

```
set system mode lec  
remodel -seq_constant -repeat  
map key points  
report mapped points  
report unmapped points -summary  
report unmapped points -extra -unreachable -notmapped  
add compared points -all  
compare  
usage  
// if seeing abort, please try 'analyze abort -compare'  
report compare data
```

Related Information

[Interfacing with Encounter Conformal Logical Equivalence Checker in Interfacing between Encounter RTL Compiler and Encounter Conformal](#)

Affects this command [write do lec](#)

Affected by this attribute: [wlec_auto_analyze](#) on page 286

wlec_analyze_setup

```
wlec_analyze_setup {false | true}
```

Default: false

Read-write root attribute. Adds the Conformal LEC command `analyze setup -verbose` to the dofile. The command analyzes setup related issues and tries to correct any key point mapping related issues before starting the comparison. Specify this attribute before writing out the intermediate netlist.

If the attribute is set to `true` in a *hierarchical* compare, the following commands will be added in the dofile:

```
write hier_compare dofile <dofile name> -noexact_pin_match -constraint \
    -usage -replace -run_hier -prepend_string
"remodel -seq_constant -repeat;analyze setup -verbose; map key points"
```

If the attribute is set to `true` in a *flat* compare, the following commands are added:

```
set system mode lec
remodel -seq_constant -repeat
analyze setup -verbose
map key points
```

Related Information

Interfacing with Encounter Conformal Logical Equivalence Checker in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects this command [write do lec](#)

Affected by this attribute: [wlec_auto_analyze](#) on page 286

wlec_auto_analyze

```
wlec_auto_analyze {true | false}
```

Default: true

Read-write root attribute. Adds the Conformal LEC command `set analyze option -auto` to the dofile. This command handles both setup related issues as well as abort points (if any). This attribute is a superset of the `wlec_analyze_abort` and `wlec_analyze_setup` attributes. Hence, if you keep this attribute at its default value of `true`, you do not need to specify neither `wlec_analyze_setup` nor `wlec_analyze_abort`.

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

By default the *hierarchical* compare will have the following commands in the dofile:

```
set flatten model -seq_constant -seq_constant_x_to 0
set flatten model -nodff_to_dlat_zero -nodff_to_dlat_feedback
set analyze option -auto
write hier_compare dofile <dofile name> -noexact_pin_match -constraint -usage \
    -replace -run_hier -prepend_string "remodel -seq_constant -repeat"
run hier_compare <dofile name>
```

By default, the *flat* compare will have the following commands in the dofile:

```
set flatten model -seq_constant -seq_constant_x_to 0
set flatten model -nodff_to_dlat_zero -nodff_to_dlat_feedback
set analyze option -auto
set system mode lec
remodel -seq_constant -repeat
```

If all of the `wlec_auto_analyze`, `wlec_analyze_setup`, and `wlec_analyze_abort` attributes are set to true, the `wlec_auto_analyze` will take precedence.

Related Information

[Interfacing with Encounter Conformal Logical Equivalence Checker in Interfacing between Encounter RTL Compiler and Encounter Conformal](#)

Affects this command [write do lec](#)

Affects these attributes: [wlec_analyze_abort](#) on page 285

[wlec_analyze_setup](#) on page 286

wlec_compare_threads

`wlec_compare_threads integer`

Default: 0

Read-write root attribute. Adds the Conformal LEC command `set compare` options with the `-threads` option to the dofile. The attribute value specifies the number of compare threads.

Related Information

[Interfacing with Encounter Conformal Logical Equivalence Checker in Interfacing between Encounter RTL Compiler and Encounter Conformal](#)

Affects this command [write do lec](#)

wlec_cut_point

```
wlec_cut_point {none| golden | revised | both}
```

Default: none

Read-write root attribute. Controls whether the `write_do_lec` command should locate all instances of `cdn_loop_breaker` gates in the current design and identify the hierarchical pins driving these `cdn_loop_breaker` gates.

If the attribute value is not an empty string, the `write_do_lec` command adds the following LEC command to the dofile for each of the identified hierarchical pins:

```
ADD CUT Point path_to_cut_point -pin -{golden|revised|both}
```

Related Information

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

Affects this command [write do lec](#)

wlec_hier_comp_threshold

```
wlec_hier_comp_threshold integer
```

Default: 50

Read-write root attribute. Specifies the minimum number of instances required in a module to perform a hierarchical comparison on that module with the Encounter® Conformal® Equivalence Checking tool.

Related Information

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

Affects this command [write do lec](#)

wlec_lib_statetable

wlec_lib_statetable {true | false}

Default: true

Read-write root attribute. Affects the `read library -liberty` command in the generated dofile. If set to `true`, that command is equipped with a `-statetable` option. If set to `false`, that command will not have `-statetable`.

Related Information

Interfacing with Encounter Conformal Logical Equivalence Checker in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects this command

[write do lec](#)

wlec_parallel_threads

wlec_parallel_threads *integer*

Default: 0

Read-write root attribute. Adds the Conformal LEC command `set parallel` option with the `-threads` option to the dofile. The attribute value specifies the number of parallel threads.

Note: This attribute takes precedence over the wlec_compare_threads attribute.

Related Information

Interfacing with Encounter Conformal Logical Equivalence Checker in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects this command

[write do lec](#)

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

wlec_set_cdn_synth_root

```
wlec_set_cdn_synth_root {false | true}
```

Default: false

Read-write root attribute. This attribute only applies if the design contains ChipWare or third party libraries. By default, the `CDN_SYNTH_ROOT` environment variable will not be set in the generated dofile. This allows you the flexibility to manually set `CDN_SYNTH_ROOT`.

If this attribute is set to `true`, the `CDN_SYNTH_ROOT` is set to the directory in which RTL Compiler is installed. This ensures that the ChipWare/third party library components are picked from the same RTL Compiler session that was used during synthesis.

Related Information

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

Affects this command

[write do lec](#)

wlec_uniquify

```
wlec_uniquify {true | false}
```

Default: true

Read-write root attribute. Controls uniquification of all modules in the design. If this attribute is set to `true`, the `write_do_lec` command adds the following LEC command to the generated dofile:

```
uniquify -all -nolib
```

This attribute is only effective if the generated dofile performs a hierarchical Conformal comparison.

Note: If the RTL has instantiated ChipWare components, the `uniquify` command will be generated in the dofile regardless of the value of the `wlec_uniquify` attribute. This is to prevent these modules from getting skipped during hierarchical compare.

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

Related Information

[Interfacing with Encounter Conformal Logical Equivalence Checker in Interfacing between Encounter RTL Compiler and Encounter Conformal](#)

Affects this command

[write_do_lec](#)

wlec_use_lec_model

wlec_use_lec_model {true | false}

Default: true

Read-write [root](#) attribute. Controls whether all ChipWare simulation models should be included.

If this attribute is set to `false`, the simulation for all ChipWare components will be loaded.

If set to `true`, the simulation model for all ChipWare components will be loaded except the following two:

CW_mult

DW02_mult

The generated dofile uses the Conformal built-in models instead. For these two components, the `write_do_lec` command

- Assumes the existence of LEC built-in models if

```
set_attribute wlec_use_lec_model true /
```
- Adds the RTL Compiler provided models into the dofile if

```
set_attribute wlec_use_lec_model false /
```

This attribute is effective only if the golden design is the RTL code that was loaded into RTL Compiler.

Related Information

[Interfacing with Encounter Conformal Logical Equivalence Checker in Interfacing between Encounter RTL Compiler and Encounter Conformal](#)

Affects this command

[write_do_lec](#)

write_sv_port_wrapper

```
write_sv_port_wrapper {false | true}
```

Default: false

Read-write root attribute. Determines if the `write_hdl` command should write out a wrapper to connect the original I/O interfaces to the port names in the Verilog it writes out.

Note: This attribute applies only to RTL using System Verilog interfaces. By default, the I/O ports in the Verilog netlist written by the `write_hdl` command do not match the test bench because interfaces were used.

Example

Consider the following RTL:

```
interface ffbus (input logic c, d, output logic q);
endinterface: ffbus

module ff (ffbus a);
    always @(posedge a.c) a.q <= a.d;
endmodule
```

Set the `write_sv_port_wrapper` attribute to `true` as shown in the following script:

```
set_attr library tutorial.lbr
set_attr write_sv_port_wrapper true
read_hdl -sv 1.v
elab
write_hdl
```

The `write_hdl` command writes out the following netlist:

```
module ff(.a({a_q, a_d, a_c}));
    input a_c, a_d;
    output a_q;
    wire a_c, a_d;
    wire a_q;
    CDN_flop a_q_reg(.clk (a_c), .d (a_d), .sena (1'b1), .aclr (1'b0),
                      .apre (1'b0), .srl (1'b0), .srd (1'b0), .q (a_q));
endmodule

`ifdef SYNTHESIS
`else
module CDN_flop(clk, d, sena, aclr, apre, srl, srd, q);
    input clk, d, sena, aclr, apre, srl, srd;
    output q;
    wire clk, d, sena, aclr, apre, srl, srd;
    wire q;
    reg qi;
    assign #1 q = qi;
endmodule
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

```
always
  @(posedge clk or posedge apre or posedge aclr)
    if (aclr)
      qi = 0;
    else if (apre)
      qi = 1;
    else if (sr1)
      qi = srd;
    else begin
      if (sena)
        qi = d;
    end
  initial
    qi = 1'b0;
endmodule
`endif
```

Related Information

Affects this command: [write_hdl](#)

write_vlog_bit_blast_bus_connections

```
write_vlog_bit_blast_bus_connections {false | true}
```

Default: false

Read-write [root](#) attribute. Determines if the `write_hdl` command will write out the busses connected to the instance pins as individual bits (in a bit-blasted style).

Examples

If the default value of `false` is maintained, the bus connection style for instance pins is the merged style. In the following example, `a` and `b` are each four bit busses:

```
sub sub(.a ({a[3], 1'b0, a[1:0]}), .b (b));
```

If the attribute is set to `true`, the instance pins will be bit-blasted:

```
sub sub(.a ({a[3],1'b0,a[1],a[0]}), .b ({b[3],b[2],b[1],b[0]}));
```

Related Information

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_constants](#) on page 294

[write_vlog_bit_blast_mapped_ports](#) on page 296

[write_vlog_bit_blast_tech_cell](#) on page 298

[write_vlog_convert_onebit_vector_to_scalar](#) on page 301
[write_vlog_declare_wires](#) on page 302
[write_vlog_empty_module_for_black_box](#) on page 304
[write_vlog_empty_module_for_logic_abstract](#) on page 305
[write_vlog_generic_gate_define](#) on page 307
[write_vlog_line_wrap_limit](#) on page 308
[write_vlog_no_negative_index](#) on page 309
[write_vlog_port_association_style](#) on page 327
[write_vlog_preserve_net_name](#) on page 310
[write_vlog_top_module_first](#) on page 311
[write_vlog_unconnected_port_style](#) on page 312
[write_vlog_wor_wand](#) on page 314

write_vlog_bit_blast_constants

`write_vlog_bit_blast_constants {false | true}`

Default: false

Read-write [root](#) attribute. Determines if the `write_hdl` command will write out constants as individual bits (in a bit-blasted style).

Example

- For the following example RTL:

```
// filename: test.v
module leaf (y, a, b);
    input [7:0] a, b;
    output [7:0] y;
    assign y = a + b;
endmodule

module top (y, a, b);
    input [7:0] a;
    input [3:0] b;
    output [7:0] y;
    leaf u1 (y, a, {4'b0110, b});
endmodule
```

If you set the `write_vlog_bit_blast_constants` attribute to `false` as follows:

```
rc:/> set_attribute library tutorial.lbr
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

```
rc:/> read_hdl test.v
rc:/> elaborate
rc:/> set_attribute write_vlog_bit_blast_constants false
rc:/> write_hdl
```

Then the `write_hdl` command writes out the following output:

```
module leaf (y, a, b);
...
endmodule

module top (y, a, b);
    input [7:0] a;
    input [3:0] b;
    output [7:0] y;
    leaf u1 (y, a, {4'b0110, b});
endmodule
```

If you set the `write_vlog_bit_blast_constants` attribute to true, the `write_hdl` command writes out the following netlist:

```
module leaf (y, a, b);
...
endmodule

module top (y, a, b);
    input [7:0] a;
    input [3:0] b;
    output [7:0] y;
    leaf u1 (y, a, {1'b0, 1'b1, 1'b1, 1'b0, b});
endmodule
```

Related Information

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293
[write_vlog_bit_blast_mapped_ports](#) on page 296
[write_vlog_bit_blast_tech_cell](#) on page 298
[write_vlog_convert_onebit_vector_to_scalar](#) on page 301
[write_vlog_declare_wires](#) on page 302
[write_vlog_empty_module_for_black_box](#) on page 304
[write_vlog_empty_module_for_logic_abstract](#) on page 305
[write_vlog_generic_gate_define](#) on page 307
[write_vlog_line_wrap_limit](#) on page 308
[write_vlog_no_negative_index](#) on page 309
[write_vlog_port_association_style](#) on page 327

[write_vlog_preserve_net_name](#) on page 310
[write_vlog_top_module_first](#) on page 311
[write_vlog_unconnected_port_style](#) on page 312
[write_vlog_wor_wand](#) on page 314

write_vlog_bit_blast_mapped_ports

`write_vlog_bit_blast_mapped_ports {false | true}`

Default: false

Read-write [root](#) attribute. Indicates if mapped ports are separated into bits (bit blasted).

Example

- For the following example RTL:

```
// filename: test.v
module leaf (y, a, b);
    input [5:0] a, b;
    output [5:0] y;
    assign y = a + b;
endmodule

module top (y, a, b);
    input [5:0] a;
    input [2:0] b;
    output [5:0] y;
    leaf u1 (y, a, {3'b010, b});
endmodule
```

If you use the following script:

```
set_attribute library tutorial.lbr
read_hdl test.v
elaborate
set_attribute write_vlog_bit_blast_mapped_ports false
write_hdl
```

Then the `write_hdl` command writes out the following netlist:

```
module leaf (y, a, b);
    input [5:0] a, b;
    output [5:0] y;
    ...
endmodule
module top (y, a, b);
    input [5:0] a;
    input [2:0] b;
    output [5:0] y;
    leaf u1 (y, a, {3'b010, b});
endmodule
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

- If you set the `write_vlog_bit_blast_mapped_ports` attribute to `true`, then the `write_hdl` command writes out the following netlist:

```
module leaf (y_5, y_4, y_3, y_2, y_1, y_0, a_5, a_4, a_3, a_2, a_1, a_0, b_5,
             b_4, b_3, b_2, b_1, b_0);
    input a_5, a_4, a_3, a_2, a_1, a_0;
    input b_5, b_4, b_3, b_2, b_1, b_0;
    output y_5, y_4, y_3, y_2, y_1, y_0;
...
endmodule

module top (y_5, y_4, y_3, y_2, y_1, y_0, a_5, a_4, a_3, a_2, a_1, a_0,
            b_2, b_1, b_0);
    input a_5, a_4, a_3, a_2, a_1, a_0;
    input b_2, b_1, b_0;
    output y_5, y_4, y_3, y_2, y_1, y_0;
    leaf u1 (y_5, y_4, y_3, y_2, y_1, y_0,
              a_5, a_4, a_3, a_2, a_1, a_0,
              1'b0, 1'b1, 1'b0, b_2, b_1, b_0);
endmodule
```

Related Information

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293

[write_vlog_bit_blast_constants](#) on page 294

[write_vlog_bit_blast_tech_cell](#) on page 298

[write_vlog_convert_onebit_vector_to_scalar](#) on page 301

[write_vlog_declare_wires](#) on page 302

[write_vlog_empty_module_for_black_box](#) on page 304

[write_vlog_empty_module_for_logic_abstract](#) on page 305

[write_vlog_generic_gate_define](#) on page 307

[write_vlog_line_wrap_limit](#) on page 308

[write_vlog_no_negative_index](#) on page 309

[write_vlog_port_association_style](#) on page 327

[write_vlog_preserve_net_name](#) on page 310

[write_vlog_top_module_first](#) on page 311

[write_vlog_unconnected_port_style](#) on page 312

[write_vlog_wor_wand](#) on page 314

write_vlog_bit_blast_tech_cell

```
write_vlog_bit_blast_tech_cell {false | true}
```

Default: false

Read-write root attribute. Determines if the `write_hdl` command will write out ports of technology cells that are explicitly instantiated in the netlist as individual bits (in a bit-blasted style).

Examples

Consider the following RTL code:

```
module lte_modem_fft_s3_ram256x38_1_fft (sin, we_n, \din[5], \din[4], \din[3],  
    \din[2], \din[1], \din[0], sout, clk, \addr[7], \addr[6], \addr[5], \addr[4],  
    \addr[3], \addr[2], \addr[1], \addr[0], \dout[5], \dout[4], \dout[3],  
    \dout[2], \dout[1], \dout[0], slp_ret_n, shift_n, slp_nret_n, scan_n, cs_n);  
    input sin;  
    input we_n;  
    input \din[5], \din[4], \din[3], \din[2], \din[1], \din[0];  
    output sout;  
    input clk;  
    input \addr[7], \addr[6], \addr[5], \addr[4], \addr[3], \addr[2],  
        \addr[1], \addr[0];  
    output \dout[5], \dout[4], \dout[3], \dout[2], \dout[1], \dout[0];  
    input slp_ret_n;  
    input shift_n;  
    input slp_nret_n;  
    input scan_n;  
    input cs_n;  
  
    // Internal wires  
    wire FE_OFN200_s2_b_im_p_2_5_;  
    wire FE_OFN199_s2_b_im_p_2_1_;  
  
    qcsram1111_22rwdng00_256x38_4_stdsp_rhrf M1 (  
        .\dout[5] (\dout[5]), .\dout[4] (\dout[4]), .\dout[3] (\dout[3]),  
        .\dout[2] (\dout[2]), .\dout[1] (\dout[1]), .\dout[0] (\dout[0]),  
        .\din[5] (FE_OFN200_s2_b_im_p_2_5_), .\din[4] (\din[4]),  
        .\din[3] (\din[3]), .\din[2] (\din[2]),  
        .\din[1] (FE_OFN199_s2_b_im_p_2_1_), .\din[0] (\din[0]),  
        .\addr[7] (\addr[7]), .\addr[6] (\addr[6]), .\addr[5] (\addr[5]),  
        .\addr[4] (\addr[4]), .\addr[3] (\addr[3]), .\addr[2] (\addr[2]),  
        .\addr[1] (\addr[1]), .\addr[0] (\addr[0]),  
        .we_n(we_n),  
        .sout(sout),  
        .slp_ret_n(slp_ret_n),  
        .slp_nret_n(slp_nret_n),  
        .sin(sin),  
        .shift_n(shift_n),  
        .scan_n(scan_n),  
        .cs_n(cs_n),  
        .clk(clk));  
endmodule
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

The following examples show how the setting of the `write_vlog_bit_blast_tech_cell` attribute affect the generated netlist. You can compare the bolded text in both examples to see the effect.

- Assume you use the following script:

```
set_attribute library lte_modem_ram_0.99v_ss_m30_cmax_m30_1.0.lib.gz
read_netlist write_vlog_bit_blast_tech_cell.v
set_attribute write_vlog_bit_blast_tech_cell false
write_hdl
```

Then the `write_hdl` command writes out the following netlist:

```
module lte_modem_fft_s3_ram256x38_1_fft(sin, we_n, \din[5], \din[4],
\ din[3], \din[2], \din[1], \din[0], sout, clk, \addr[7], \addr[6],
\addr[5], \addr[4], \addr[3], \addr[2], \addr[1], \addr[0], \dout[5],
\dout[4], \dout[3], \dout[2], \dout[1], \dout[0], slp_ret_n, shift_n,
slp_nret_n, scan_n, cs_n);
    input sin, we_n, \din[5], \din[4], \din[3], \din[2], \din[1],
    \din[0], clk, \addr[7], \addr[6], \addr[5], \addr[4], \addr[3],
    \addr[2], \addr[1], \addr[0], slp_ret_n, shift_n, slp_nret_n,
    scan_n, cs_n;
    output sout, \dout[5], \dout[4], \dout[3], \dout[2], \dout[1],
    \dout[0];
    wire sin, we_n, \din[5], \din[4], \din[3], \din[2], \din[1],
    \din[0], clk, \addr[7], \addr[6], \addr[5], \addr[4], \addr[3],
    \addr[2], \addr[1], \addr[0], slp_ret_n, shift_n, slp_nret_n,
    scan_n, cs_n;
    wire sout, \dout[5], \dout[4], \dout[3], \dout[2], \dout[1], \dout[0];
    wire FE_OFN199_s2_b_im_p_2_1_, FE_OFN200_s2_b_im_p_2_5_;
    qcsram1111_22rwdng00_256x38_4_stdsp_rhrf M1(.clk (clk), .sin (sin),
    .we_n (we_n), .din ({FE_OFN200_s2_b_im_p_2_5_, \din[4], \din[3],
    \din[2], FE_OFN199_s2_b_im_p_2_1_, \din[0]}), .addr ({\addr[7],
    \addr[6], \addr[5], \addr[4], \addr[3], \addr[2], \addr[1],
    \addr[0]}), .shift_n (shift_n), .slp_ret_n (slp_ret_n),
    .slp_nret_n (slp_nret_n), .scan_n (scan_n), .cs_n (cs_n),
    .sout (sout), .dout ({\dout[5], \dout[4], \dout[3], \dout[2],
    \dout[1], \dout[0]}));
endmodule
```

- If you set the `write_vlog_bit_blast_tech_cell` attribute to `true`, then the `write_hdl` command writes out the following netlist.

```
module lte_modem_fft_s3_ram256x38_1_fft(sin, we_n, \din[5], \din[4],
\ din[3], \din[2], \din[1], \din[0], sout, clk, \addr[7], \addr[6],
\addr[5], \addr[4], \addr[3], \addr[2], \addr[1], \addr[0], \dout[5],
\dout[4], \dout[3], \dout[2], \dout[1], \dout[0], slp_ret_n, shift_n,
slp_nret_n, scan_n, cs_n);
    input sin, we_n, \din[5], \din[4], \din[3], \din[2], \din[1],
    \din[0], clk, \addr[7], \addr[6], \addr[5], \addr[4], \addr[3],
    \addr[2], \addr[1], \addr[0], slp_ret_n, shift_n, slp_nret_n,
    scan_n, cs_n;
    output sout, \dout[5], \dout[4], \dout[3], \dout[2], \dout[1],
    \dout[0];
    wire sin, we_n, \din[5], \din[4], \din[3], \din[2], \din[1],
    \din[0], clk, \addr[7], \addr[6], \addr[5], \addr[4], \addr[3],
    \addr[2], \addr[1], \addr[0], slp_ret_n, shift_n, slp_nret_n,
    scan_n, cs_n;
    wire sout, \dout[5], \dout[4], \dout[3], \dout[2], \dout[1], \dout[0];
    wire FE_OFN199_s2_b_im_p_2_1_, FE_OFN200_s2_b_im_p_2_5_;
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

```
qcsram1111_22rwdng00_256x38_4_stdsp_rhrc M1(.clk (clk), .sin (sin),
      .we_n (we_n), .\din[5] (FE_OFN200_S2_B_IM_P_2_5),
      .\din[4] (\din[4]), .\din[3] (\din[3]), .\din[2] (\din[2]),
      .\din[1] (FE_OFN199_S2_B_IM_P_2_1), .\din[0] (\din[0]),
      .\addr[7] (\addr[7]), .\addr[6] (\addr[6]), .\addr[5] (\addr[5]),
      .\addr[4] (\addr[4]), .\addr[3] (\addr[3]), .\addr[2] (\addr[2]),
      .\addr[1] (\addr[1]), .\addr[0] (\addr[0]), .shift_n (shift_n),
      .slp_ret_n (slp_ret_n), .slp_nret_n (slp_nret_n), .scan_n (scan_n),
      .cs_n (cs_n), .sout (sout), .\dout[5] (\dout[5]),
      .\dout[4] (\dout[4]), .\dout[3] (\dout[3]), .\dout[2] (\dout[2]),
      .\dout[1] (\dout[1]), .\dout[0] (\dout[0]));
endmodule
```

Related Information

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293

[write_vlog_bit_blast_constants](#) on page 294

[write_vlog_bit_blast_mapped_ports](#) on page 296

[write_vlog_convert_onebit_vector_to_scalar](#) on page 301

[write_vlog_declare_wires](#) on page 302

[write_vlog_empty_module_for_black_box](#) on page 304

[write_vlog_empty_module_for_logic_abstract](#) on page 305

[write_vlog_generic_gate_define](#) on page 307

[write_vlog_line_wrap_limit](#) on page 308

[write_vlog_no_negative_index](#) on page 309

[write_vlog_port_association_style](#) on page 327

[write_vlog_preserve_net_name](#) on page 310

[write_vlog_top_module_first](#) on page 311

[write_vlog_unconnected_port_style](#) on page 312

[write_vlog_wor_wand](#) on page 314

write_vlog_convert_onebit_vector_to_scalar

```
write_vlog_convert_onebit_vector_to_scalar {false | true}
```

Default: false

Read-write root attribute. Specifies whether to write out one bit vectors present in RTL as scalars in the generated netlist.

Example

Consider the following RTL code:

```
module x(X, A1, A2);
    input [0:0] A1 ;
    input [0:0] A2 ;
    output X ;
    and U$1(X, A1, A2) ;
endmodule
```

- If you use the default value, the netlist written out will be similar to:

```
module x(X, A1, A2);
    input [0:0] A1, A2;
    wire [0:0] A1, A2;
    wire X;
    and U$1 (X, A1, A2);
endmodule
```

- If you set this attribute to true, the generated netlist will be similar to:

```
module x(X, A1, A2);
    input A1, A2;
    output X;
    wire A1, A2;
    wire X;
    and U$1 (X, A1, A2);
endmodule
```

Related Information

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293

[write_vlog_bit_blast_constants](#) on page 294

[write_vlog_bit_blast_mapped_ports](#) on page 296

[write_vlog_bit_blast_tech_cell](#) on page 298

[write_vlog_declare_wires](#) on page 302

[write_vlog_empty_module_for_black_box](#) on page 304

[write_vlog_empty module for logic abstract](#) on page 305
[write_vlog_generic gate define](#) on page 307
[write_vlog_line wrap limit](#) on page 308
[write_vlog_no_negative index](#) on page 309
[write_vlog_port association style](#) on page 327
[write_vlog_preserve net name](#) on page 310
[write_vlog_top module first](#) on page 311
[write_vlog_unconnected port style](#) on page 312
[write_vlog_wor wand](#) on page 314

write_vlog_declare_wires

`write_vlog_declare_wires {true | false}`

Default: true

Read-write [root](#) attribute. Specifies whether to generate a netlist with or without implicit wires. When set to false, RTL Compiler suppresses the declarations of implicit wires.

Example

- For the following example RTL:

```
// filename: test.v
module test (y, a, b, c);
    input [3:0] a, b, c;
    output [3:0] y;
    assign y = a & b & c;
endmodule
```

If you set the `write_vlog_declare_wires` to true as follows:

```
rc:/> set_attribute library tutorial.lbr
rc:/> read_hdl test.v
rc:/> elaborate
rc:/> set_attribute write_vlog_declare_wires true
rc:/> write_hdl
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

Then the `write_hdl` command writes out the following netlist:

```
module test (y, a, b, c);
    input [3:0] a, b, c;
    output [3:0] y;
    wire [3:0] a, b, c;
    wire [3:0] y;
    wire n_9, n_10, n_11, n_12;
    and g1 (n_9, a[0], b[0]);
    and g3 (n_10, a[1], b[1]);
    and g4 (n_11, a[2], b[2]);
    and g5 (n_12, a[3], b[3]);
    and g6 (y[0], n_9, c[0]);
    and g2 (y[1], n_10, c[1]);
    and g7 (y[2], n_11, c[2]);
    and g8 (y[3], n_12, c[3]);
endmodule
```

If you set the `write_vlog_declare_wires` to `false`, then the `write_hdl` command writes out the following netlist:

```
module test (y, a, b, c);
    input [3:0] a, b, c;
    output [3:0] y;
    and g1 (n_9, a[0], b[0]);
    and g3 (n_10, a[1], b[1]);
    and g4 (n_11, a[2], b[2]);
    and g5 (n_12, a[3], b[3]);
    and g6 (y[0], n_9, c[0]);
    and g2 (y[1], n_10, c[1]);
    and g7 (y[2], n_11, c[2]);
    and g8 (y[3], n_12, c[3]);
endmodule
```

Related Information

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293

[write_vlog_bit_blast_constants](#) on page 294

[write_vlog_bit_blast_mapped_ports](#) on page 296

[write_vlog_bit_blast_tech_cell](#) on page 298

[write_vlog_convert_onebit_vector_to_scalar](#) on page 301

[write_vlog_empty_module_for_black_box](#) on page 304

[write_vlog_empty_module_for_logic_abstract](#) on page 305

[write_vlog_generic_gate_define](#) on page 307

[write_vlog_line_wrap_limit](#) on page 308

[write_vlog_no_negative_index](#) on page 309

[write_vlog_port_association_style](#) on page 327
[write_vlog_preserve_net_name](#) on page 310
[write_vlog_top_module_first](#) on page 311
[write_vlog_unconnected_port_style](#) on page 312
[write_vlog_wor_wand](#) on page 314

write_vlog_empty_module_for_black_box

`write_vlog_empty_module_for_black_box {false | true}`

Default: false

Read-write [root](#) attribute: Controls whether an empty module will be written out for an unresolved black box. If set to true, the unresolved black box is written as an empty module. If set to false, the unresolved black box model remains unresolved in the netlist.

Related Information

[Modeling Logic Abstracts in HDL Modeling in Encounter RTL Compiler](#).

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293
[write_vlog_bit_blast_constants](#) on page 294
[write_vlog_bit_blast_mapped_ports](#) on page 296
[write_vlog_bit_blast_tech_cell](#) on page 298
[write_vlog_convert_onebit_vector_to_scalar](#) on page 301
[write_vlog_declare_wires](#) on page 302
[write_vlog_empty_module_for_black_box](#) on page 304
[write_vlog_empty_module_for_logic_abstract](#) on page 305
[write_vlog_generic_gate_define](#) on page 307
[write_vlog_line_wrap_limit](#) on page 308
[write_vlog_no_negative_index](#) on page 309
[write_vlog_port_association_style](#) on page 327
[write_vlog_preserve_net_name](#) on page 310

[write_vlog_top module first](#) on page 311

[write_vlog_unconnected_port_style](#) on page 312

[write_vlog_wor_wand](#) on page 314

write_vlog_empty_module_for_logic_abstract

`write_vlog_empty_module_for_logic_abstract {true | false}`

Default: true

Read-write [root](#) attribute: Controls how an unresolved logic abstract model is written out in a Verilog netlist. If set to `true`, the unresolved logic abstract model is written as an empty module. If set to `false`, the unresolved logic abstract model remains unresolved in the netlist.

Example

- For the following example RTL:

```
// filename: test.vhd
library ieee;
use ieee.std_logic_1164.all;
entity my_top is
    generic (w : integer := 4);
    port (a, b, c : in std_logic_vector (w-1 downto 0);
          y : out std_logic_vector (w-1 downto 0));
end my_top;
architecture rtl of my_top is
    signal t : std_logic_vector (w-1 downto 0);
    component my_sub
        generic (w : integer := 4);
        port (p, q : in std_logic_vector (w-1 downto 0);
              x : out std_logic_vector (w-1 downto 0));
    end component;
begin
    u1: my_sub generic map (w => w)
        port map (p => a, q => b, x => t);
    y <= t or c;
end rtl;
```

If you set the `write_vlog_empty_module_for_logic_abstract` attribute to `true`:

```
rc:/> set_attribute library tutorial.lbr
rc:/> read_hdl -vhdl test.vhd
rc:/> elaborate
rc:/> set_attribute write_vlog_empty_module_for_logic_abstract true
rc:/> write_hdl
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

Then the `write_hdl` command writes out the following netlist:

```
module my_sub_w4 (p, q, x); // an empty module
    input [3:0] p, q;
    output [3:0] x;
endmodule

module my_top (a, b, c, y);
    input [3:0] a, b, c;
    output [3:0] y;
    wire t0, t1, t2, t3;
    my_sub_w4 u1 (.p(a), .q(b), .x({t3, t2, t1, t0}));
    or g1 (y[0], t0, c[0]);
    or g2 (y[1], t1, c[1]);
    or g3 (y[2], t2, c[2]);
    or g4 (y[3], t3, c[3]);
endmodule
```

If you set the `write_vlog_empty_module_for_logic_abstract` attribute to `false`, then the `write_hdl` command writes out the following netlist:

```
module my_top (a, b, c, y);
    input [3:0] a, b, c;
    output [3:0] y;
    wire t0, t1, t2, t3;
    my_sub_w4 u1 (.p(a), .q(b), .x({t3, t2, t1, t0}));
    or g1 (y[0], t0, c[0]);
    or g2 (y[1], t1, c[1]);
    or g3 (y[2], t2, c[2]);
    or g4 (y[3], t3, c[3]);
endmodule
```

Related Information

[Modeling Logic Abstracts in HDL Modeling in Encounter RTL Compiler.](#)

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293

[write_vlog_bit_blast_constants](#) on page 294

[write_vlog_bit_blast_mapped_ports](#) on page 296

[write_vlog_bit_blast_tech_cell](#) on page 298

[write_vlog_convert_onebit_vector_to_scalar](#) on page 301

[write_vlog_declare_wires](#) on page 302

[write_vlog_empty_module_for_black_box](#) on page 304

[write_vlog_empty_module_for_black_box](#) on page 304

[write_vlog_generic_gate_define](#) on page 307

[write_vlog_line_wrap_limit](#) on page 308

[write vlog no negative index](#) on page 309
[write vlog port association style](#) on page 327
[write vlog preserve net name](#) on page 310
[write vlog top module first](#) on page 311
[write vlog unconnected port style](#) on page 312
[write vlog wor wand](#) on page 314

write_vlog_generic_gate_define

`write_vlog_generic_gate_define string`

Default: RC_CDN_GENERIC_GATE

Read-write [root](#) attribute. Specifies the Verilog macro name that is written in the output Verilog netlist to enclose module descriptions of RTL Compiler built-in generic gates within '`ifdef`' and '`endif`' Verilog compiler directives.

A netlist written after elaborate can have the following module description:

```
'ifdef RC_CDN_GENERIC_GATE
`else
module CDN_flop(clk, d, sena, aclr, apre, srl, srd, q);
...
`endif
```

You must include module descriptions of generic gates in the output netlist to simulate the design, but the module descriptions are not needed to read in the netlist in RTL Compiler.

Related Information

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293

[write_vlog_bit_blast_constants](#) on page 294

[write_vlog_bit_blast_mapped_ports](#) on page 296

[write_vlog_bit_blast_tech_cell](#) on page 298

[write_vlog_convert_onebit_vector_to_scalar](#) on page 301

[write_vlog_declare_wires](#) on page 302

[write_vlog_empty_module_for_black_box](#) on page 304

[write_vlog_empty_module_for_logic_abstract](#) on page 305

[write vlog line wrap limit](#) on page 308
[write vlog no negative index](#) on page 309
[write vlog port association style](#) on page 327
[write vlog preserve net name](#) on page 310
[write vlog top module first](#) on page 311
[write vlog unconnected port style](#) on page 312
[write vlog wor wand](#) on page 314

write_vlog_line_wrap_limit

`write_vlog_line_wrap_limit integer`

Default: 72

Read-write [root](#) attribute. Specifies the number of printable characters on a single line in the written netlist.

Note: This attribute is not supported for writing out encrypted modules.

Related Information

Affects these commands: [write hdl](#)

Related Attributes: [write vlog bit blast bus connections](#) on page 293
[write vlog bit blast constants](#) on page 294
[write vlog bit blast mapped ports](#) on page 296
[write vlog bit blast tech cell](#) on page 298
[write vlog convert onebit vector to scalar](#) on page 301
[write vlog declare wires](#) on page 302
[write vlog empty module for black box](#) on page 304
[write vlog empty module for logic abstract](#) on page 305
[write vlog generic gate define](#) on page 307
[write vlog no negative index](#) on page 309
[write vlog port association style](#) on page 327
[write vlog preserve net name](#) on page 310

[write_vlog_top_module_first](#) on page 311
[write_vlog_unconnected_port_style](#) on page 312
[write_vlog_wor_wand](#) on page 314

write_vlog_no_negative_index

`write_vlog_no_negative_index {false | true}`

Default: false

Read-write [root](#) attribute. When set to true, RTL Compiler converts negative ranges to the standard (normalized) range and keeps the direction unchanged as follows:

`input a[-4:0] --> input a[0:4]`
`input b[0:-4] --> input b[4:0]`

Note: This attribute also applies to internal bus wires.

Related Information

Affects these commands: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293
[write_vlog_bit_blast_constants](#) on page 294
[write_vlog_bit_blast_mapped_ports](#) on page 296
[write_vlog_bit_blast_tech_cell](#) on page 298
[write_vlog_convert_onebit_vector_to_scalar](#) on page 301
[write_vlog_declare_wires](#) on page 302
[write_vlog_empty_module_for_black_box](#) on page 304
[write_vlog_empty_module_for_logic_abstract](#) on page 305
[write_vlog_generic_gate_define](#) on page 307
[write_vlog_line_wrap_limit](#) on page 308
[write_vlog_port_association_style](#) on page 327
[write_vlog_preserve_net_name](#) on page 310
[write_vlog_top_module_first](#) on page 311
[write_vlog_unconnected_port_style](#) on page 312
[write_vlog_wor_wand](#) on page 314

write_vlog_preserve_net_name

`write_vlog_preserve_net_name {true | false}`

Default: `false`

Read-write `root` attribute. When set to `true`, RTL Compiler preserves the wire names present in the input design. This will result in an increase in the number of assign statements in the output netlist.

Note: This attribute is supported in both RTL and Structural flows.

Related Information

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293
[write_vlog_bit_blast_constants](#) on page 294
[write_vlog_bit_blast_mapped_ports](#) on page 296
[write_vlog_bit_blast_tech_cell](#) on page 298
[write_vlog_convert_onebit_vector_to_scalar](#) on page 301
[write_vlog_declare_wires](#) on page 302
[write_vlog_empty_module_for_black_box](#) on page 304
[write_vlog_empty_module_for_logic_abstract](#) on page 305
[write_vlog_generic_gate_define](#) on page 307
[write_vlog_line_wrap_limit](#) on page 308
[write_vlog_no_negative_index](#) on page 309
[write_vlog_port_association_style](#) on page 327
[write_vlog_top_module_first](#) on page 311
[write_vlog_unconnected_port_style](#) on page 312
[write_vlog_wor_wand](#) on page 314

write_vlog_top_module_first

```
write_vlog_top_module_first {false | true}
```

Default: false

Read-write root attribute. Indicates if the top module is written first in the Verilog output.

Example

- For the following example RTL:

```
module top(in, out);
    input in;
    output out;
    sub a1(.in(in), .out(out));
endmodule

module sub(in, out);
    input in;
    output out;
    assign out = in;
endmodule
```

If you set the `write_vlog_top_module_first` attribute to `true`, the `write_hdl` command writes out the netlist with the top module first. The default behavior is shown below:

```
module sub(in, out);
    input in;
    output out;
    assign out = in;
endmodule

module top(in, out);
    input in;
    output out;
    sub a1(.in(in), .out(out));
endmodule
```

Related Information

Affects these commands: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293

[write_vlog_bit_blast_constants](#) on page 294

[write_vlog_bit_blast_mapped_ports](#) on page 296

[write_vlog_bit_blast_tech_cell](#) on page 298

[write_vlog_convert_onebit_vector_to_scalar](#) on page 301

[write_vlog_declare_wires](#) on page 302

[write_vlog_empty module for black box on page 304](#)
[write_vlog_empty module for logic abstract on page 305](#)
[write_vlog_generic gate define on page 307](#)
[write_vlog_line wrap limit on page 308](#)
[write_vlog_no_negative_index on page 309](#)
[write_vlog_port association style on page 327](#)
[write_vlog_preserve_net_name on page 310](#)
[write_vlog_unconnected_port_style on page 312](#)
[write_vlog_wor_wand on page 314](#)

write_vlog_unconnected_port_style

`write_vlog_unconnected_port_style {full | none | partial}`

Default: full

Read-write root attribute. Selects the Verilog style for unconnected *instance* pins. By default, the `write_hdl` command writes out dummy wires for unconnected instance pins.

Note: This attribute does **not** apply to Verilog gate *primitives*.

Examples

- Consider the following RTL example:

```
module test1 (A1,A2,Y1,Y2);
    input A1,A2;
    output Y1,Y2;
    DELAY1 DL (.A(A2));
endmodule
```

RTL Compiler generates the following Verilog netlists depending upon the value of the `write_vlog_unconnected_port_style` attribute:

- ❑ `write_vlog_unconnected_port_style full`

```
module test1(A1, A2, Y1, Y2);
    input A1, A2;
    output Y1, Y2;
    wire A1, A2;
    wire Y1, Y2;
    wire UNCONNECTED;
    DELAY1 DL(.A (A2), .Z (UNCONNECTED));
endmodule
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

- ❑ `write_vlog_unconnected_port_style partial`

```
module test1(A1, A2, Y1, Y2);
    input A1, A2;
    output Y1, Y2;
    wire A1, A2;
    wire Y1, Y2;
    DELAY1 DL(.A (A2), .Z ());
endmodule
```

- ❑ `write_vlog_unconnected_port_style none`

```
module test1(A1, A2, Y1, Y2);
    input A1, A2;
    output Y1, Y2;
    wire A1, A2;
    wire Y1, Y2;
    DELAY1 DL(.A (A2));
endmodule
```

- Consider the following basic gate:

```
module test(in1,out1);
    input in1;
    output out1;
    not n1(w,in1);
endmodule
```

RTL Compiler generates the following Verilog netlist, independent of the setting of the attribute:

```
module test(in1,out1);
    input in1;
    output out1;
    not n1(w,in1);
endmodule
```

Related Information

Affects these commands: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293

[write_vlog_bit_blast_constants](#) on page 294

[write_vlog_bit_blast_mapped_ports](#) on page 296

[write_vlog_bit_blast_tech_cell](#) on page 298

[write_vlog_convert_onebit_vector_to_scalar](#) on page 301

[write_vlog_declare_wires](#) on page 302

[write_vlog_empty_module_for_logic_abstract](#) on page 305

[write_vlog_empty_module_for_logic_abstract](#) on page 305

[write_vlog_generic_gate_define](#) on page 307

[write_vlog_line_wrap_limit](#) on page 308
[write_vlog_no_negative_index](#) on page 309
[write_vlog_port_association_style](#) on page 327
[write_vlog_preserve_net_name](#) on page 310
[write_vlog_top_module_first](#) on page 311
[write_vlog_wor_wand](#) on page 314

write_vlog_wor_wand

`write_vlog_wor_wand {true | false}`

Default: true

Read-write `root` attribute. When set to `false`, a `wor` (wand) wire is declared as a `wire` rather than a `wor` (wand).

Examples

By default, RTL Compiler does not synthesize wor/wand logic, even if it is present in the HDL read into RTL Compiler. However, if a signal is originally declared as a `wor` signal, then the `write_hdl` command writes it out in the synthesized netlist.

- The following command prevents signals from being declared as wor/wand in the RTL Compiler generated netlist:

```
rc:/> set_attribute write_vlog_wor_wand false
```

For example, for the following RTL:

```
module leaf (y, a, b);
    parameter w = 4;
    input [w-1:0] a, b;
    output [w-1:0] y;
    assign y = a & b;
endmodule

module top (y, data_0, data_1, sel_0, sel_1);
    parameter w = 4;
    input [w-1:0] data_0, data_1, sel_0, sel_1;
    output [w-1:0] y;
    wor [w-1:0] y;
    leaf #(w) u0 (y, data_0, sel_0);
    leaf #(w) u1 (y, data_1, sel_1);
endmodule
```

RTL Compiler generates the following Verilog netlists depending upon the value of the `write_vlog_wor_wand` attribute:

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

- ❑ `write_vlog_wor_wand true (default)`

```
module leaf_w4(y, a, b);
  ...
endmodule

module top(y, data_0, data_1, sel_0, sel_1);
  input [3:0] data_0, data_1, sel_0, sel_1;
  output [3:0] y;
  wire [3:0] data_0, data_1, sel_0, sel_1, y;
  wor n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8; // <== wor
  leaf_w4 u0({n_7, n_5, n_3, n_1}, data_0, sel_0);
  leaf_w4 u1({n_8, n_6, n_4, n_2}, data_1, sel_1);
  or g1 (y[0], n_1, n_2);
  or g2 (y[1], n_3, n_4);
  or g3 (y[2], n_5, n_6);
  or g4 (y[3], n_7, n_8);
endmodule
```

- ❑ `write_vlog_wor_wand false`

```
module leaf_w4(y, a, b);
  ...
endmodule

module top(y, data_0, data_1, sel_0, sel_1);
  input [3:0] data_0, data_1, sel_0, sel_1;
  output [3:0] y;
  wire [3:0] data_0, data_1, sel_0, sel_1, y;
  wire n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8; // <== wire
  leaf_w4 u0({n_7, n_5, n_3, n_1}, data_0, sel_0);
  leaf_w4 u1({n_8, n_6, n_4, n_2}, data_1, sel_1);
  or g1 (y[0], n_1, n_2);
  or g2 (y[1], n_3, n_4);
  or g3 (y[2], n_5, n_6);
  or g4 (y[3], n_7, n_8);
endmodule
```

Related Information

Affects these commands: [write_hdl](#)

Related Attributes:

- [write_vlog_bit_blast_bus_connections](#) on page 293
- [write_vlog_bit_blast_constants](#) on page 294
- [write_vlog_bit_blast_mapped_ports](#) on page 296
- [write_vlog_bit_blast_tech_cell](#) on page 298
- [write_vlog_convert_onebit_vector_to_scalar](#) on page 301
- [write_vlog_declare_wires](#) on page 302
- [write_vlog_empty_module_for_black_box](#) on page 304
- [write_vlog_empty_module_for_logic_abstract](#) on page 305
- [write_vlog_generic_gate_define](#) on page 307

Attribute Reference for Encounter RTL Compiler

Input and Output—Root Attributes

[write_vlog_line_wrap_limit](#) on page 308

[write_vlog_no_negative_index](#) on page 309

[write_vlog_port_association_style](#) on page 327

[write_vlog_preserve_net_name](#) on page 310

[write_vlog_top_module_first](#) on page 311

[write_vlog_unconnected_port_style](#) on page 312

Design Attributes

Contain information about the specified design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design
```

or

```
set_attribute attribute_name attribute_value [find /des* -design name]
```

- To get a design attribute value, type

```
get_attribute attribute_name design
```

arch_filename

`arch_filename string`

Read-write `design` attribute. Returns the name and physical location of the source file that contains the specified architecture (in VHDL) or module (in Verilog).

Note: This attribute is only supported in the RTL flow.

Example

When multiple design files are loaded, it might be difficult to correlate a module or architecture to the file in which it was instantiated. The following example illustrates how to find the VHDL file for a particular submodule `my_block`, using the `get_attribute` command.

```
rc:/> get_attribute arch_filename /designs/top/subdesigns/my_block
```

The above command would return something like the following output, showing that the `my_block` submodule was instantiated in the file `block_arch.vhdl`:

```
.../rtl/archs/block_arch.vhdl
```

Related Information

[Using the get_attribute Command to Search for Information in Using Encounter RTL Compiler](#)

Related attributes:

(subdesign) `arch_filename` on page 337

(design) `entity_filename` on page 318

(subdesign) `entity_filename` on page 338

embedded_script

`embedded_script string`

Read-write design attribute. Stores the embedded SDC command script in the RTL code describing the module or entity of this design or subdesign.

Using the exec_embedded_script command applies the stored script onto the design or subdesign.

You can use a synthesis script to modify the contents of the `embedded_script` attribute before executing it.

The embedded script can have any SDC command that is supported by the `read_sdc` command.

Related Information

Related attributes: (subdesign) embedded_script on page 338

entity_filename

`entity_filename string`

Read-write design attribute. Returns the name and physical location of the source file that contains the specified VHDL entity. Since Verilog does not have the concept of VHDL entities, this attribute will return the file that contains the Verilog module. For Verilog modules, the `entity_filename` and `arch_filename` attributes behave identically.

Note: This attribute is only supported in the RTL flow.

Example

When multiple entity files are loaded, it might be difficult to correlate an entity to the file in which it was instantiated. The following example illustrates how to find the entity file for a particular submodule `my_block`, using the `get_attribute` command.

```
rc:/> get_attribute entity_filename /designs/top/subdesign/my_block
```

The above command would return something like the following output, showing that the `my_block` submodule was instantiated in the file `block_ent.vhdl`:

```
../rtl/entities/block_ent.vhdl
```

Related Information

Related attributes:	(design) arch_filename on page 317
	(subdesign) arch_filename on page 337
	(subdesign) entity_filename on page 338

hdl_all_filelist

```
hdl_all_filelist {hdl_library language_standard {define_value ... }  
{hdl_file ... } {search_path}}...
```

Read-write [design](#) attribute. Stores the information required by the `read_hdl` command to elaborate the design. The information is stored as a list of values that can be used as options for the `read_hdl` command. The list of files includes any ‘include and package files read into RTL Compiler for the specified design.

- *hdl_library* is the value for the `-library` option.
- *language_standard* corresponds to one of the following: `-v1995`, `-v2001`, `-vhdl1987`, `-vhdl1193`, `-vhdl2008`, or `-sv`.
For vhdl, the value of the `hdl_vhdl_read_version` attribute is appended to `-vhdl`.
- *define_value* is the value for the `-define` option (default: SYNTHESIS)
- *hdl_file* corresponds to the list of files required to elaborate the design.
- *search_path* is the directory of the location of the HDL files. An empty value signifies the current directory.

Since the attribute contains only those files that have been used during elaboration, you can use this attribute to prune unnecessary `read_hdl` commands from your scripts.

Note: This attribute is supported in the RTL flow and the structural flow.

Example

Consider the following RTL files:

File top.v

```
module top(a,b);  
    input [1:0] a;  
    output [1:0] b;  
    bot u1(.x(a),.y(b));  
endmodule
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Design Attributes

File bot.v

```
`include "siz.h"
`include "inc.h"
module bot(x,y);
    input [`SIZE:0] x;
    output [`SIZE:0] y;
    wire [`SIZE:0] tmp;
    assign y = `ADD(x,tmp);
endmodule
```

File siz.h

```
`define SIZE 1
```

File inc.h

```
`define ADD(a,b) a+b
```

Consider the following script:

```
set_attr library tutorial.lib
read_hdl top.v
read_hdl bot.v
elab top
```

After elaboration, the `hdl_all_filelist` for design `top` will return:

```
{default -v2001 {SYNTHESIS} {top.v} {}} {default -v2001 {SYNTHESIS} {bot.v siz.h inc.h} {}}
```

The difference between `hdl_filelist` and `hdl_all_filelist` is that `hdl_all_filelist` also lists the include files `siz.h` and `inc.h`.

Related Information

- | | |
|---------------------|--|
| Related attributes: | (design) hdl_filelist on page 321 |
| | (subdesign) hdl_all_filelist on page 339 |

hdl_config_name

`hdl_config_name` string

Read-write design attribute. Specifies the name of the configuration used to build this design.

Related Information

- | | |
|---------------------|---|
| Related attributes: | (subdesign) hdl_config_name on page 340 |
|---------------------|---|

hdl_filelist

```
hdl_filelist {hdl_library language_standard {define_value ...} {hdl_file ...} {search_path}}...
```

Read-write design attribute. Stores the information required by the `read_hdl` command to elaborate the design. The information is stored as a list of values that can be used as options for the `read_hdl` command.

- `hdl_library` is the value for the `-library` option.
- `language_standard` corresponds to one of the following: `-v1995`, `-v2001`, `-vhdl1987`, `-vhdl1193`, `-vhdl2008`, or `-sv`.

For vhdl, the value of the `hdl_vhdl_read_version` attribute is appended to `-vhdl`.

- `define_value` is the value for the `-define` option (default: SYNTHESIS)
- `hdl_file` corresponds to the list of files required to elaborate the design.
- `search_path` is the directory of the location of the HDL files. An empty value signifies the current directory.

Since the attribute contains only those files that have been used during elaboration, you can use this attribute to prune unnecessary `read_hdl` commands from your scripts.

Note: This attribute is supported in the RTL flow and the structural flow.

Examples

- The following command shows the value of the `hdl_filelist` attribute on design `m1` after elaborating the design.

```
rc:/ read_hdl -v2001 top.v
rc:/ read_hdl -vhdl -lib mylib sub.vhdl
rc:/ elaborate
rc:/ get_attribute hdl_filelist m1
{default -v2001 {SYNTHESIS} {top.v} {} } {mylib -vhdl1193 {SYNTHESIS} {sub.vhdl} {} }
```

- Consider the following RTL files:

File `top.v`

```
module top(a,b);
  ...
  mid u1(.c(a),.d(b));
endmodule
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Design Attributes

File mid.v

```
module mid(c,d);
  ...
  bot u2(.x(c),.y(d));
endmodule
```

File bot.v

```
module bot(x,y);
  ...
endmodule
```

File foo.v

```
module foo(q,w);
  ...
endmodule
```

Consider the following script:

```
set_attr library tutorial.lib
read_hdl top.v mid.v bot.v foo.v
elab top
```

After elaboration, the `hdl_file_list` for design `top` will return:

```
{default -v1995 {SYNTHESIS} {top.v mid.v bot.v} {}}
```

File `foo.v` is not stored because it was not used during elaboration.

Related Information

Affects these commands:

[elaborate](#)

[write do lec](#)

Affected by this attribute:

[hdl vhdl read version](#) on page 264

Related attributes:

(design) [hdl_all_filelist](#) on page 319

(subdesign) [hdl_filelist](#) on page 341

hdl_user_name

`hdl_user_name string`

Read-write design attribute. Represents the name of the Verilog module or the VHDL entity from which the given design was derived. The design's name may differ from the `hdl_user_name` value and from the addition of suffixes for the module name unqualification.

For example, a parameterized module named `adder` with `wA` and `wB` parameters may result in a design whose name is `adder_wA5_wB3`. However, the `hdl_user_name` attribute for this design would still be `adder`.

Related Information

Related attributes: (subdesign) [hdl_user_name](#) on page 342

hdl_v2001

`hdl_v2001 string`

Read-write design attribute. Sets the specified Verilog 2001 attributes on this design.

Example

The following command removes the Verilog 2001 attributes from the design:

```
set_attr hdl_v2001 {" " } [find . -design test]
```

Related Information

Set by this command: [read_netlist -v2001](#)

Related attributes: (instance) [hdl_v2001](#) on page 326
(pin) [hdl_v2001](#) on page 328
(pgpin) [hdl_v2001](#) on page 329
(port) [hdl_v2001](#) on page 336
(subdesign) [hdl_v2001](#) on page 342
(subport) [hdl_v2001](#) on page 346

protected

```
protected {false | true}
```

Default: false

Read-only design attribute. This attribute is set to `true` if the design object represents a module or entity whose HDL source code is encrypted or only partially encrypted.

Related Information

IP Protection in *Using Encounter RTL Compiler*

Affects this command: [write hdl](#)

[report datapath](#)

Related attribute: (hdl_arch) [protected](#) on page 331

 (subdesign) [protected](#) on page 343

wcdc_synchronizer_type

```
wcdc_synchronizer_type {dff | mux | module}...
```

Default: {dff mux}

Read-write design attribute. Specifies the type of synchronizer circuit used in the design. You can specify multiple values.

Example

The following command sets this attribute to `dff`.

```
set_attribute wcdc_synchronizer_type dff design
```

This attribute setting causes the following synchronization rules:

```
add synchronization rule r_1 -dff 2 infinity -sync_chain buffer single_chain \
-cdc_path logic multiple_destination -source -all -destination -all \
-from -all -to -all
add synchronization rule r_2 -dff 2 infinity -sync_chain wire single_chain \
-cdc_path buffer single_destination -source -all -destination -all \
-from -all -to -all
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Design Attributes

Related Information

[Interfacing with Encounter Conformal Extended Checks](#) in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects this command

[write do verify cdc](#)

Instance Attributes

Contain information about a combinational, sequential, or hierarchical instance in the specified design. Instance objects are found in the `instances_comb`, `instances_seq`, and `instances_hier` directories.

- To set an instance attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -instance name]
```

- To get an instance attribute value, type

```
get_attribute attribute_name [find /des*/design -instance name]
```

hdl_v2001

`hdl_v2001` *string*

Read-write instance attribute. Sets the specified Verilog 2001 attributes on this instance.

Example

The following command retrieves the Verilog 2001 attributes set on instance `test/u0`.

```
rc:/designs> get_attr hdl_v2001 [find . -instance test/u0]
{top_inst_name="u1"}
```

Related Information

Set by this command:

[read_netlist -v2001](#)

Related attributes:

(design) [hdl_v2001](#) on page 323

(pin) [hdl_v2001](#) on page 328

(pgpin) [hdl_v2001](#) on page 329

(port) [hdl_v2001](#) on page 336

(subdesign) [hdl_v2001](#) on page 342

(subport) [hdl_v2001](#) on page 346

write_vlog_port_association_style

```
write_vlog_port_association_style {default | positional | named}
```

Default: default

Read-write instance attribute. Determines the style for writing port connections of an instance. The style can be default, which specifies that the instance is written out the same as the original design input, positional, which specifies that the instance is written out as a positional instance, or named, which specifies that the instance is written out as a named instance.

Note: This attribute is supported only for hierarchical and blackbox instances.

Related Information

Affects this command: [write_hdl](#)

Related Attributes: [write_vlog_bit_blast_bus_connections](#) on page 293

[write_vlog_bit_blast_constants](#) on page 294

[write_vlog_bit_blast_mapped_ports](#) on page 296

[write_vlog_bit_blast_tech_cell](#) on page 298

[write_vlog_convert_onebit_vector_to_scalar](#) on page 301

[write_vlog_declare_wires](#) on page 302

[write_vlog_empty_module_for_black_box](#) on page 304

[write_vlog_empty_module_for_logic_abstract](#) on page 305

[write_vlog_generic_gate_define](#) on page 307

[write_vlog_line_wrap_limit](#) on page 308

[write_vlog_no_negative_index](#) on page 309

[write_vlog_preserve_net_name](#) on page 310

[write_vlog_top_module_first](#) on page 311

[write_vlog_unconnected_port_style](#) on page 312

[write_vlog_wor_wand](#) on page 314

Pin Attributes

Contain information about the specified instance pin. A pin is a single bit connection point on an instance. Pin objects are stored in `pins_in` and `pins_out` directories.

- To set a pin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pin instance/pin]
```

- To get a pin attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pin instance/pin]
```

Note: You may need to specify more than one instance to serve as the search path that will uniquely identify the relevant pin.

hdl_v2001

`hdl_v2001` *string*

Read-write `pin` attribute. Sets the specified Verilog 2001 attributes on this instance pin. The instance can be a primitive, gate, blackbox, logic abstract, or hierarchical instance.

Example

The following command retrieves the Verilog 2001 attributes set on pin `u10/A`.

```
rc:/designs> rc:/> get_attr hdl_v2001 [find / -pin [find / -pin test/u10/A]
{bit_0}]
```

Related Information

Set by this command: [read_netlist -v2001](#)

Related attributes: [\(design\) hdl_v2001](#) on page 323
[\(pgpin\) hdl_v2001](#) on page 329
[\(port\) hdl_v2001](#) on page 336
[\(subdesign\) hdl_v2001](#) on page 342
[\(subport\) hdl_v2001](#) on page 346

Pgpin Attributes

Contain information about power and ground instance pins. Pgpin objects are stored in `pgpins_in` and `pgpins_out` directories.

- To set a pgpin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pgpin instance/pin]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pgpin instance/pin]
```

hdl_v2001

`hdl_v2001` *string*

Read-write pgpin attribute. Sets the specified Verilog 2001 attributes on this instance pgpin. The instance can be a primitive, gate, blackbox, logic abstract, or hierarchical instance.

Related Information

Set by this command:

[read_netlist -v2001](#)

Related attributes:

(design) [hdl_v2001](#) on page 323

((pin) [hdl_v2001](#) on page 328

(port) [hdl_v2001](#) on page 336

(subdesign) [hdl_v2001](#) on page 342

(subport) [hdl_v2001](#) on page 346

hdl_arch Attributes

Contain information about user-defined modules (or entities).

- To set an hdl_arch attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_arch name]
```

- To get a an hdl_arch attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_arch name]
```

encrypted

```
encrypted {false | true}
```

Read-only hdl_arch attribute. Indicates whether the format of the input architecture was encrypted.

Related Information

Related command: [read_hdl](#)

parameters

```
parameters string
```

Read-only hdl_arch attribute. In Verilog, this attribute keeps an ordered list of all parameters of the module represented by the hdl_arch object. In VHDL, this attribute keeps an ordered list of generics of the entity represented by the hdl_arch object.

Note: Do not confuse this attribute with the parameters branch of vdir objects attached to the hdl_object. Under that branch, each parameter is represented by its own hdl_param object.

Related Information

Related attribute: [\(hdl_comp\) parameters on page 154](#)

pins

`pins pin_list`

Read-only `hdl_arch` attribute. Keeps an ordered list of all pins of the Verilog module or VHDL entity represented by the `hdl_arch` object.

Note: Do not confuse this attribute with the pins branch of vdir objects attached to the `hdl_arch` object. Under that branch, each pin is represented by its own `hdl_pin` object.

Related Information

Related attribute: [\(hdl_comp\) pins](#) on page 154

protected

`protected {false | true}`

Default: `false`

Read-only `hdl_arch` attribute. This attribute is set to `true` if the architecture was read in encrypted format and the output netlist will be in encrypted format.

Related Information

[IP Protection in Using Encounter RTL Compiler](#)

Affects this command: [write_hdl](#)
[report datapath](#)

Related attribute: [\(design\) protected](#) on page 324
[\(subdesign\) protected](#) on page 343

start_source_line

`start_source_line integer`

Read-only `hdl_arch` attribute. Returns the start line number in an HDL input file for a module.

Note: This attribute is supported only in the RTL flow.

structural

```
structural {true | false}
```

Read-only hdl_arch attribute. Returns whether the architecture specified in an HDL file is structural.

verilog_macros

```
verilog_macros list
```

Read-only hdl_arch attribute. Returns information about all the `define macros and their values used in this user-defined module using the following Tcl list format:

```
{file_name line_number {macro_name macro_value}}
```

This attribute must be used after the `read_hdl` command, but before the `elaborate` command.

Example

Consider module `test` defined in file `test.v`

```
module test(c);
    `define MACRO4 6
    output c;
    `ifdef MACRO4
        assign c = `MACRO4 ;
    `else
        assign c = 1 ;
    `endif
endmodule // test
```

The following command returns information about the `'define` macro commands used in module `test`. In this case only `MACRO4` was defined and used.

```
rc:/>get_attr verilog_macros /hdl_libraries/default/architectures/test
{test.v 5 {MACRO4 DEFINED}} {test.v 6 {MACRO4 6}}
```

Related Information

Related command: [read_hdl](#)

hdl_config Attributes

Contain information about VHDL configurations.

- To set an hdl_config attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_config name]
```

- To get a an hdl_comp_arch attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_config name]
```

entity

entity string

Read-only hdl_config attribute. Specifies the name of the entity to which this configuration applies.

location

location pathname

Read-only hdl_config attribute. Returns the location of the VHDL source file that contains the specified VHDL configuration.

Related Information

Related attributes: (hdl_pack) location on page 335

hdl_inst Attributes

Contain information about HDL instances.

- To set an hdl_inst attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_inst name]
```

- To get a an hdl_inst attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_inst name]
```

component

component string

Read-only hdl_inst attribute. Returns the instance.

This attribute is supported only in the RTL flow.

hdl_pack Attributes

Contain information about the VHDL packages.

- To get a an hdl_pack attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_pack name]
```

default_location

`default_location pathname`

Read-only hdl_pack attribute. Returns the physical location of the source file that contains the registered VHDL package. Registered VHDL packages are only created through the `hdl_create` package command.

Related Information

Affected by this command: [hdl_create parameter](#)

Related Attributes
(`hdl_config`) [location](#) on page 333
(`hdl_pack`) [location](#) on page 335

location

`location pathname`

Read-only hdl_pack attribute. Returns the name and physical location of the source file that contains the specified VHDL package.

Note: This attribute is supported only in the RTL flow.

Related Information

Related attributes: [default_location](#) on page 335
(`hdl_config`) [location](#) on page 333

Port Attributes

Contain information about single bit connection points in the specified top-level design. Port objects are stored in `ports_in` and `ports_out` directories.

- To set a port attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -port name]
```

- To get a port attribute value, type

```
get_attribute attribute_name [find /des*/design -port name]
```

hdl_v2001

`hdl_v2001` *string*

Read-write `port` attribute. Sets the specified Verilog 2001 attributes on this port.

Example

The following command sets the `TB_KFLAG` attribute to `+ES` on port `PCLK`.

```
set_attr hdl_v2001 {TB_KFLAG="+ES"} [find . port PCLK]
```

Related Information

Set by this command:

[read_netlist -v2001](#)

Related attributes:

(design) [hdl_v2001](#) on page 323

(instance) [hdl_v2001](#) on page 326

(pin) [hdl_v2001](#) on page 328

(pgpin) [hdl_v2001](#) on page 329

(subdesign) [hdl_v2001](#) on page 342

(subport) [hdl_v2001](#) on page 346

Subdesign Attributes

Contain information about the subdesigns in the specified design. Subdesigns correspond to Verilog modules or VHDL entities instantiated in the top-level Verilog module or top-level VHDL entity.

- To set a subdesign attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subdesign name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subdesign name]
```

arch_filename

`arch_filename string`

Read-write subdesign attribute. Returns the name and physical location of the source file that contains the specified architecture (in VHDL) or module (in Verilog).

Note: This attribute is only supported in the RTL flow.

Example

When multiple design files are loaded, it might be difficult to correlate a module or architecture to the file in which it was instantiated. The following example illustrates how to find the VHDL file for a particular submodule `my_block`, using the `get_attribute` command.

```
rc:/> get_attribute arch_filename /designs/top/subdesign/my_block
```

The above command would return something like the following output, showing that the `my_block` submodule was instantiated in the file `block_arch.vhdl`:

```
../rtl/archs/block_arch.vhdl
```

Related Information

[Using the get_attribute Command to Search for Information in Using Encounter RTL Compiler](#)

Related attributes:	(design) <u>arch_filename</u> on page 317
	(design) <u>entity_filename</u> on page 318
	(subdesign) <u>entity_filename</u> on page 338

embedded_script

`embedded_script string`

Read-write subdesign attribute. Stores the embedded SDC command script in the RTL code describing the module or entity of this design or subdesign.

Using the exec embedded script command applies the stored script onto the design or subdesign.

You can use a synthesis script to modify the contents of the `embedded_script` attribute before executing it.

The embedded script can have any SDC command that is supported by the `read_sdc` command.

Related Information

Related attributes: (design) embedded_script on page 318

entity_filename

`entity_filename string`

Read-write subdesign attribute. Returns the name and physical location of the source file that contains the specified VHDL entity. Since Verilog does not have the concept of VHDL entities, this attribute will return the file that contains the Verilog module. For Verilog modules, the `entity_filename` and `arch_filename` attributes behave identically.

Note: This attribute is only supported in the RTL flow.

Example

When multiple entity files are loaded, it might be difficult to correlate an entity to the file in which it was instantiated. The following example illustrates how to find the entity file for a particular submodule `my_block`, using the `get_attribute` command.

```
rc:/> get_attribute entity_filename /designs/top/subdesign/my_block
```

The above command would return something like the following output, showing that the `my_block` submodule was instantiated in the file `block_ent.vhdl`:

```
../rtl/entities/block_ent.vhdl
```

Related Information

- Related attributes:
- (design) [arch_filename](#) on page 317
 - (subdesign) [arch_filename](#) on page 337
 - (design) [entity_filename](#) on page 318

hdl_all_filelist

```
hdl_all_filelist {hdl_library language_standard {define_value ... }  
{hdl_file ...} {search_path}}...
```

Read-write [subdesign](#) attribute. Stores the information required by the `read_hdl` command to elaborate the subdesign. The information is stored as a list of values that can be used as options for the `read_hdl` command. The list of files includes any 'include and package files read into RTL Compiler for the specified design.

- *hdl_library* is the value for the `-library` option.
- *language_standard* corresponds to one of the following: `-v1995`, `-v2001`, `-vhdl1987`, `-vhdl1193`, `-vhdl2008`, or `-sv`.
For vhdl, the value of the `hdl_vhdl_read_version` attribute is appended to `-vhdl`.
- *define_value* is the value for the `-define` option (default: SYNTHESIS)
- *hdl_file* corresponds to the list of files required to elaborate the design.
- *search_path* is the directory of the location of the HDL files. An empty value signifies the current directory.

Since the attribute contains only those files that have been used during elaboration, you can use this attribute to prune unnecessary `read_hdl` commands from your scripts.

Note: This attribute is supported in the RTL flow and the structural flow.

Example

Consider the following RTL files:

File top.v

```
module top(a,b);  
    input [1:0] a;  
    output [1:0] b;  
    bot u1(.x(a),.y(b));...  
endmodule
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Subdesign Attributes

File bot.v

```

`include "siz.h"
`include "inc.h"
module bot(x,y);
    input [ `SIZE:0 ] x;
    output [ `SIZE:0 ] y;
    wire [ `SIZE:0 ] tmp;
    assign y = `ADD(x,tmp);
endmodule

```

File siz.h

```
`define SIZE 1
```

File inc.v

```
`define ADD(a,b) a+b
```

Consider the following script:

```
set_attr library tutorial.lib  
read_hdl top.v  
read_hdl bot.v  
elab top
```

After elaboration, the `hdl_all_filelist` for subdesign bot will return:

```
{default -v1995 {SYNTHESIS} {bot.v siz.h inc.h}}}
```

The difference between `hdl_filelist` and `hdl_all_filelist` is that `hdl_all_filelist` also lists the include files `siz.h` and `inc.h`.

Related Information

Related attributes: (design) [hdl_all_filelist](#) on page 319
(subdesign) [hdl_filelist](#) on page 341

hdl_config_name

hdl_config_name string

Read-write subdesign attribute. Specifies the name of the configuration used to build this design.

Related Information

Related attributes: (design) [hdl_config_name](#) on page 320

hdl_filelist

```
hdl_filelist {hdl_library language_standard {define_value ...} {hdl_file ...} {search_path}}...
```

Read-write subdesign attribute. Stores the information required by the `read_hdl` command to elaborate the subdesign. The information is stored as a list of values that can be used as options for the `read_hdl` command.

- `hdl_library` is the value for the `-library` option.
- `language_standard` corresponds to one of the following: `-v1995`, `-v2001`, `-vhdl1987`, `-vhdl1193`, `-vhdl2008`, or `-sv`.

For vhdl, the value of the `hdl_vhdl_read_version` attribute is appended to `-vhdl`.

- `define_value` is the value for the `-define` option (default: SYNTHESIS)
- `hdl_file` corresponds to the list of files required to elaborate the design.
- `search_path` is the directory of the location of the HDL files. An empty value signifies the current directory.

Since the attribute contains only those files that have been used during elaboration, you can use this attribute to prune unnecessary `read_hdl` commands from your scripts.

Note: This attribute is supported in the RTL flow and the structural flow.

Example

- Consider the following RTL files:

File top.v

```
module top(a,b);
    ...
    mid u1(.c(a),.d(b));
endmodule
```

File mid.v

```
module mid(c,d);
    ...
    bot u2(.x(c),.y(d));
endmodule
```

File bot.v

```
module bot(x,y);
    ...
endmodule
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Subdesign Attributes

File foo.v

```
module foo(q,w);  
...  
endmodule
```

Consider the following script:

```
set_attr library tutorial.lib  
read_hdl top.v mid.v bot.v foo.v  
elab top
```

After elaboration, the `hdl_filelist` for subdesign `mid` will return:

```
{default -v1995 {SYNTHESIS} {mid.v bot.v} {}}
```

Files `top.v` and `foo.v` are not stored because they were not used during elaboration of the subdesign `mid`.

Related Information

- Related attributes: (design) [hdl_filelist](#) on page 321
(subdesign) [hdl_all_filelist](#) on page 339

hdl_user_name

`hdl_user_name` *string*

Read-write `subdesign` attribute. Represents the name of the Verilog module or the VHDL entity from which the given subdesign was derived. The design's name may differ from the `hdl_user_name` value and from the addition of suffixes for the module name uniquification.

For example, a parameterized module named `adder` with `wA` and `wB` parameters may result in a design whose name is `adder_wA5_wB3`. However, the `hdl_user_name` attribute value for this design would still be `adder`.

Related Information

- Related attributes: (design) [hdl_user_name](#) on page 323

hdl_v2001

`hdl_v2001` *string*

Read-write `subdesign` attribute. Sets the specified Verilog 2001 attributes on this subdesign.

Example

The following command sets the `is_macro` attribute to true on subdesign `mytest`.

```
set_attr hdl_v2001 {is_macro="true"} [find . -subdesign mytest]
```

Related Information

Set by this command: [read_netlist -v2001](#)

Related attributes: (design) [hdl_v2001](#) on page 323

(instance) [hdl_v2001](#) on page 326

(pin) [hdl_v2001](#) on page 328

(pgpin) [hdl_v2001](#) on page 329

(port) [hdl_v2001](#) on page 336

(subport) [hdl_v2001](#) on page 346

protected

```
protected {false | true}
```

Default: false

Read-only `subdesign` attribute. This attribute is set to `true` if the subdesign object represents a module or entity whose HDL source code is encrypted or only partially encrypted.

Related Information

[IP Protection in Using Encounter RTL Compiler](#)

Affects this command: [write_hdl](#)

[report_datapath](#)

Related attributes: (design) [protected](#) on page 324

(hdl_arch) [protected](#) on page 331

write_vlog_empty_module_for_subdesign

```
write_vlog_empty_module_for_subdesign {false | true}
```

Default: false

Read-only subdesign attribute. Controls how a subdesign is written out in a Verilog netlist. If set to true, the subdesign is written as an empty module.

Example

Consider the following design :

```
module sub(in, out);
  input in;
  output out;
  assign out = in;
endmodule

module WRTV(in, out);
  input in;
  output [4:0]out;
  sub s0(in, out[4]);
endmodule
```

- Set the `write_vlog_empty_module_for_subdesign` to true to write out the sub subdesign as an empty module:

```
set_attr write_vlog_empty_module_for_subdesign true \
/designs/WRTV/subdesigns/sub

module sub(in, out);
  input in;
  output out;
endmodule

module WRTV(in, out);
  input in;
  output [4:0]out;
  sub s0(in, out[4]);
endmodule
```

- Using the default setting for `write_vlog_empty_module_for_subdesign`, will result in the following netlist::

```
set_attr write_vlog_empty_module_for_subdesign false \
/designs/WRTV/subdesigns/sub

module sub(in, out);
  input in;
  output out;
  wire in;
  wire out;
  assign out = in;
endmodule

module WRTV(in, out);
  input in;
```

Attribute Reference for Encounter RTL Compiler

Input and Output—Subdesign Attributes

```
output [4:0]out;  
sub s0(in, out[4]);  
endmodule
```

Related Information

Affects this command: [write_hdl](#)

Subport Attributes

Contain information about subports in the specified design. A subport is a single bit connection point within a hierarchical instance, that is a module or entity that has been instantiated.

- To set a subport attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subport hier_instance/name]
```

- To get a subport attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subport hier_instance/name]
```

Note: You may need to specify more than one (hierarchical) instance to serve as the search path that will uniquely identify the relevant subport.

hdl_v2001

`hdl_v2001 string`

Read-write subport attribute. Sets the specified Verilog 2001 attributes on this subport.

Example

The following command retrieves the Verilog attributes set on subport A of subdesign u0.

```
rc:/designs/test> get_attr hdl_v2001 [find . -subport u0/A]
{subport_in="A"}
```

Related Information

Set by this command:

[read_netlist -v2001](#)

Related attributes:

(design) [hdl_v2001](#) on page 323

(instance) [hdl_v2001](#) on page 326

(pin) [hdl_v2001](#) on page 328

(pgpin) [hdl_v2001](#) on page 329

(port) [hdl_v2001](#) on page 336

(subdesign) [hdl_v2001](#) on page 342

Attribute Reference for Encounter RTL Compiler

Input and Output—Subport Attributes

Attribute Reference for Encounter RTL Compiler

Input and Output—Subport Attributes

Physical

Root Attributes

- [cap_table_file](#) on page 364
- [congestion_effort](#) on page 364
- [def_output_escape_multibit](#) on page 365
- [def_output_version](#) on page 365
- [enc_assign_buffer](#) on page 366
- [enc_assign_removal](#) on page 366
- [enc_clk_gate_recloning](#) on page 366
- [enc_cpu_usage](#) on page 367
- [enc_force_place_incr](#) on page 368
- [enc_gzip_interface_files](#) on page 368
- [enc_launch_servers](#) on page 369
- [enc_memory_usage](#) on page 370
- [enc_module_plan](#) on page 370
- [enc_postload_script](#) on page 371
- [enc_pre_place_opt](#) on page 371
- [enc_preelexport_script](#) on page 372
- [enc_preload_script](#) on page 372
- [enc_temp_dir](#) on page 373
- [enc_timing_driven_place](#) on page 374
- [enc_user_constraint_file](#) on page 375

Attribute Reference for Encounter RTL Compiler

Physical—List

- [enc_user_mode_file](#) on page 376
- [force_via_resistance](#) on page 376
- [interconnect_mode](#) on page 377
- [lef_manufacturing_grid](#) on page 377
- [lef_stop_on_error](#) on page 377
- [lef_units](#) on page 378
- [lib_lef_consistency_check_enable](#) on page 378
- [phys_annotation_ndr_nets](#) on page 378
- [phys_checkout_encounter_license](#) on page 379
- [phys_fix_multi_height_cells](#) on page 379
- [phys_flow_effort](#) on page 380
- [phys_legalization_enhancement](#) on page 380
- [phys_legalize](#) on page 381
- [phys_mp_constraints](#) on page 381
- [pqos_ignore_msv](#) on page 381
- [pqos_ignore_scan_chains](#) on page 382
- [pqos_placement_effort](#) on page 382
- [phys_pre_place_iopt](#) on page 382
- [phys_premorph_density](#) on page 383
- [qos_report_power](#) on page 383
- [qrc_tech_file](#) on page 384
- [scale_of_cap_per_unit_length](#) on page 384
- [scale_of_res_per_unit_length](#) on page 385
- [shrink_factor](#) on page 385
- [use_area_from_lef](#) on page 386
- [via_resistance](#) on page 386

Design Attributes

- [aspect_ratio](#) on page 387
- [avoid_no_row.libcell](#) on page 387
- [blockages](#) on page 388
- [def_extension](#) on page 389
- [def_file](#) on page 389
- [def_history](#) on page 389
- [def_technology](#) on page 390
- [def_version](#) on page 390
- [die_area](#) on page 391
- [fplan_height](#) on page 391
- [fplan_width](#) on page 391
- [groups](#) on page 392
- [init_core_utilization](#) on page 392
- [number_of_routing_layers](#) on page 393
- [pcells](#) on page 393
- [phys_ignore_nets](#) on page 395
- [phys_ignore_special_nets](#) on page 395
- [physical_aware_mapping](#) on page 395
- [physical_aware_restructuring](#) on page 396
- [physical_aware_structuring](#) on page 396
- [preroute_as_obstruction](#) on page 394
- [regions](#) on page 396
- [sdp_files](#) on page 397
- [sdp_type](#) on page 397
- [utilization](#) on page 397
- [utilization_threshold](#) on page 398

Attribute Reference for Encounter RTL Compiler

Physical—List

Pin Attributes

- [location_x](#) on page 399
- [location_y](#) on page 399
- [physical](#) on page 400
- [physical_del](#) on page 400
- [physical_res](#) on page 400
- [placement_status](#) on page 401

Pgpin Attributes

- [location_x](#) on page 402
- [location_y](#) on page 402
- [physical](#) on page 402
- [placement_status](#) on page 403

Net Attributes

- [physical_cap](#) on page 404

Port Attributes

- [location_x](#) on page 405
- [location_y](#) on page 405
- [physical_del](#) on page 405
- [physical_res](#) on page 406
- [placement_status](#) on page 406

Subdesign Attributes

- [fplan_height](#) on page 407
- [fplan_width](#) on page 407
- [physical_aware_restructuring](#) on page 408

Attribute Reference for Encounter RTL Compiler

Physical—List

- [physical_aware_structuring](#) on page 408

Instance Attributes

- [cx](#) on page 409
- [cy](#) on page 409
- [llx](#) on page 410
- [lly](#) on page 410
- [height](#) on page 411
- [location_x](#) on page 411
- [location_y](#) on page 411
- [placement_status](#) on page 412
- [skip_in_write_def](#) on page 412
- [urx](#) on page 413
- [ury](#) on page 413
- [width](#) on page 413
- [width](#) on page 413

Blockage Attributes

- [boxes](#) on page 414
- [component](#) on page 414
- [density](#) on page 415
- [has_fills](#) on page 415
- [has_slots](#) on page 416
- [is_exceptpgnet](#) on page 416
- [is_partial](#) on page 416
- [is_pushdown](#) on page 417
- [is_soft](#) on page 417

Attribute Reference for Encounter RTL Compiler

Physical—List

- [layer](#) on page 417
- [location_x](#) on page 418
- [location_y](#) on page 418
- [max_layer](#) on page 419
- [min_layer](#) on page 419
- [polygons](#) on page 419
- [spacing](#) on page 420
- [type](#) on page 420
- [user_created](#) on page 420
- [visible](#) on page 421
- [width](#) on page 421

Bump Attributes

- [def_name](#) on page 422
- [height](#) on page 422
- [location_x](#) on page 423
- [location_y](#) on page 423
- [model](#) on page 423
- [orientation](#) on page 424
- [placement_status](#) on page 424
- [properties](#) on page 425
- [urx](#) on page 425
- [ury](#) on page 426
- [weight](#) on page 426
- [width](#) on page 427

Attribute Reference for Encounter RTL Compiler

Physical—List

DEF Pin Attributes

- [direction](#) on page 428
- [layers](#) on page 428
- [location_x](#) on page 429
- [location_y](#) on page 429
- [net_expr](#) on page 429
- [net_name](#) on page 430
- [orientation](#) on page 430
- [placement_status](#) on page 430
- [polygons](#) on page 431
- [ports](#) on page 431
- [special](#) on page 432
- [use](#) on page 432
- [vias](#) on page 433
- [visible](#) on page 433

Fill Attributes

- [boxes](#) on page 434
- [layer](#) on page 434
- [polygons](#) on page 435

Gcell Attributes

- [box](#) on page 437
- [horizontal_remaining](#) on page 437
- [instance_count](#) on page 438
- [instances](#) on page 438
- [pin_count](#) on page 439

Attribute Reference for Encounter RTL Compiler

Physical—List

- [pin_density](#) on page 439
- [pins](#) on page 439
- [utilization](#) on page 440
- [vertical_remaining](#) on page 440

Group Attributes

- [def_name](#) on page 441
- [members](#) on page 441
- [properties](#) on page 442
- [region](#) on page 442
- [user_created](#) on page 443

Layer Attributes

- [cap_multiplier](#) on page 444
- [capacitance](#) on page 444
- [cap_table_name](#) on page 445
- [color](#) on page 445
- [direction](#) on page 445
- [layer_index](#) on page 446
- [min_spacing](#) on page 446
- [offset](#) on page 446
- [offset_x](#) on page 446
- [offset_y](#) on page 446
- [pitch](#) on page 447
- [pitch_x](#) on page 447
- [pitch_y](#) on page 447
- [resistance](#) on page 448

Attribute Reference for Encounter RTL Compiler

Physical—List

- [utilization](#) on page 448
- [visible](#) on page 449
- [width](#) on page 449

Nondefaultrule Attributes

- [from_lef](#) on page 450
- [hardspacing](#) on page 451
- [layers](#) on page 451
- [mincuts](#) on page 452
- [properties](#) on page 452
- [viarules](#) on page 453
- [vias](#) on page 453

Pcell Attributes

- [def_name](#) on page 454
- [height](#) on page 454
- [location_x](#) on page 455
- [location_y](#) on page 455
- [model](#) on page 456
- [orientation](#) on page 456
- [placement_status](#) on page 456
- [properties](#) on page 457
- [urx](#) on page 457
- [ury](#) on page 458
- [weight](#) on page 458
- [width](#) on page 459

Attribute Reference for Encounter RTL Compiler

Physical—List

Pdomain Attributes

- [boundary](#) on page 460
- [boxes](#) on page 460
- [cutouts](#) on page 461
- [mingap](#) on page 461
- [net](#) on page 461
- [rsext](#) on page 462

Pnet Attributes

- [capacitance](#) on page 463
- [components](#) on page 463
- [def_name](#) on page 464
- [fixedbump](#) on page 464
- [frequency](#) on page 464
- [name](#) on page 464
- [nondefaultrule](#) on page 465
- [original_name](#) on page 465
- [path_count](#) on page 465
- [path_index](#) on page 465
- [path_value](#) on page 466
- [pattern](#) on page 466
- [properties](#) on page 466
- [rc_name](#) on page 467
- [shieldnet](#) on page 467
- [source](#) on page 467
- [use](#) on page 468
- [visible](#) on page 468

Attribute Reference for Encounter RTL Compiler

Physical—List

- [weight](#) on page 468
- [xtalk](#) on page 469

Power Domain Attributes

- [box_list](#) on page 470
- [disjoint_hinst_box_list](#) on page 470
- [min_gaps](#) on page 470
- [rs_ext](#) on page 471

Region Attributes

- [boxes](#) on page 472
- [def_name](#) on page 472
- [derived_from_power_domain](#) on page 473
- [group](#) on page 473
- [location_x](#) on page 473
- [location_y](#) on page 473
- [properties](#) on page 474
- [type](#) on page 474
- [user_created](#) on page 474

Row Attributes

- [height](#) on page 475
- [is_horizontal](#) on page 475
- [location_x](#) on page 476
- [location_y](#) on page 476
- [macro](#) on page 477
- [orientation](#) on page 477
- [user_created](#) on page 477

Attribute Reference for Encounter RTL Compiler

Physical—List

- [width](#) on page 478

Site Attributes

- [class](#) on page 479
- [height](#) on page 479
- [symmetry](#) on page 479
- [width](#) on page 480

Slot Attributes

- [boxes](#) on page 481
- [layer](#) on page 481
- [polygons](#) on page 482

Special Net Attributes

- [components](#) on page 483
- [def_name](#) on page 483
- [fixedbump](#) on page 483
- [name](#) on page 484
- [original_name](#) on page 484
- [path_count](#) on page 484
- [path_index](#) on page 485
- [path_value](#) on page 485
- [pattern](#) on page 485
- [polygons](#) on page 486
- [properties](#) on page 486
- [rc_name](#) on page 486
- [rectangles](#) on page 486
- [source](#) on page 487

Attribute Reference for Encounter RTL Compiler

Physical—List

- [style](#) on page 487
- [type](#) on page 488
- [use](#) on page 488
- [voltage](#) on page 489
- [weight](#) on page 489

Style Attributes

- [index](#) on page 490
- [polygon](#) on page 490

Track Attributes

- [count](#) on page 491
- [is_horizontal](#) on page 491
- [is_used](#) on page 492
- [layer](#) on page 492
- [macro](#) on page 492
- [start](#) on page 493
- [step](#) on page 494

Via Attributes

- [bottom_layer](#) on page 495
- [boxes](#) on page 495
- [cut_cols](#) on page 496
- [cut_layer](#) on page 496
- [cut_pattern](#) on page 496
- [cut_rows](#) on page 497
- [polygons](#) on page 497
- [top_layer](#) on page 497

Attribute Reference for Encounter RTL Compiler

Physical—List

- [viarule_name](#) on page 497
- [xbottom_enclosure](#) on page 498
- [xbottom_offset](#) on page 498
- [xcut_size](#) on page 498
- [xcut_spacing](#) on page 499
- [xorigin_offset](#) on page 499
- [xtop_enclosure](#) on page 499
- [xtop_offset](#) on page 500
- [ybottom_enclosure](#) on page 500
- [ybottom_offset](#) on page 500
- [ycut_size](#) on page 501
- [ycut_spacing](#) on page 501
- [yorigin_offset](#) on page 501
- [ytop_enclosure](#) on page 502
- [ytop_offset](#) on page 502

SDP Column Attributes

- [flip](#) on page 503
- [index](#) on page 503
- [justify_by](#) on page 504
- [orient](#) on page 504
- [size_same](#) on page 504
- [skip_value](#) on page 505

SDP Group Attributes

- [hier_path](#) on page 507
- [orient](#) on page 507

Attribute Reference for Encounter RTL Compiler

Physical—List

- [origin](#) on page 508

SDP Instance Attributes

- [flip](#) on page 509
- [index](#) on page 509
- [instance](#) on page 510
- [justify_by](#) on page 510
- [orient](#) on page 510
- [size_fixed](#) on page 511
- [skip_value](#) on page 511

SDP Row Attributes

- [flip](#) on page 513
- [index](#) on page 513
- [justify_by](#) on page 514
- [orient](#) on page 514
- [size_same](#) on page 514
- [skip_value](#) on page 515

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

cap_table_file

```
cap_table_file capacitance_table_file
```

Read-write `root` attribute. Specifies the capacitance table file for technology mapping. You can specify only one file. This file can be encrypted. RTL Compiler uses the capacitance table to derive the capacitance unit per length, resistance unit per length, and area unit per length.

The LEF file can contain the same information but in all cases, RTL Compiler will use the latest specification.

Note: You must specify the LEF files before the capacitance table file.

Related Information

Affects this command:

[synthesize -to_placed](#)

Related attributes:

[lef_library](#) on page 272

[scale_of_cap_per_unit_length](#) on page 384

[scale_of_res_per_unit_length](#) on page 385

congestion_effort

```
congestion_effort {off | low | medium | high}
```

Default: off

Read-write `root` attribute. Specifies the effort that the incremental optimization engine should use to optimize congestion. By default, congestion is not taken into account.

Related Information

Affects this command: [synthesize -incremental](#)

def_output_escape_multibit

`def_output_escape_multibit {true | false}`

Default: true

Read-write [root](#) attribute. Specifies whether to add an escape character (\) before the brackets in component names and before bit blasted ports and nets.

Example

By default, `reg_out[1]` is written as `reg_out\[1\]`.

def_output_version

`def_output_version string`

Default: 5.7

Read-write [root](#) attribute. Specifies the DEF version to use for DEF export. Supported versions are 5.3 and up. The default for the attribute is the latest supported version.

Related Information

Affects these commands: [synthesize -to_placed](#)
[write_def](#)
[write_design](#)
[write_encounter design](#)

enc_assign_buffer

`enc_assign_buffer string`

Read-write root attribute. Specifies the buffer to use for assign removal in the Encounter® place and route tool. If you set the attribute value to `auto`, the tool will automatically select the buffer type. If you set the attribute value to `none`, the tool will use virtual buffers.

Specify this attribute before you issue the `synthesize -to_placed` command.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_assign_removal

`enc_assign_removal {false | true}`

Default: `false`

Read-write root attribute. Specifies whether to perform assign removal in the Encounter® place and route tool.

Specify this attribute before you issue the `synthesize -to_placed` command.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_clk_gate_recloning

`enc_clk_gate_recloning {false | true}`

Default: `false`

Read-write root attribute. Specifies whether to perform clock-gating recloning in the Encounter® place and route tool (that is, use the `-clkGateRecloning` option of the Encounter® `placeDesign` command during placement).

Specify this attribute before you issue the `synthesize -to_placed` command.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_cpu_usage

`enc_cpu_usage integer`

Default: no_value

Read-write root attribute. Specifies the number of CPUs to be used by the Encounter® place and route tool.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_design_mode_process

`enc_design_mode_process integer`

Default: no_value

Read-write root attribute. Specifies the value for the `setDesignMode -process` command of the Encounter® place and route tool. This command option specifies the process technology value (in nanometers). Valid values are in the range of 10 to 250.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_enable_useful_skew

`enc_enable_useful_skew {false | true}`

Default: false

Read-write root attribute. Controls whether to use the useful skew information—dumped by the Encounter® place and route tool in the `latency.sdc` file—during placement optimization in RTL Compiler.

By default, the useful skew is not used.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_force_place_incr

`enc_force_place_incr {true | false}`

Default: true

Read-write root attribute. Specifies whether to perform incremental placement after EDI-based optimization in the RC-P flow.

Specify this attribute before you issue the `synthesize -to_placed` command.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_gzip_interface_files

`enc_gzip_interface_files {true | false}`

Default: true

Read-write root attribute. Compresses the Encounter® interface files (for example, .def, .v, .g, and so on.) generated by the `synthesize -to_placed` command in gzip format. These files will remain compressed when you write them out with `write_encounter design`.

Example

The following example compresses the Encounter interface files during `synthesize -to_placed`:

```
rc:/> set_attribute enc_gzip_interface_files true
...
elaborate
...
synthesize -to_placed
...
```

You can also keep these files uncompressed during `synthesize -to_placed` and only compress them when they are written out using the `-gzip_files` option of the `write_encounter design` command.

Attribute Reference for Encounter RTL Compiler

Physical—Root Attributes

Related Information

Affects these commands: [synthesize -to_placed](#)
[write_encounter design](#)

enc_launch_servers

`enc_launch_servers string`

Read-write [root](#) attribute. Specifies a list of servers that can be used to launch an Encounter® batch process. The tool first tries to launch the batch job on the first specified server. If this attempt fails, the tool will try to launch on the next server, and so on.

The batch job is used to run placement in the physical flow.

The process uses the same controls as for RTL Compiler's super-threading capability and has the same license requirements as super-threading.

Note: You can use other servers than those specified with the [super_thread_servers](#) attribute.

Example

The following command specifies to run the placement job on servers linux21 and linux24.

```
set_attr enc_launch_servers {linux21 linux24} /
```

Related Information

[Super-threading in Using Encounter RTL Compiler](#)

Affects this command: [synthesize -to_placed](#)
Related attributes: [super_thread_batch_command](#) on page 122
 [super_thread_kill_command](#) on page 125
 [super_thread_status_command](#) on page 128

enc_memory_usage

`enc_memory_usage float`

Read-only root attribute. Specifies the total peak memory usage by the Encounter® place and route tool during `synthesize -to_place` and `synthesize -spatial`.

Example

- The following example checks the peak memory usage before synthesis is run.

```
get_attribute enc_memory_usage  
no_value
```

- The following example checks the peak memory usage after the placement run.

```
rc:/> synthesize -spatial  
...  
rc:/> get_attr enc_memory_usage /  
322.445312
```

Related Information

Affected by this command: [synthesize -spatial](#)
[synthesize -to_placed](#)

Affects this command: [report qor](#)

enc_module_plan

`enc_module_plan {true | false}`

Default: true

Read-write root attribute. Specifies whether to use the `-modulePlan` option of the Encounter® `placeDesign` command when generating the prototype placement.

Specify this attribute before you issue the `synthesize -to_placed` command.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_postload_script

`enc_postload_script string`

Read-write root attribute. Specifies the script to include in the Encounter setup file after the setup steps (after the libraries, design, user constraints, and user modes are loaded) and before placement is started.

The script will be sourced after the `rc2enc.enc_setup.tcl` file.

Example

Following is an example of such a script:

```
timeDesign -prePlace outDir ./ zwlm
```

Related Information

Affects this command: [synthesize -to_placed](#)

enc_pre_place_opt

`enc_pre_place_opt {true | false}`

Default: true

Read-write root attribute. Specifies whether to perform buffer and inverter pair removal during placement. This pre-placement optimization occurs when using the `synthesize -to_placed` command.

Specify this attribute before you issue the `synthesize -to_placed` command.

Example

The following example turns on pre-placement optimization:

```
...
rc:/> set_attribute enc_pre_place_opt true
rc:/> synthesize -to_placed
...
```

Related Information

Affects this command: [synthesize -to_placed](#)

enc_preexport_script

`enc_preexport_script string`

Read-write root attribute. Specifies the script to be sourced in an Encounter batch file prior to data export (that is after placement is completed, but before leaving Encounter).

Note: When sourcing this script, the tool will check out a license for the Encounter Digital Implementation System and all its relevant options for the specific flow.

Do not use this attribute in conjunction with the phys_flow_effort root attribute.

Example

Following is an example of such a script:

```
set_global report_timing_format {hpin cell fanout load slew delay arrival}
set_table_style -name report_timing -max_widths {256}
report_timing
```

Related Information

Affects this command:

synthesize -to_placed

enc_preload_script

`enc_preload_script string`

Read-write root attribute. Specifies the script to include in the Encounter setup file prior to the setup steps (before the libraries and the design are loaded).

Example

Following is an example of such a script:

```
set_global report_timing_format {hpin cell fanout load slew delay arrival}
set_table_style -name report_timing -max_widths {256}
suppressMessage SOCLF 200
```

Related Information

Affects this command:

synthesize -to_placed

enc_save_db

`enc_save_db {false| true}`

Default: false

Read-write root attribute. Enables saving of the Encounter database during the RC-physical flow.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_scan_def_file

`enc_scan_def_file string`

Read-write root attribute. Specifies the scanDEF file to be used in Encounter. If you set this attribute, RTL Compiler does not write out a scanDEF file.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_temp_dir

`enc_temp_dir string`

Read-write root attribute. Specifies the directory in which the Encounter® interface files (such as, .conf, .def, .sdc, netlist, and so on) generated during `synthesize -to_placed` must be stored.

Files with the `rc2enc` prefix can be used to transfer data from RTL Compiler to the Encounter® place and route tool. Files with the `enc2rc` prefix can be used to transfer data from the Encounter place and route tool to RTL Compiler.

The directory also contains setup files to reload the design in the Encounter tool and RTL Compiler. The setup files allow the user to reload the state of the design before and after placement.

`rc2enc.rc_setup.tcl` — reloads the design in RTL Compiler before placement.

Attribute Reference for Encounter RTL Compiler

Physical—Root Attributes

`rc2enc.enc_setup.tcl`—reloads the design in Encounter place and route before placement.

`enc2rc.rc_setup.tcl`—reloads the design in RTL Compiler after placement.

`enc2rc.enc_setup.tcl`—reloads the design in Encounter place and route after placement.

Note: When you do not set this attribute, the directory is deleted after the `synthesize -to_placed` command completes.

Example

The following command requests to store the files in the `rc_phys_data` directory.

```
set_attribute enc_temp_dir rc_phys_data /  
synthesize -to_placed
```

Related Information

Affects this command: [synthesize -to_placed](#)

Related attribute: [enc_gzip_interface_files](#) on page 368

enc_timing_driven_place

`enc_timing_driven_place {true | false}`

Default: true

Read-write `root` attribute. Specifies whether to activate Encounter's timing-driven placement to generate the silicon virtual prototype when using the `synthesize -to_placed` command. Timing-driven placement balances the importance of meeting setup timing constraints with routability, resulting in placement that is better suited for timing closure.

Specify this attribute before you issue the `synthesize -to_placed` command.

Note: Using this attribute can cause an increase in run-time.

Related Information

Affects this command: [synthesize -to_placed](#)

enc_user_constraint_file

`enc_user_constraint_file string`

Read-write root attribute. Specifies an additional constraint file to be used in an Encounter® session. Include any commands that you want to execute in an Encounter session in this file. The file is simply sourced during the Encounter session.

Example

The following command reads the `viewDefinition.tcl` file containing the commands to be executed in the Encounter session.

```
set_attribute enc_user_constraint_file viewDefinition.tcl /
```

The `viewDefinition.tcl` file might look like:

```
create_library_set -name timelib_max\  
    -timing\  
        [list ... ]  
create_rc_corner -name rc_max\  
    -cap_table /xxxxx/abc/xyz_AP_Worst.captable \  
    -res_factor 0.95\  
    -default_cap_factor 1.0\  
    -detailed_cap_factor 1.157\  
    -xcap_factor 1\  
    -detailed_clock_cap_factor 0.816\  
    -T 25\  
    -qx_tech_file /xxxxx/abc/xyz/qrcTechFile  
create_delay_corner -name delay_max\  
    -library_set timelib_max\  
    -opcond_library CORE65LPLVT\  
    -opcond wc_1.10V_125C\  
    -rc_corner rc_max  
create_constraint_mode -name prects\  
    -sdc_files\  
        [list ... ]  
create_constraint_mode -name postcts\  
    -sdc_files\  
        [list ... ]  
create_analysis_view -name setup_prects -constraint_mode prects  
    -delay_corner delay_max  
create_analysis_view -name setup_postcts -constraint_mode postcts  
    -delay_corner delay_max  
set_analysis_view -setup [list setup_prects] -hold [list setup_prects]  
set_timing_derate -delay_corner delay_max -early 0.95 -clock  
set_timing_derate -delay_corner delay_max -late 1.05 -clock
```

Related Information

Affects this command: [synthesize -to_placed](#)

Attribute Reference for Encounter RTL Compiler

Physical—Root Attributes

enc_user_mode_file

`enc_user_mode_file string`

Read-write root attribute. Specifies an additional mode file to include in Encounter® data set.

Related Information

Affects this command: [synthesize -to_placed](#)

force_via_resistance

`force_via_resistance float`

Default: no_value

Read-write root attribute. When specified, overrides the via resistance value read from the capacitance table file or the LEF library. Specify the new value in ohm. A warning is given when the specified value is set too high or too low. For example, you can use this attribute to account for double via routing.

Example

```
rc:/> get_attr via_resistance /  
4.0  
rc:/> set_attr force_via_resistance 8 /  
Forced via resistance is too large. Single via resistance is '4.000'.  
Setting attribute of root '/': 'force_via_resistance' = 8.0  
rc:/> set_attr force_via_resistance 1.5 /  
Forced via resistance is too small. Single via resistance is '4.000'.  
Setting attribute of root '/': 'force_via_resistance' = 1.5  
rc:/> set_attr force_via_resistance 2.5 /  
Setting attribute of root '/': 'force_via_resistance' = 2.5  
rc:/> set_attr force_via_resistance 5.5 /  
Setting attribute of root '/': 'force_via_resistance' = 5.5
```

Related Information

Related attributes: [cap_table_file](#) on page 364
[lef_library](#) on page 272
[via_resistance](#) on page 386

interconnect_mode

```
interconnect_mode {wireload | ple}
```

Default: wireload

Read-write root attribute. Determines whether RTL Compiler should use the wire-load models or physical layout estimators (PLEs) during synthesis.

When you read in LEF libraries, the `interconnect_mode` attribute is automatically set to `ple`. In this case, RTL Compiler will use the physical information from the LEF and capacitance table file during synthesis instead of the wire-load model from the technology library.

In `ple` mode the cell area defined in the LEF will be used in place of those in the timing library (`.lib`) area. The timing library area will be used if the physical libraries do not contain any cell definitions.

lef_manufacturing_grid

```
lef_manufacturing_grid float
```

Read-only root attribute. Returns the MANUFACTURINGGRID value defined in the LEF library in microns.

Related Information

Related attribute: [lef_library on page 272](#)

lef_stop_on_error

```
lef_stop_on_error {false | true}
```

Default: false

Read-write root attribute. Specifies whether the tool should stop when it encounters an error in the LEF file. By default, the tool ignores the LEF parse errors.

Related Information

Related attribute: [lef_library on page 272](#)

lef_units

`lef_units integer`

Read-only root attribute. Returns the units specified in the LEF file.

Related Information

Related attribute: [lef_library](#) on page 272

lib_lef_consistency_check_enable

`lib_lef_consistency_check_enable {true | false}`

Default: true

Read-write root attribute. Controls the consistency checking between the cells defined in the LEF library and the timing library. If there is at least one MACRO definition in the LEF file, RTL Compiler checks if all the cells in the timing library have a corresponding definition in the LEF library. Any cells that are defined in the timing library but not in the LEF will be marked as `avoid` (they will not be used during synthesis) and a warning message will be issued. To turn off the consistency checking, set this attribute to `false` before you read the LEF libraries.

Related Information

Affects this attribute: [lef_library](#) on page 272

phys_annotation_ndr_nets

`phys_annotation_ndr_nets {true | false}`

Default: true

Read-write root attribute. Annotates SPEF data for non-default routes (NDRs) and for nets assigned to higher metal layers. Using the SPEF data improves correlation with the EDI System and improves QoS by focusing on the correct timing paths.

RC Physical does not natively use higher metal layers or NDRs for route estimation. By using EDI SPEF, RC Physical can accurately model the R/C for these special routes and ensure good correlation.

Attribute Reference for Encounter RTL Compiler

Physical—Root Attributes

Note: If you want to enable NDRs for optimization, you need to specify the appropriate settings in the script passed to EDI using the `enc_postload_script` root attribute.

Specify this attribute before you issue the `synthesize -to_placed` command.

Related Information

Affects this command: [synthesize -to_placed](#)

phys_checkout_encounter_license

`phys_checkout_encounter_license {false | true}`

Default: false

Read-write root attribute. Controls whether the Encounter Digital Implementation System can check out the appropriate Encounter license(s) for the requested features. By default, RC Physical uses the basic placement capabilities available in EDI without using any Encounter license. Set this attribute to true to force the EDI System to check out the appropriate Encounter license(s) needed for the features that you use in your script. You will get an error if you do not have access to the required licenses.

Related Information

Affects this command: [synthesize -to_placed](#)

phys_fix_multi_height_cells

`phys_fix_multi_height_cells {false | true}`

Default: false

Read-write root attribute. Controls whether to consider the placement of instances of multiple-height cells (multiple of the standard cell height) fixed during incremental placement.

Multiple-height cells commonly have routing blockages on a particular metal layer in the LEF library. These metal layers usually have power routes.

Related Information

Affects this command: [synthesize -to_placed](#)

phys_flow_effort

```
phys_flow_effort {medium| high | none}
```

Default: medium

Read-write root attribute. Controls the placement and optimization steps in the RC Physical flow.

- High effort provides the best QoR at the cost of runtime.
- Medium effort offers the best trade-off between the runtime and QoR and turns legalization off by default.
- None will result in the best runtime .

Note: This flow does not require an EDI license.

Related Information

Affects this command: synthesize -to_placed

phys_legalization_enhancement

```
phys_legalization_enhancement {auto | false | true}
```

Default: auto

Read-write root attribute. Enables enhancement in legalization after global placement. This helps to preserve relative placement of cells on critical paths and improves timing and wire length. This is useful for timing-critical designs, or designs with double or triple height cells. This attribute can have the following values:

auto	Automatically turns on (off) enhancement in legalization after global placement if (no) double or triple height cells are detected in the design.
false	Turns off the enhancement in legalization after global placement.
true	Turns on the enhancement in legalization after global placement.

Related Information

Affects this command: synthesize -to_placed

phys_legalize

```
phys_legalize {auto | true | false}
```

Default: auto

Read-write root attribute. Controls legalization during physical optimization. This attribute can have the following values:

auto	Automatically turns on legalization during physical optimization if you have set the <code>phys_flow_effort</code> root attribute to <code>high</code> .
false	Turns off the legalization during physical optimization.
true	Turns on the legalization during physical optimization.

phys_mp_constraints

```
phys_mp_constraints string
```

Read-write root attribute. Specifies the name of the Automatic Floorplan Synthesis constraint file that can be used the Encounter `planDesign` command to guide the floorplan estimation.

Related Information

Affects this command: [synthesize -to_placed](#)

pqos_ignore_msv

```
pqos_ignore_msv {false | true}
```

Default: false

Read-write root attribute. Specifies whether to pass library or power domain information to the Encounter place and route tool.

Related Information

Affects this command: [synthesize -to_placed](#)

pqos_ignore_scan_chains

```
pqos_ignore_scan_chains {false | true}
```

Default: false

Read-write root attribute. Specifies whether to ignore scan chains during placement estimation. Set this to true to disable the placement-based reordering in the Encounter® place and route tool.

Related Information

Affects this command: [synthesize -to_placed](#)

pqos_placement_effort

```
pqos_placement_effort {no_value | low | medium | high}
```

Default: no_value

Read-write root attribute. Specifies the effort level to be used for congestion optimization during placement in Encounter® place and route. Sets the value of the `-congEffort` option of the Encounter® `setPlaceMode` command.

Related Information

Affects this command: [synthesize -to_placed](#)

phys_pre_place_iopt

```
phys_pre_place_iopt {auto | true | false}
```

Default: auto

Read-write root attribute. Controls whether to run fast parallel optimization before placement in the physical flow. This attribute can have the following values:

auto	Automatically runs fast parallel optimization before placement during <code>synthesize -to placed</code> .
false	Prevents fast parallel optimization before placement.
true	Runs fast parallel optimization before placement.

Attribute Reference for Encounter RTL Compiler

Physical—Root Attributes

Related Information

Affects this command: [synthesize -to_placed](#)

phys_premorph_density

`phys_premorph_density float`

Default: 0.96

Read-write [root](#) attribute. Specifies the target maximum density during the placement spreading step before legalization. Set this to a lower number for designs with large timing jumps during legalization. The recommended value is between 0.85 and 0.95.

Related Information

Affects this command: [synthesize -to_placed](#)

qos_report_power

`qos_report_power {false | true}`

Default: false

Read-write [root](#) attribute. Specifies whether to include leakage and dynamic power in QoS statistics table.

Related Information

Affects these commands: [generate_reports](#)
 [summary_table](#)

qrc_tech_file

`qrc_tech_file string`

Read-write root attribute. Specifies the QRC technology file from where the process and extraction information must be read.

If you read in both a QRC technology file and a capacitance table file, the QRC technology file will take precedence.

You must read in the LEF files before you can read in the QRC technology file.

Note: For technologies below 28nm, the Encounter® Digital Implementation System requires a QRC technology file instead of a capacitance table file.

Related Information

Affects this command: [synthesize](#)

Related attribute: [cap_table_file](#)

scale_of_cap_per_unit_length

`scale_of_cap_per_unit_length float`

Default: 1.0

Read-write root attribute. Specifies the scale for the wire capacitance value. This attribute is used as multiplier to modify the capacitance values to resolve minor discrepancies between the default, detail, and sign-off extractors in the Encounter® place and route tool.

Example

The following example sets the scale factor to 1.2:

```
rc:/> set_attribute scale_of_cap_per_unit_length 1.2
Setting attribute of root '/': 'scale_of_cap_per_unit_length' = 1.2
```

Related Attributes

Related attribute: [scale_of_res_per_unit_length](#) on page 385

scale_of_res_per_unit_length

`scale_of_res_per_unit_length float`

Default: 1.0

Read-write root attribute. Specifies the scale for the wire resistance value. This attribute is used as multiplier to modify the resistance values to resolve minor discrepancies between the default, detail, and sign-off extractors in the Encounter® place and route tool.

Example

The following example sets the scale factor to 1.2:

```
rc:/> set_attribute scale_of_res_per_unit_length 1.2
Setting attribute of root '/' : 'scale_of_res_per_unit_length' = 1.2
```

Related Information

Related attribute:

[scale_of_cap_per_unit_length](#) on page 384

shrink_factor

`shrink_factor float`

Default: 1.0

Read-write root attribute. Specifies, as a floating point number less than 1.0, how much to shrink the LEF geometries for timing purposes. Geometries in a LEF and DEF can be defined much larger than what will actually be scribed on the silicon. In such cases, a process shrink factor is used.

The shrink factor affects the layer widths. Since the layer widths are used to pick the resistance and capacitance values from the captable, this attribute affect the resistance and capacitance calculation.

Example

The following example sets the shrink factor to 65%. Values above 1.0 are not valid.

```
rc:/> set_attribute shrink_factor 0.65
Setting attribute of root '/' : 'shrink_factor' = 0.65
```

Related Information

Affected by this attribute [shrink_factor](#)

Affects this command: [report area](#)

use_area_from_lef

`use_area_from_lef { false | true | 0 | 1 | auto}`

Default: `false`

Read-write [root](#) attribute. Specifies whether to use the cell area from the LEF libraries. If you set this attribute to `auto`, the cell area will be read from the technology area unless the LEF library is available and it has at least one macro definition. By default, the cell area will not be read from the LEF library.

Note: `0` and `false` are equivalent, and `1` and `true` are equivalent.

Related Information

Affects this command: [synthesize -to_placed](#)

via_resistance

`via_resistance float`

Read-only [root](#) attribute. Returns the average resistance of vias between metal1 and metal2 layers and between metal2 and metal3 layers. This is a *computed* attribute.

Related Information

Related attributes: [cap_table_file](#) on page 364

[lef_library](#) on page 272

[force_via_resistance](#) on page 376

Design Attributes

Contain information about the specified design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design
```

or

```
set_attribute attribute_name attribute_value [find /des* -design name ]
```

- To get a design attribute value, type

```
get_attribute attribute_name /designs/design
```

aspect_ratio

```
aspect_ratio float
```

Default: 1.0

Read-write design attribute. Specifies the aspect ratio for the physical layout estimator (PLE). The ratio is specified as height divided by width.

Example

The following example specifies a aspect ratio with a height of 2 and a width of 3:

```
rc:/> set_attribute aspect_ratio .667 /designs/top
```

avoid_no_row_libcell

```
avoid_no_row_libcell {false| true}
```

Default: false

Read-write design attribute. Prevents the use of libcells for which no rows have been defined in the DEF file.

A libcell site must match that of a defined row, otherwise the cell is unplaceable.

Related Information

Affects this command: [synthesize](#)

Related command: [read_def](#)

blockages

`blockages string`

Read-only design attribute. Returns information about all blockages specified for the design in the DEF file.

Example

```
rc:/designs/rct> get_attr blockages
{1_placement_halo placement_halo {} {} {{103.1 7.8} {105.9 16.4}} {} {slots 0 fills
0 pushdown 0 exceptpgnet 0 density 0.0 spacing 0.0 width 0.0}} {1_placement
placement {} {} {{300.6 151.0} {305.6 152.0}} {} {slots 0 fills 0 pushdown 0
exceptpgnet 0 density 0.0 spacing 0.0 width 0.0}} {2_placement placement {} g3
{{299.2 152.0} {299.4 152.4}} {} {slots 0 fills 0 pushdown 0 exceptpgnet 0 density
0.0 spacing 0.0 width 0.0}} {3_routing_metal1 routing metall1 {} {{266.4 150.0}
{267.4 153.5}} {} {slots 0 fills 0 pushdown 0 exceptpgnet 0 density 0.0 spacing 0.0
width 0.0}} {4_routing_metal1 routing metall1 g5 {{266.4 152.0} {267.4 153.0}} {} {slots
0 fills 0 pushdown 0 exceptpgnet 0 density 0.0 spacing 0.0 width 0.0}}
```

Related Information

Set by this command:

[read_def](#)

Related attributes:

[Blockage Attributes](#) on page 414

def_component_mask_shift

`def_component_mask_shift string`

Read-only design attribute. Returns the imported DEF component mask shift information. The COMPONENTMASKSHIFT statement in the DEF file defines which layers of a component are allowed to be shifted from the original mask colors in the LEF. This can be useful to shift all the layers of a specific component in order to align the masks with other component or router mask settings to increase routing density. This definition allows a specific component to compactly describe the mask shifting for that component.

Related Information

Set by this command:

[read_def](#)

def_extension

```
def_extension string...
```

Read-only design attribute. Returns one or more strings each containing the tag and the extension specified between the BEGINEXT and ENDEXT statements in the imported DEF file.

Example

Assume the DEF file contains the following statements:

```
BEGINEXT "tag_ext"  
  FOO string  
  BAR time  
ENDEXT
```

The attribute will return the following value:

```
rc:/> get_attr def_extension /designs/rct  
{ "tag_ext"  
  FOO string  
  BAR time  
ENDEXT}
```

Related Information

Set by this command: [read_def](#)

def_file

```
def_file def_filename
```

Read-only design attribute. Returns the name of the loaded DEF file.

Related Information

Set by this command: [read_def](#)

def_history

```
def_history string
```

Read-only design attribute. Returns one or more strings each containing a historical record (content of a HISTORY statement) specified in the imported DEF file.

Attribute Reference for Encounter RTL Compiler

Physical—Design Attributes

Example

Assume the DEF file contains the following statements:

```
HISTORY this is history line 1 ;
HISTORY this is history line 2 ;
```

The attribute will return the following value:

```
rc:/> get_attr def_history /designs/rct
{ this is history line 1 } { this is history line 2 }
```

Related Information

Set by this command: [read_def](#)

def_technology

```
def_technology string
```

Read-only design attribute. Returns the technology name specified in the TECHNOLOGY statement of the imported DEF file.

Example

Assume the DEF file contains the following statement:

```
TECHNOLOGY my_tech;
```

The attribute will return the following value:

```
rc:/> get_attr def_technology /designs/rct
my_tech
```

Related Information

Set by this command: [read_def](#)

def_version

```
def_version string
```

Read-only design attribute. Returns the version of the loaded DEF file.

Example

```
rc:/> get_attr def_version /designs/rct
5.7
```

Related Information

Set by this command: [read_def](#)

die_area

`die_area float`

Read-only [design](#) attribute. Returns the die area, in square microns.

Related Information

Set by this command: [read_def](#)

fplan_height

`fplan_height float`

Read-only [design](#) attribute. Returns the height, in microns, of the physical block for a design.

Related Information

Related attributes: [fplan_width](#) on page 391

fplan_width

`fplan_width float`

Read-only [design](#) attribute. Returns the width, in microns, of the physical block for a design.

Related Information

Related attribute: [fplan_height](#) on page 391

groups

groups string

Read-only design attribute. Returns information such as the members, the region, and properties for all groups defined for the design in the DEF file.

Example

```
rc:/designs/rct> get_attr groups
{GROUP_ONE {U1/g365 U1/g5} REGION_ONE {{grpPropString {group string}}
{grpPropInteger 12} {grpPropReal 1.23}}} {GROUP_TWO {U1/g20 U1/g21} {} {}}
{GROUP_THREE add_23_3* REGION_TWO {{grpPropInteger 42}}}
```

Related Information

Set by this command: [read_def](#)

Related attributes: [Group Attributes](#) on page 441

init_core_utilization

init_core_utilization float

Default: no_value

Read-write design attribute. Specifies the initial utilization to be used by Encounter® MasterPlan to create a floorplan if no floorplan was read in.

During automatic floorplan generation, the die size (based on specified initial utilization) is calculated using the area of both standard cells and hard macros.

Note: After the floorplan is created, the `get_attr utilization` command will return the core utilization value. Because the core area does not include the hard macro area, a mismatch between the returned utilization and the specified initial utilization is expected.

Related Information

Related commands: [synthesize -to_placed](#)
[synthesize -spatial](#)

number_of_routing_layers

`number_of_routing_layers integer`

Read-write design attribute. Limits the number of routing layers that will be used to calculate the area, capacitance, and resistance per unit length. The attribute has no default value, which indicates that all routing layers will be used.

Related Information

Related attributes: [cap_table_file](#) on page 364
[lef_library](#) on page 272

obstruction_routing_layer

`obstruction_routing_layer integer`

Default: 2

Read-write design attribute. Specifies the maximum (top) layer index that the tool should look at to derive (create) placement blockages from existing routing blockages.

Note: The layers are defined in the LEF file in process order from bottom to top.

Related Information

Related attributes: [cap_table_file](#) on page 364
[lef_library](#) on page 272

pcells

`pcells string`

Read-only design attribute. Returns physical cell (pcell) information. A physical cell is a cell that is instantiated in the DEF COMPONENTS section but is not instantiated in the netlist. These cells have neither logical functions or timing arcs. This type of cell is often used to fill gaps in a power grid. Examples include I/O pad corner cells, I/O power supply pads, filler cells, and decoupling caps.

The pcell attribute returns the following physical cell data: The cell name (instance name), macro name (library cell name), placement status (unplaced, placed and moveable or placed and not movable), origin (placement location), orientation (how the cell is flipped and rotated),

Attribute Reference for Encounter RTL Compiler

Physical—Design Attributes

halo, and properties. The halo specifies a placement blockage that is tied to the instance and will therefore move with the instance. The halo is specified relative to the boundary of the cell starting from left, bottom, right and top, in that order. The properties come from the DEF mechanism to track user defined instance properties.

The output is of the form:

```
{cellname macroname placement_status {origin} orientation {halo} {properties}}
```

Example

The following example shows the physical data for the following information in the design called test:

```
cell name: U1/foo
macro name: ANALOG_MOD.
placement status: fixed
origin: {12.600 24.000}
orientation: East
halo: 0.200 0.200 0.200 0.200
properties: NETLIST
rc:/designs/> get_attribute pcells test
{U1/fool ANALOG_MOD fixed {12.600 24.000} E {0.200 0.200 0.200 0.200} NETLIST}
```

Related Information

Related command: [read_def](#)

preroute_as_obstruction

```
preroute_as_obstruction string
```

Read-write [design](#) attribute. Specifies to create placement blockages for the prerouted nets on the layers identified by the specified layer indexes. You must list all layer indexes individually, no ranges are allowed.

Example

Consider the following attribute setting:

```
set_attribute preroute_as_obstruction 1 2 4 /designs/mydesign
```

This will create placement blockages for any prerouted nets on layers M1, M2 and M4.

Related Information

Related command: [read_def](#)

phys_ignore_nets

```
phys_ignore_nets {false | true}
```

Default: false

Read-write design attribute. Controls processing of the NETS section in the DEF file by the `read_def` or `write_def` commands. By default, the NETS section is processed.

Related Information

Affects these commands: [read_def](#)
[write_def](#)

phys_ignore_special_nets

```
phys_ignore_special_nets {false | true}
```

Default: false

Read-write design attribute. Controls processing of the SPECIALNETS section in the DEF file by the `read_def` or `write_def` commands.

Related Information

Affects these commands: [read_def](#)
[write_def](#)

physical_aware_mapping

```
physical_aware_mapping {false | true}
```

Default: false

Read-write design attribute. Enables physical-aware global mapping. Cluster placement is performed during target-driven global mapping to get accurate long wire prediction. This influences datapath structuring and cell selection for convergent timing closure.

Related Information

Affects this command: [synthesize -to_mapped](#)

physical_aware_restructuring

`physical_aware_restructuring {false | true}`

Default: false

Read-write design attribute. Enables physical-aware restructuring after detailed global placement in the RC-Physical flow. Logic trees are re-optimized and balanced with physical context, and concurrently placed to optimize for timing and wire length.

Related Information

Affects this command: [synthesize -to_placed](#)

Related attribute: (subdesign) [physical_aware_restructuring](#) on page 408

physical_aware_structuring

`physical_aware_structuring {false | true}`

Default: false

Read-write design attribute. Enables physical-aware structuring (PAS) on an RTL design. PAS optimizes for congestion and is timing-aware. The tool makes a tradeoff between area and congestion. The logic gate area might therefore increase when you enable PAS. PAS utilizes cluster placement at the generic level to make logic structuring physical-aware. It targets logic structures with large multiplexers and large homogeneous trees.

Related Information

Affects this command: [synthesize -generic](#)

Related attribute: (subdesign) [physical_aware_structuring](#) on page 408

regions

`regions string`

Read-only design attribute. Returns information for all regions defined for the design in the DEF file.

Example

`rc:/designs/rct> get_attr regions`

Attribute Reference for Encounter RTL Compiler

Physical—Design Attributes

```
{REGION_ONE region {{120.0 120.0} {180.0 180.0} {240.0 240.0} {300.0 300.0}}
{{regPropString {prop string}} {regPropInteger 42} {regPropReal 3.14}}}
{REGION_TWO fence {{30.0 30.0} {50.0 50.0}} {}}
{REGION_THREE guide {{80.0 80.0} {90.0 90.0}} {}}
```

Related Information

Affects this command: [read_def](#)

Related attributes: [Region Attributes](#) on page 472

sdp_files

`sdp_files`

Read-only design attribute. Returns the SDP relative placement file that has been read in.

Related Information

Set by this command: [read_sdp_file](#)

sdp_type

`sdp_type {no_value | datapath | ff_column | both }`

Default: no_value

Read-write design attribute. Specifies the type of the SDP relative placement file read in for the design to create SDP constraints.

Set by this command: [read_sdp_file](#)

utilization

`utilization float`

Read-only design attribute. Returns the design utilization.

utilization_threshold

`utilization_threshold float`

Default: 0.95

Read-write design attribute. Specifies the maximum value that the design can reach for utilization during synthesis.

The specified value must be in the range of [0.0, 1.0] where a value of 1.0 corresponds to 100% utilization for the design.

Related Information

Related commands:

[synthesize -to_placed](#)
[synthesize -spatial](#)

Pin Attributes

Contain information about the specified instance pin. A pin is a single bit connection point on an instance. Pin objects are stored in `pins_in` and `pins_out` directories.

- To set a pin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pin instance/pin]
```

- To get a pin attribute value, type

```
get_attribute attribute_name [find /des*/design -pin instance/pin]
```

Note: You may need to specify more than one instance to serve as the search path that will uniquely identify the relevant pin.

location_x

`location_x float`

Read-only `pin` attribute. Returns the physical X coordinate of the pin in microns. The `location_x` value is derived from the DEF.

Related Information

Related attribute: [location_y on page 405](#)

location_y

`location_y float`

Read-only `pin` attribute. Returns the physical Y coordinate of the pin in microns. The `location_y` value is derived from the DEF.

Related Information

Related attribute: [location_x on page 405](#)

physical

`physical {false | true}`

Read-only pin attribute. Indicates whether this is a pure physical pin or not. Pure physical pins are only defined in the LEF library.

Related Information

Related attribute: [\(pgpin\) physical on page 402](#)

physical_del

`physical_del delay`

Read-only pin attribute. Returns the delay, in picoseconds, computed from the parasitic annotation.

Related Information

Related attribute: [physical_res on page 400](#)

physical_res

`physical_res integer`

Read-only pin attribute. Returns the resistance, in ohms, from the parasitic annotation.

Related Information

Related attributes: [physical_del on page 400](#)

placement_status

`placement_status string`

Read-only pin attribute. Returns the placement status of the pin. Possible values are:

- cover—pin has a location, is part of the cover macro and cannot be moved by automatic tools
- fixed—pin has a location and cannot be moved by automatic tools
- placed—pin has a location and can be moved by automatic tools
- unplaced—pin has no location

Example

```
rc:/designs/rct/> get_att placement_status [find / -pin *foo2]  
placed
```

Related Information

Set by this command: [read_def](#)

Pgpin Attributes

Contain information about power and ground instance pins. Pgpin objects are stored in the pgpins_in and pgpins_out directories.

- To set a pgpin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pgpin instance/pin]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pgpin instance/pin]
```

location_x

location_x float

Read-only pgpin attribute. Returns the physical X coordinate of the pgpin in microns.

Related Information

Related attribute: [location_y on page 402](#)

location_y

location_y float

Read-only pgpin attribute. Returns the physical Y coordinate of the pgpin in microns.

Related Information

Related attribute: [location_x on page 402](#)

physical

physical {false | true}

Read-only pgpin attribute. Indicates whether this pgpin is a pure physical pin or not. Pure physical pins are only defined in the LEF library.

Related Information

Related attribute: [\(pin\) physical on page 400](#)

placement_status

`placement_status string`

Read-only pgpin attribute. Returns the placement status of the pgpin. Possible values are:

- `cover`—pgpin has a location, is part of the cover macro and cannot be moved by automatic tools
- `fixed`—pgpin has a location and cannot be moved by automatic tools
- `placed`—pgpin has a location and can be moved by automatic tools
- `unplaced`—pgpin has no location

Example

```
rc:/designs/rct/> get_att placement_status [find / -pgpin *foo2]  
placed
```

Related Information

Set by this command: [read_def](#)

Net Attributes

Contain information about a net in the specified design. A net can be a top-level net or can belong to a hierarchical instance.

- To set a net attribute, type

```
set_attribute attribute_name attribute_value \  
[find /des*/design -net instance/net]
```

- To get a net attribute value, type

```
get_attribute attribute_name [find /des*/design -net instance/net]
```

Note: You may need to specify several instances to uniquely identify the net.

Note: These attributes are located at /designs/design/nets.

physical_cap

```
physical_cap float
```

Read-write net attribute. Specifies the capacitance, in femtofarads, from the parasitic annotation.

Port Attributes

Contain information about single bit connection points in the specified top-level design. Port objects are stored in `ports_in` and `ports_out` directories.

- To set a port attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -port name]
```

- To get a port attribute value, type

```
get_attribute attribute_name [find /des*/design -port name]
```

location_x

`location_x float`

Read-only `port` attribute. Returns the physical X coordinate of the port in microns. The `location_x` value is derived from the DEF.

Related Information

Related attribute: [location_y on page 405](#)

location_y

`location_y float`

Read-only `port` attribute. Returns the physical Y coordinate of the port in microns. The `location_y` value is derived from the DEF.

Related Information

Related attribute: [location_x on page 405](#)

physical_del

`physical_del delay`

Read-only `port` attribute. Returns the delay, in picoseconds, computed from the parasitic annotation.

Related Information

Related attribute: [physical_res](#) on page 406

physical_res

`physical_res integer`

Read-only [port](#) attribute. Returns the resistance, in ohms, from the parasitic annotation.

Related Information

Related attribute: [physical_del](#) on page 405

placement_status

`placement_status string`

Read-only [port](#) attribute. Returns the placement status of the pgpin. Possible values are:

- `cover`—port has a location, is part of the cover macro and cannot be moved by automatic tools
- `fixed`—port has a location and cannot be moved by automatic tools
- `placed`—port has a location and can be moved by automatic tools
- `unplaced`—port has no location

Example

```
rc:/designs/rct/> get_att placement_status [find / -port *foo2]  
placed
```

Related Information

Set by this command: [read_def](#)

Subdesign Attributes

Contain information about the subdesigns in the specified design. Subdesigns correspond to Verilog modules or VHDL entities instantiated in the top-level Verilog module or top-level VHDL entity.

- To set a subdesign attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subdesign name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name [find /des*/design -subdesign name]
```

fplan_height

fplan_height float

Read-only subdesign attribute. Returns the height, in microns, of the physical block for a subdesign.

Related Information

Related attribute: [fplan_width on page 407](#)

fplan_width

fplan_width float

Read-only subdesign attribute. Returns the width, in microns, of the physical block for a subdesign.

Related Information

Related attribute: [fplan_height on page 407](#)

physical_aware_restructuring

```
physical_aware_restructuring {false | true}
```

Default: false

Read-write subdesign attribute. Enables physical-aware mapping and restructuring on the subdesign after global placement in the RC-Physical flow. Logic trees are re-optimized and balanced with physical context, and concurrently placed to optimize for timing and wire length.

Related Information

Affects this command: [synthesize -to_placed](#)

Related attribute: (design) [physical_aware_restructuring](#) on page 396

physical_aware_structuring

```
physical_aware_structuring {inherited | false | true}
```

Default: inherited

Read-write subdesign attribute. Controls physical-aware structuring (PAS) on the RTL subdesign. PAS optimizes for congestion and is timing-aware. The tool makes a tradeoff between area and congestion. The logic gate area might therefore increase when you enable PAS. PAS utilizes cluster placement at the generic level to make logic structuring physical-aware. It targets logic structures with large multiplexers and large homogeneous trees. If the value of this attribute is `inherited`, the subdesign attribute inherits the value from the `physical_aware_structuring` design attribute.

Related Information

Affects this command: [synthesize -generic](#)

Related attribute: (subdesign) [physical_aware_structuring](#) on page 408

Instance Attributes

Contain information about the specified instance.

- To set a `instance` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -instance name]
```

- To get a `instance` attribute value, type

```
get_attribute attribute_name [find /des*/design -instance name]
```

CX

`cx float`

Read-only `instance` attribute. Returns the X-coordinate of the center of the instance in microns.

Example

Consider the following code to determine the X-coordinate of an instance:

```
set llx [get_attr location_x instance]
set urx [expr {$llx + [get_attr width instance]}]
set cx [expr {$llx + (($urx - $llx) / 2.0)}]
```

These 3 lines can be replaced with:

```
set cx [get_attr cx instance]
```

Related Information

Related attributes: [cy on page 409](#)

CY

`cy float`

Read-only `instance` attribute. Returns the Y-coordinate of the center of the instance in microns.

Example

Consider the following code to determine the Y-coordinate of an instance:

```
set lly [get_attr location_y instance]  
set ury [expr {$lly + [get_attr height instance] }]  
set cy [expr {$lly + ($ury - $lly) / 2.0}]
```

These 3 lines can be replaced with:

```
set cy [get_attr cy instance]
```

Related Information

Related attributes: [cx](#) on page 409

llx

llx float

Read-only [*instance*](#) attribute. Returns the physical X-coordinate of the lower left hand corner of the instance in microns. The *llx* value is derived from the DEF.

Related Information

Related attributes: [lly](#) on page 410
 [urx](#) on page 413
 [ury](#) on page 413

lly

lly float

Read-only [*instance*](#) attribute. Returns the physical Y-coordinate of the lower left hand corner of the instance in microns. The *lly* value is derived from the DEF.

Related Information

Related attributes: [llx](#) on page 410
 [urx](#) on page 413
 [ury](#) on page 413

height

`height float`

Read-only instance attribute. Returns the height, in microns, of the object based on the information from the physical library.

Related Information

Related attributes:

(libcell) height on page 197
(instance) width on page 413
(libcell) width on page 211

location_x

`location_x float`

Read-only instance attribute. Returns the physical X-coordinate of the lower left hand corner of the instance in microns. The `location_x` value is derived from the DEF.

Related Information

Related attribute: [location_y](#) on page 411

location_y

`location_y float`

Read-only instance attribute. Returns the physical Y-coordinate of the lower left hand corner of the instance in microns. The `location_x` value is derived from the DEF.

Related Information

Related attribute: [location_x](#) on page 411

placement_status

`placement_status string`

Read-only instance attribute. Returns the placement status of the instance. Possible values are:

- `cover`—instance has a location, is part of the cover macro and cannot be moved by automatic tools
- `fixed`—instance has a location and cannot be moved by automatic tools
- `placed`—instance has a location and can be moved by automatic tools
- `unplaced`—instance has no location

Example

```
rc:/designs/rct/> get_att placement_status [find / -instance *foo2]  
placed
```

Related Information

Set by this command: [read_def](#)

skip_in_write_def

`skip_in_write_def {false | true}`

Default: `false`

Read-write instance attribute. Controls whether to skip the content of this instance when writing out the DEF file.

Note: Applies only to hierarchical instances.

You can use this attribute in a hierarchical DEF flow to create a top-level DEF file that does not contain the content of the hierarchical instances. Content of these hierarchical instances can be added later.

Related Information

Affects this command: [write_def](#)

Attribute Reference for Encounter RTL Compiler

Physical—Instance Attributes

urx

urx float

Read-only instance attribute. Returns the physical X-coordinate of the upper right hand corner of the instance in microns. The *urx* value is derived from the DEF.

Related Information

Related attributes: [llx](#) on page 410
 [lly](#) on page 410
 [ury](#) on page 413

ury

ury float

Read-only instance attribute. Returns the physical Y-coordinate of the upper right hand corner of the instance in microns. The *ury* value is derived from the DEF.

Related Information

Related attributes: [llx](#) on page 410
 [lly](#) on page 410
 [urx](#) on page 413

width

width float

Read-only instance attribute. Returns the width, in microns, of the object based on the information from the physical library.

Related Information

Related attributes: [height](#) on page 411
 (libcell) [height](#) on page 197
 (libcell) [width](#) on page 211

Blockage Attributes

Contain information about the specified blockage. These attributes are read-only attributes, so you cannot set their values.

- To get a blockage attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -blockage blockage]
```

Note: These attributes are located at

```
/designs/design/physical/blockages/blockage
```

boxes

```
boxes {{layer {llx lly urx ury} ...}}
```

Read-only blockage attribute. Returns a Tcl list with the lower left and upper right coordinates of each rectangular-shaped box that is part of this blockage.

Example

```
rc:/designs/DTMF_CHIP> get_att boxes [find / -blockage 4_placement]
{893.64 482.16 931.26 1081.92} {759.0 451.92 1106.82 502.32}
```

Related Information

Set by this command: [read_def](#)

Related attribute: (region) [boxes](#) on page 472

component

```
component string
```

Read-only blockage attribute. Returns a component name if the blockage is associated with a component.

Related Information

Set by this command: [read_def](#)

Related attribute: (hdl_inst) [component](#) on page 334

density

density float

Default: 0.00

Read-only blockage attribute. Returns the percentage of the blockage area that can be used for standard cells during initial placement if the blockage was marked partial.

Example

In the following example 50% of the 3_placement blockage can be used for placement of standard cells.

```
rc:/designs/DTMF_CHIP> get_att density [find / -blockage 3_placement]  
0.50
```

Related Information

Set by this command: [read_def](#)

Related attribute: [is_partial](#) on page 416

has_fills

has_fills {false | true}

Default: false

Read-only blockage attribute. Indicates whether the blockage has metal fills.

Related Information

Set by this command: [read_def](#)

has_slots

`has_slots {false |true}`

Default: false

Read-only blockage attribute. Indicates whether the blockage has slots.

Related Information

Set by this command: [read_def](#)

is_exceptpgnet

`is_exceptpgnet {false |true}`

Default: false

Read-only blockage attribute. Indicates whether the blockage only blocks signal net routing, and does not block power or ground net routing.

Related Information

Set by this command: [read_def](#)

is_partial

`is_partial {false |true}`

Default: false

Read-only blockage attribute. Indicates whether the blockage can be partially used for initial placement.

Related Information

Set by this command: [read_def](#)

Related attribute: [density on page 415](#)

is_pushdown

`is_pushdown {false |true}`

Default: false

Read-only blockage attribute. Indicates whether the blockage was pushed down into the block from the top level of the design.

Related Information

Set by this command: [read_def](#)

is_soft

`is_soft {false |true}`

Default: false

Read-only blockage attribute. Indicates whether the blockage area can be used in phases of the design after initial placement.

Related Information

Set by this command: [read_def](#)

layer

`layer string`

Read-only blockage attribute. Returns the name of the layer for which the blockage is defined.

Note: This attribute applies only to routing blockages.

Example

```
rc:/designs/DTMF_CHIP> get_att layer [find / -blockage 9_routing_Metal3]
Metal3
```

Related Information

Set by this command: [read_def](#)

location_x

`location_x float`

Read-only blockage attribute. Returns the physical X-coordinate of the lower left hand corner of the blockage in microns. The `location_x` value is derived from the DEF.

Related Information

Related attribute: [location_y on page 418](#)

location_y

`location_y float`

Read-only blockage attribute. Returns the physical Y-coordinate of the lower left hand corner of the blockage in microns. The `location_x` value is derived from the DEF.

Related Information

Related attribute: [location_x on page 418](#)

mask

`mask integer`

Read-only blockage attribute. Returns the mask number used for double- or triple-patterning lithography.

Related Information

Set by this command: [read_def](#)

max_layer

`max_layer string`

Read-only blockage attribute. Defines the top layer of a range of metal layers to which this blockage applies.

Note: This attribute applies only to routing halo blockages.

Related Information

Set by this command: [read_def](#)

min_layer

`min_layer string`

Read-only blockage attribute. Defines the bottom layer of a range of metal layers to which this blockage applies.

Note: This attribute applies only to routing halo blockages.

Related Information

Set by this command: [read_def](#)

polygons

`polygons string`

Read-only blockage attribute. Returns a Tcl list of coordinates of at least three points if the blockage has a polygon geometry.

Example

```
rc:/designs/DTMF_CHIP> get_att polygons [find / -blockage 5_routing_Metal6]
{{555.945 290.93} {555.945 323.32} {616.1 323.32}} {{616.1 323.32} {616.1 290.93}
{555.945 290.93}}
```

Related Information

Set by this command: [read_def](#)

spacing

`spacing float`

Default: 0.000

Read-only blockage attribute. Returns the minimum spacing allowed between the blockage and any other routing shape. For a routing blockage either the spacing or the width is specified.

Related Information

Set by this command: [read_def](#)

Related attribute: [\(blockage\) width](#) on page 421

type

`type string`

Read-only blockage attribute. Returns the blockage type. This attribute can have any of the following values: placement_soft, placement_halo, placement, placement_halo_soft, placement_partial, routing, routing_halo.

Example

```
rc:/designs/DTMF_CHIP> get_att type [find / -blockage 9_routing_Metal3]  
routing
```

Related Information

Set by this command: [read_def](#)

user_created

`user_created {false | true}`

Read-only blockage attribute. Indicates whether the blockage was created by the user in RTL Compiler. The attribute returns false if the blockage was defined in the DEF file.

Related Information

Set by these commands: [create_placement_blockage](#)

Attribute Reference for Encounter RTL Compiler

Physical—Blockage Attributes

[create_placement_halo_blockage](#)
[create_routing_blockage](#)
[create_routing_halo_blockage](#)
[read_def](#)

visible

`visible {false | true}`

Default: false

Read-write [blockage](#) attribute. Indicates whether this blockage is visible in the GUI.

width

`width float`

Default: 0.000

Read-only [blockage](#) attribute. Returns the effective width that must be used for the purposes of spacing calculations. For a routing blockage either the spacing or the width is specified.

Related Information

Affected by this command: [read_def](#)

Related attribute: [spacing](#) on page 420

Bump Attributes

Contain information about the specified bump. Bumps represent the solder bumps on a chip. Bumps are instantiated in the DEF COMPONENTS section but not instantiated in the netlist. Bump cells are usually placed with + COVER placement status. In addition, they have a libcell of type cover_bump. These attributes are read-only attributes, so you cannot set their values.

- To get a bump attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -bump bump]
```

Note: These attributes are located at

```
/designs/design/physical/bumps/bump
```

def_name

```
def_name string
```

Read-only bump attribute. Returns the DEF name of the bump.

Related Information

Set by this command: [read_def](#)

height

```
height float
```

Default: no_value

Read-only bump attribute. Returns the height of the bump.

Example

```
rc:/designs/rct> get_att height [find / -bump *bump2]  
4000.000
```

Related Information

Set by this command: [read_def](#)

Related attribute: [width](#) on page 427

Attribute Reference for Encounter RTL Compiler

Physical—Bump Attributes

location_x

`location_x float`

Default: no_value

Read-only bump attribute. Returns the x-coordinate of the lower left corner of the bump.

Example

```
rc:/designs/rct> get_att location_x [find / -bump *bump2]
17740.000
```

Related Information

Set by this command: [read_def](#)

Related attribute [location_y](#) on page 423

location_y

`location_y float`

Default: 0.000

Read-only bump attribute. Returns the y-coordinate of the lower left corner of the bump.

Example

```
rc:/designs/rct/physical> get_att location_y [find / -bump *bump2]
24000.000
```

Related Information

Set by this command: [read_def](#)

Related attribute [location_x](#) on page 423

model

`model string`

Read-only bump attribute. Returns the model of which this bump is an instance. The model refers to a MACRO defined in the LEF library.

Attribute Reference for Encounter RTL Compiler

Physical—Bump Attributes

Example

```
rc:/designs/rct/physical> get_att model [find / -bump *bump2]  
bumpcell
```

Related Information

Set by this command: [read_def](#)

orientation

orientation string

Read-only [bump](#) attribute. Returns the orientation of the bump. Following are the possible orientations: N, S, E, W, FN, FS, FE, or FW.

Example

```
rc:/designs/rct> get_att orientation [find / -bump *bump2]  
W
```

Related Information

Set by this command: [read_def](#)

Related attribute: (row) [orientation](#) on page 477

placement_status

placement_status string

Read-only [bump](#) attribute. Returns the placement status of the bump. Possible values are:

- cover—bump has a location, is part of the cover macro and cannot be moved by automatic tools
- fixed—bump has a location and cannot be moved by automatic tools
- placed—bump has a location and can be moved by automatic tools
- unplaced—bump has no location

Note: Bump cells are usually placed with + COVER placement status.

Attribute Reference for Encounter RTL Compiler

Physical—Bump Attributes

Example

```
rc:/designs/rct/> get_att placement_status [find / -bump *bump2]  
cover
```

Related Information

Set by this command: [read_def](#)

properties

properties string

Read-only [bump](#) attribute. Returns the properties associated with the bump.

Related Information

Set by this command: [read_def](#)

urx

urx float

Default: 0.000

Read-only [bump](#) attribute. Returns the x-coordinate of the upper right corner of the bump.

Example

```
rc:/designs/rct/physical> get_att urx [find / -bump *bump2]  
25740.000
```

Related Information

Set by this command: [read_def](#)

ury

ury float

Default: 0.000

Example

```
rc:/designs/rct/physical> get_att ury [find / -bump *bump2]  
28000.000
```

Read-only bump attribute. Returns the y-coordinate of the upper right corner of the bump.

Related Information

Set by this command: [read_def](#)

weight

weight integer

Read-only bump attribute. Returns the weight assigned to the bump, which determines whether or not automatic placement attempts to keep the bump near the location specified in the DEF file. The weight is only meaningful when the bump is placed.

Related Information

Set by this command: [read_def](#)

Attribute Reference for Encounter RTL Compiler

Physical—Bump Attributes

width

`width float`

Default: 0.000

Read-only bump attribute. Returns the width of the bump.

Example

```
rc:/designs/rct/physical> get_att width [find / -bump *bump2]  
8000.000
```

Related Information

Set by this command: [read_def](#)

Related attribute: [height](#) on page 422

DEF_Pin Attributes

Contain information about the external pins present in the design. These attributes are created when the DEF file is read in. The information is based on the PINS statement in the DEF file.

These attributes are read-only attributes, so you cannot set their values.

- To get a `def_pin` attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -def_pin pin_name]
```

Note: These attributes are located at

```
/designs/design/physical/def_pins/pin_name
```

direction

`direction string`

Read-only `def_pin` attribute. Returns the direction of the physical pin. Following are the possible values:

- INPUT—Pin that accepts signals coming into the cell.
- OUTPUT—Pin that drives signals out of the cell.
- INOUT—Pin that can accept signals going either in or out of the cell.
- FEEDTHRU—Pin that goes completely across the cell.

Related Information

Set by this command: [read_def](#)

layers

```
layers {layer minspacing designrulewidth {llx lly urx ury}} ...
```

Read-only `def_pin` attribute. Returns one or more lists with the layer information of the physical pin if the pin was placed. Each list contains the layer name, the minimum spacing allowed between this pin and any other routing shape, the effective layer width, and the lower left and upper right coordinates for each pin location.

Attribute Reference for Encounter RTL Compiler

Physical—DEF_Pin Attributes

Note: In the DEF file, you can only specify the minimum spacing or the effective width. The attribute, however, lists two numbers after the layer name, one of which will be 0. The non-zero value corresponds to the information that was specified in the DEF file.

Note: This attribute will be empty if the pin has no PORT statements **and** no LAYER statements.

Related Information

Set by this command: [read_def](#)

location_x

`location_x float`

Read-only [def_pin](#) attribute. Returns the x-coordinate of the physical pin if the pin was placed.

Related Information

Set by this command: [read_def](#)

location_y

`location_y float`

Read-only [def_pin](#) attribute. Returns the y-coordinate of the physical pin if the pin was placed.

Related Information

Set by this command: [read_def](#)

net_expr

`net_expr "netExprPropName defaultNetName"`

Read-only [def_pin](#) attribute. Returns the net expression associated with the physical pin.

Related Information

Set by this command: [read_def](#)

net_name

`net_name string`

Read-only def_pin attribute. Returns the internal net name associated with the physical pin. This net name is defined in the NETS or SPECIALNETS statement.

Related Information

Set by this command: [read_def](#)

orientation

`orientation string`

Read-only def_pin attribute. Returns the orientation of the physical pin if the pin was placed.

Related Information

Set by this command: [read_def](#)

placement_status

`placement_status string`

Read-only def_pin attribute. Returns the placement status of physical pin. Possible values are:

- COVER—Pin is part of the cover macro and cannot be moved
- FIXED—Pin cannot be moved by automatic tools, but can be moved by interactive commands.
- PLACED—Pin can be moved by automatic tools.
- UNPLACED—Pin has not been placed.

Related Information

Set by this command: [read_def](#)

polygons

```
polygons {layer minspacing designrulewidth {pt pt pt [pt]} } ...
```

Read-only def_pin attribute. Returns one or more lists with polygon information of the physical pin if the pin was placed. Each list contains the layer name, the minimum spacing allowed between this pin and any other routing shape, the effective layer width, and a list of coordinates of at least 3 points. A polygon is generated by connecting each successive point, and then the first and last points.

Note: In the DEF file, you can only specify the minimum spacing or the effective width. The attribute, however, lists two numbers after the layer name, one of which will be 0. The non-zero value corresponds to the information that was specified in the DEF file.

Note: This attribute will be empty if the pin has no PORT statements **and** no POLYGON statements.

Related Information

Set by this command: [read_def](#)

ports

```
ports { placementStatus location_x location_y orientation
    {{layer minspacing designrulewidth {llx lly urx ury}} ...}
    {{layer minspacing designrulewidth {pt pt pt [pt]}} ...}
    {{vianame location_x location_y}...} }...
```

Read-only def_pin attribute. Returns one or more lists if the DEF file contains PORT statement(s) for the physical pin . Each list contains

- placement status
- location of the physical pin
- orientation
- a list with layer information (corresponding to the LAYER statements for the PORT)
- a list with polygon information (corresponding to the POLYGON statements for the PORT)
- a list with via information (corresponding to the VIA statements for the PORT)

If the DEF did not contain a LAYER, POLYGON or VIA statement for the PORT, the corresponding list will be empty.

Related Information

Set by this command: [read_def](#)

special

`special {false | true}`

Default: false

Read-only [def_pin](#) attribute. Indicates whether the pin is a special pin. In the place and route tool, a special router routes special wiring to special pins.

Related Information

Set by this command: [read_def](#)

use

`use string`

Read-only [def_pin](#) attribute. Returns the use of the pin. Following are the possible values:

- ANALOG—Used for analog connectivity
- CLOCK—Used for clock net connectivity
- GROUND—Used for connectivity to the chip-level ground distribution network
- POWER—Used for connectivity to the chip-level power distribution network
- RESET—Used as a reset pin
- SCAN—Used as a scan pin
- SIGNAL—Used for regular net connectivity
- TIEOFF—Used as a tie-high or tie-low pin

Related Information

Set by this command: [read_def](#)

vias

```
vias {vianame location_x location_y}...
```

Read-only def_pin attribute. Returns one or more lists with via information of the physical pin if the pin was placed. Each list contains the name of a previously defined via (in the DEF or LEF) and the location of the via.

Note: This attribute will be empty is the pin has no PORT statements **and** no VIA statements.

Related Information

Set by this command: [read_def](#)

visible

```
visible {true | false}
```

Default: true

Read-write def_pin attribute. Indicates whether this physical pin is visible in the GUI.

Fill Attributes

Contain information about the metal fills present in the design. These attributes are created when the DEF file is read in. The information is based on the **FILLS** statement in the DEF file.

These attributes are read-only attributes, so you cannot set their values.

- To get a `fill` attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -fill fill]
```

Note: These attributes are located at

```
/designs/design/physical/fills/fill
```

boxes

```
boxes {{llx lly urx ury} ...}
```

Read-only `fill` attribute. Returns one or more lists. Each list contains the lower left and upper right coordinates of a rectangular-shaped box that is part of this fill.

Related Information

Set by this command: [read_def](#)

Related attributes: [layer](#) on page 434

[polygons](#) on page 435

layer

```
layer string
```

Read-only `fill` attribute. Returns the layer associated with this fill.

Related Information

Set by this command: [read_def](#)

Related attributes: [boxes](#) on page 434

[polygons](#) on page 435

mask

`mask integer`

Read-only fill attribute. Returns the mask number used for the layer of this fill in case double- or triple-patterning lithography is used.

Related Information

Set by this command: [read_def](#)

opc

`opc {false | true}`

Read-only fill attribute. Indicates whether the fill shape requires OPC correction during mask generation.

Related Information

Set by this command: [read_def](#)

polygons

`polygons {{pt pt pt [pt]} ...}`

Read-only fill attribute. Returns one or more lists. Each list contains the coordinates of at least 3 points of a polygon that is part of this fill. The polygon is generated by connecting each successive point, and then the first and last points.

Related Information

Set by this command: [read_def](#)

Related attributes: [boxes](#) on page 434

[layer](#) on page 434

via

`via string`

Read-only fill attribute. Returns the name of the via associated with this fill.

Related Information

Set by this command: [read_def](#)

via_mask

`via_mask integer`

Read-only fill attribute. Returns the mask number associated with the fill via in case double or triple patterning lithography is used.

Related Information

Set by this command: [read_def](#)

via_opc

`via_opc {false | true}`

Read-only fill attribute. Indicates whether the fill via requires OPC correction during mask generation.

Related Information

Set by this command: [read_def](#)

via_points

`via_points {pt pt pt [pt]}`

Read-only fill attribute. Returns the polygon geometry for the fill via.

Related Information

Set by this command: [read_def](#)

Gcell Attributes

Contain information about the specified gcell (global routing cell). Gcells are derived from the GCELLGRID statements in the DEF file. These attributes are read-only attributes, so you cannot set their values.

- To get a gcell attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -gcell gcell]
```

Note: These attributes are located at

```
/designs/design/physical/gcells/gcell
```

box

```
box {llx lly urx ury}
```

Read-only gcell attribute. Returns the lower left and upper right coordinates of the gcell. The coordinates can be floating numbers.

Example

```
rc:/designs/fifo> get_att box [find / -gcell 1:2,8:81]  
9.430 33.005 14.030 37.105
```

Related Information

Set by this command: [read_def](#)

horizontal_remaining

```
horizontal_remaining integer
```

Default: no_value

Read-only gcell attribute. Returns the number of horizontal tracks that are not required for horizontal routing.

Related Information

Set by this command: [read_def](#)

instance_count

instance_count integer

Default: 0

Read-only gcell attribute. Lists the number of instances that the gcell contains.

Example

```
rc:/designs/fifo> get_att instance_count [find / -gcell 1:2,8:81]
3
```

Related Information

Set by this command: [read_def](#)

instances

instances string

Read-only gcell attribute. Lists the names of the instances that the gcell contains.

Example

```
rc:/designs/fifo> get_att instances [find / -gcell 1:2,8:81]
{mem_reg[3][1]} {mem_reg[7][2]} {mem_reg[5][1]}
```

Related Information

Set by this command: [read_def](#)

pin_count

`pin_count integer`

Default: 0

Read-only gcell attribute. Returns the number of instance pins that can be accessed within the gcell.

Example

```
rc:/designs/fifo> get_att pin_count [find / -gcell 1:2,8:81]  
1
```

Related Information

Set by this command: [read_def](#)

pin_density

`pin_density float`

Default: 0.00

Read-only gcell attribute. Returns the number of pins in the gcell divided by the gcell area.

Related Information

Set by this command: [read_def](#)

pins

`pins pin_list`

Read-only gcell attribute. Lists the names of the instance pins that can be accessed within the gcell.

Example

```
rc:/designs/fifo> get_att pins [find / -gcell 1:2,8:81]  
{mem_reg[7][2]/CK}
```

Related Information

Set by this command: [read_def](#)

utilization

`utilization float`

Read-only gcell attribute. Returns the actual utilization of the gcell.

Related Information

Set by this command: [read_def](#)

vertical_remaining

`vertical_remaining integer`

Default: no_value

Read-only gcell attribute. Returns the number of vertical tracks that are not required for vertical routing.

Related Information

Set by this command: [read_def](#)

Group Attributes

Contain information about the specified DEF group. These attributes are read-only attributes, so you cannot set their values.

- To get a gcell attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -group group]
```

Note: These attributes are located at

/designs/design/physical/groups/group

def_name

def_name string

Read-only group attribute. Returns either the original DEF group name if the group was defined in the DEF file, or the user-defined name in case the group was created in RTL Compiler.

Related Information

Set by these commands: [create_group](#)

[read_def](#)

members

members string

Read-only group attribute. Lists the members associated with the group.

Example

```
rc:/designs/rct> get_att members [find / -group GROUP_THREE]
add_23_3*
```

Related Information

Set by these commands: [create_group](#)

[read_def](#)

Attribute Reference for Encounter RTL Compiler

Physical—Group Attributes

properties

properties string

Read-only group attribute. Lists the properties (names and values) that are associated with the group.

Example

```
rc:/designs/rct> get_att properties [find / -group GROUP_THREE]  
{grpPropInteger 42}
```

Related Information

Affected by this command: [read_def](#)

region

region string

Read-only group attribute. Lists the name of the region that is associated with the group.

Note: All instances assigned to this group must be placed within the specified region but other instances not belonging to the group can also be placed within the region.

Example

```
rc:/designs/rct> get_att region [find / -group GROUP_THREE]  
REGION_TWO
```

Related Information

Set by these commands: [create_group](#)

[read_def](#)

Related attribute: [groups](#) on page 392

Attribute Reference for Encounter RTL Compiler

Physical—Group Attributes

user_created

```
user_created {false | true}
```

Read-only group attribute. Indicates whether the group was created by the user in RTL Compiler. The attribute returns `false` if the group was defined in the DEF file.

Related Information

Set by these commands:

[create_group](#)

[read_def](#)

Layer Attributes

Contain information specified for a layer in the LEF LAYER section.

- To set a layer attribute value, type

```
set_attribute attribute_name attribute_value \
[find /des*/*/phys* -layer layer]
```

- To get a layer attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -layer layer]
```

Note: These attributes are located at

```
/designs/design/physical/layers/layer
```

cap_multiplier

```
cap_multiplier float
```

Read-only layer attribute. Returns the multiplier for the interconnect capacitance that accounts for increases in capacitance caused by nearby wires.

Example

```
rc:/designs/rct> get_attr cap_multiplier [find / -layer METAL2]
1.0
```

Related Information

Set by this attribute: [lef library](#) on page 272

capacitance

```
capacitance float
```

Read-only layer attribute. Returns the capacitance for each square unit, in picofarads per square micron.

Example

```
rc:/designs/rct> get_attr capacitance [find / -layer METAL2]
3.625e-05
```

Attribute Reference for Encounter RTL Compiler

Physical—Layer Attributes

Related Information

Set by this attribute: [lef_library](#) on page 272

Related attribute: (libpin) [capacitance](#) on page 213

cap_table_name

`cap_table_name string`

Read-only [layer](#) attribute. Returns the corresponding layer name in the capacitance table file.

Example

```
rc:/designs/DTMF_CHIP> get_attr cap_table_name [find / -layer Metal2]  
M2
```

Related Information

Set by this attribute: [cap_table_file](#) on page 364

color

`color string`

Read-write [layer](#) attribute. Specifies the color of the layer in the GUI.

direction

`direction {horizontal | vertical}`

Read-only [layer](#) attribute. Returns the preferred routing direction for this layer.

Note: The tool ignores the diag45 and diag135 values

Example

```
rc:/designs/rct> get_attr direction [find / -layer METAL2]  
vertical
```

Related Information

Set by this attribute: [lef_library](#) on page 272

layer_index

`layer_index integer`

Read-only layer attribute. Returns the index for this layer. The first routing layer in the process has index 0.

Note: The layers are defined in the LEF file in process order from bottom to top.

Example

```
rc:/designs/DTMF_CHIP> get_attr layer_index [find / -layer Metal3]  
2
```

Related Information

Set by this attribute: [cap_table_file](#) on page 364

min_spacing

`min_spacing float`

Read-only layer attribute. Returns the minimum spacing between tracks on the layer.

offset

`offset float`

Read-only layer attribute. Returns the offset from the design origin for the preferred direction routing tracks of the layer.

offset_x

`offset_x float`

Read-only layer attribute. Returns the x offset from the design origin for the vertical routing tracks of the layer.

offset_y

`offset_y float`

Read-only layer attribute. Returns the y offset from the design origin for the horizontal routing tracks of the layer.

pitch

pitch float

Read-only layer attribute. Returns the required routing pitch (in microns) for the layer. The pitch for a given routing layer specifies the distance between routing tracks in the preferred direction for that layer.

Example

```
rc:/designs/rct> get_attr pitch [find / -layer METAL2]  
0.8
```

Related Information

Set by this attribute: [lef_library on page 272](#)

pitch_x

pitch_x float

Read-only layer attribute. Returns the required x routing pitch (in microns) for the layer. The x pitch specifies the distance between the vertical routing tracks for that layer.

Related Information

Set by this attribute: [lef_library on page 272](#)

pitch_y

pitch_y float

Read-only layer attribute. Returns the required y routing pitch (in microns) for the layer. The y pitch specifies the distance between the horizontal tracks for that layer.

Related Information

Set by this attribute: [lef_library on page 272](#)

Attribute Reference for Encounter RTL Compiler

Physical—Layer Attributes

resistance

`resistance float`

Read-only layer attribute. Returns the resistance for a square of wire, in ohms per square.

Example

```
rc:/designs/rct> get_attr resistance [find / -layer METAL2]  
0.076
```

Related Information

Set by this attribute: [lef library](#) on page 272

type

`type {cut | implant | masterslice | overlap | routing}`

Read-only layer attribute. Returns the type of the layer.

Example

```
rc:/designs/rct> get_attr type [find / -layer METAL2]  
routing
```

Related Information

Set by this attribute: [lef library](#) on page 272

utilization

`utilization float`

Read-write layer attribute. Specifies the layer utilization.

Each metal layer has a fixed number of tracks that can be used for routing. This attribute lets you define the percentage of tracks that the router can use for this layer.

Example

The following command sets the utilization to 50%.

```
rc:/designs/rct> set_attr utilization 0.5 [find / -layer METAL2]
```

Related Information

Set by this attribute: [lef_library](#) on page 272

visible

`visible {false | true}`

Default: false

Read-write [layer](#) attribute. Indicates whether this layer is visible in the GUI.

width

`width float`

Read-only [layer](#) attribute. Returns the default routing width (in microns) to use for all regular wiring on the layer.

Example

```
rc:/designs/rct> get_attr width [find / -layer METAL2]  
0.4
```

Related Information

Set by this attribute: [lef_library](#) on page 272

Nondefaultrule Attributes

Contain information about any nondefault rules used in this design. These attributes are created when the LEF and DEF files are read in. The information is based on the NONDEFAULTRULES statement in the LEF or DEF file.

Note: If the same rule exists in both LEF and DEF, the name of the second one read in will be appended with _copy.

These attributes are read-only attributes, so you cannot set their values.

- To get a nondefaultrule attribute value, type

```
get_attribute attribute_name [find /des*//*/phys* -nondefaultrule rule]
```

Note: These attributes are located at

```
/designs/design/physical/nondefaultrules/rule
```

from_lef

```
from_lef {false | true}
```

Read-only nondefaultrule attribute. Indicates whether the nondefault rule was specified in the LEF file. If false, the nondefaultrule was specified in the DEF file.

Related Information

Set by this command: [read_def](#)

Related attributes: [hardspacing](#) on page 451

[layers](#) on page 451

[mincuts](#) on page 452

[properties](#) on page 452

[viarules](#) on page 453

[vias](#) on page 453

hardspacing

hardspacing {false | true}

Read-only nondefaultrule attribute. Specifies whether any spacing values that exceed the LEF LAYER ROUTING spacing requirements are *hard* rules instead of *soft* rules. By default, routers treat extra spacing requirements as soft rules.

Related Information

Set by this command: [read_def](#)

Related attributes: [from_lef](#) on page 450

[layers](#) on page 451

[mincuts](#) on page 452

[properties](#) on page 452

[viarules](#) on page 453

[vias](#) on page 453

layers

layers {layer layername width minwidth diagwidth diagWidth spacing minspacing wireext wireext} ...

Read-only nondefaultrule attribute. Returns one or more lists. Each list contains the name of a routing layer and various width and spacing values to be used for this nondefault rule.

Related Information

Set by this command: [read_def](#)

Related attributes: [from_lef](#) on page 450

[hardspacing](#) on page 451

[mincuts](#) on page 452

[properties](#) on page 452

[viarules](#) on page 453

[vias](#) on page 453

mincuts

```
mincuts {layer cutLayerName numCuts}...
```

Read-only nondefaultrule attribute. Returns one or more lists. Each list contains the layer name, the cutlayer name and the minimum number of cuts required for this non-default rule.

Related Information

Set by this command: [read_def](#)

Related attributes: [from_lef](#) on page 450

[hardspacing](#) on page 451

[layers](#) on page 451

[properties](#) on page 452

[viarules](#) on page 453

[vias](#) on page 453

properties

```
properties {propertyName propertyValue}...
```

Read-only nondefaultrule attribute. Returns one or more lists. Each list contains a property defined for this non-default rule, that is, a property name followed by its value.

Related Information

Set by this command: [read_def](#)

Related attributes: [from_lef](#) on page 450

[hardspacing](#) on page 451

[layers](#) on page 451

[mincuts](#) on page 452

[viarules](#) on page 453

[vias](#) on page 453

viarules

`viarules viarulename ...`

Read-only nondefaultrule attribute. Returns the viarule(s) to be used with this nondefault rule previously defined in a LEF VIARULE GENERATE statement.

Related Information

Set by this command: [read_def](#)

Related attributes: [from_lef](#) on page 450

[hardspacing](#) on page 451

[layers](#) on page 451

[mincuts](#) on page 452

[properties](#) on page 452

[vias](#) on page 453

vias

`vias vianame ...`

Read-only nondefaultrule attribute. Returns previously defined LEF or DEF vias to be used with this nondefault rule.

Related Information

Set by this command: [read_def](#)

Related attributes: [from_lef](#) on page 450

[hardspacing](#) on page 451

[layers](#) on page 451

[mincuts](#) on page 452

[properties](#) on page 452

[viarules](#) on page 453

Pcell Attributes

Contain information about the specified pcell. Pcells are instantiated in the DEF COMPONENTS section but not instantiated in the netlist. In addition, the SOURCE type specified in the COMPONENTS section is either DIST or USER. These attributes are read-only attributes, so you cannot set their values.

- To get a pcell attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -pcell pcell]
```

Note: These attributes are located at

```
/designs/design/physical/pcells/pcell
```

def_name

```
def_name string
```

Read-only pcell attribute. Returns the DEF name of the pcell.

Example

```
rc:/designs/rct> get_attr def_name [find /des*/*/phys* -pcell pcell_1]  
U1/foo3
```

Related Information

Set by this command: [read_def](#)

height

```
height float
```

Default: no_value

Read-only pcell attribute. Returns the height of the pcell.

Example

```
rc:/designs/rct> get_att height [find / -pcell *foo2]  
4000.000
```

Related Information

Set by this command: [read_def](#)

Related attribute: [width on page 459](#)

location_x

location_x float

Default: no_value

Read-only pcell attribute. Returns the x-coordinate of the lower left corner of the pcell.

Example

```
rc:/designs/rct> get_att location_x [find / -pcell *foo2]  
17740.000
```

Related Information

Set by this command: [read_def](#)

Related attribute [location_y on page 455](#)

location_y

location_y float

Default: 0.000

Read-only pcell attribute. Returns the y-coordinate of the lower left corner of the pcell.

Example

```
rc:/designs/rct/physical> get_att location_y [find / -pcell *foo2]  
24000.000
```

Related Information

Set by this command: [read_def](#)

Related attribute [location_x on page 455](#)

model

model string

Read-only pcell attribute. Returns the model of which this pcell is an instance. The model refers to a MACRO defined in the LEF library.

Example

```
rc:/designs/rct/physical> get_att model [find / -pcell *foo2]
OAI21XL
```

Related Information

Set by this command: [read_def](#)

orientation

orientation string

Read-only pcell attribute. Returns the orientation of the pcell. Following are the possible orientations: N, S, E, W, FN, FS, FE, or FW.

Example

```
rc:/designs/rct> get_att orientation [find / -pcell *foo2]
W
```

Related Information

Set by this command: [read_def](#)

Related attribute: (row) [orientation](#) on page 477

placement_status

placement_status string

Read-only pcell attribute. Returns the placement status of the pcell. Possible values are:

- cover—pcell has a location, is part of the cover macro and cannot be moved by automatic tools
- fixed—pcell has a location and cannot be moved by automatic tools
- placed—pcell has a location and can be moved by automatic tools

Attribute Reference for Encounter RTL Compiler

Physical—Pcell Attributes

- unplaced—pcell has no location

Example

```
rc:/designs/rct/> get_att placement_status [find / -pcell *foo2]  
placed
```

Related Information

Set by this command: [read_def](#)

properties

properties string

Read-only pcell attribute. Returns the properties associated with the pcell.

Related Information

Set by this command: [read_def](#)

urx

urx float

Default: 0.000

Read-only pcell attribute. Returns the x-coordinate of the upper right corner of the pcell.

Example

```
rc:/designs/rct/physical> get_att urx [find / -pcell *foo2]  
25740.000
```

Related Information

Set by this command: [read_def](#)

ury

ury float

Default: 0.000

Example

```
rc:/designs/rct/physical> get_att ury [find / -pcell *foo2]  
28000.000
```

Read-only pcell attribute. Returns the y-coordinate of the upper right corner of the pcell.

Related Information

Set by this command: [read_def](#)

weight

weight integer

Read-only pcell attribute. Returns the weight assigned to the pcell, which determines whether or not automatic placement attempts to keep the pcell near the location specified in the DEF file. The weight is only meaningful when the pcell is placed.

Related Information

Set by this command: [read_def](#)

width

`width float`

Default: 0.000

Read-only pcell attribute. Returns the width of the pcell.

Example

```
rc:/designs/rct/physical> get_att width [find / -pcell *foo2]  
8000.000
```

Related Information

Set by this command: [read_def](#)

Related attribute: [height](#) on page 454

Pdomain Attributes

Contain physical information about the power domains specified in the DEF file. These attributes are read-only attributes, so you cannot set their values.

- To get a pdomain attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -pdomain power_domain]
```

Note: These attributes are located at

```
/designs/design/physical/pdomains/power_domain
```

boundary

boundary string

Read-only pdomain attribute. Returns the coordinates of the physical boundary of the power domain.

Example

```
rc:/designs/dtmf_recv_core> get_attr boundary [find . -pdomain PLL]
30.94 251.86 171.92 329.98
```

Related Information

Set by this command: [read_def](#)

boxes

boxes string

Read-only pdomain attribute. Returns a Tcl list of power domain region boxes after cutout processing.

Example

```
rc:/designs/dtmf_recv_core> get_attr boxes [find . -pdomain PLL]
{30.94 251.86 124.88 329.98} {124.88 289.66 171.92 329.98}
```

Related Information

Set by this command: [read_def](#)

cutouts

cutouts string

Read-only pdomain attribute. Returns the list of power domain region cutouts. For example, if a square region boundary is defined, you can remove a corner by specifying a cutout.

Related Information

Set by this command: [read_def](#)

mingap

mingap string

Read-only pdomain attribute. Returns the top, bottom, left, and right distance, in microns, that must be reserved from the power domain boundary edges for power routing.

Example

```
rc:/> get_attr mingap [find / -pdomain pd_08v]  
10.1 20.2 30.5 40
```

Related Information

Set by this command: [read_def](#)

net

net string

Read-only pdomain attribute. Returns the list of power and ground nets, and global nets that apply to the power domain.

Example

```
rc:/designs/dtmf_recv_core> get_attr net [find . -pdomain PLL]  
VDD Avdd Avss
```

Related Information

Set by this command: [read_def](#)

rsext

`rsext string`

Read-only pdomain attribute. Returns the top, bottom, left, and right boundary for legal targets to be used by the power planning and routing commands, in conjunction with the power domain boundary.

Example

```
rc:/> get_attr rsext [find / -pdomain pd_08v]  
1.1 2.3 3.5 4.7
```

Related Information

Set by this command: [read_def](#)

Pnet Attributes

Contain information about netlist connectivity for nets. These attributes are created when the DEF file is read in. The information is based on the NETS section in the DEF file.

- To set a pnet attribute value, type

```
set_attribute attribute_name attribute_value [find /des*/*/phys* -pnet net]
```

- To get a pnet attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -pnet net]
```

Note: These attributes are located at

```
/designs/design/physical/pnets/net
```

capacitance

capacitance float

Default: no_value

Read-only pnet attribute. Returns the estimated wire capacitance. This is the value specified for the ESTCAP keyword in the DEF file.

Related Information

Set by this command: [read_def](#)

components

```
components {componentName pinName {0|1} }...
```

Read-only pnet attribute. Returns one or more lists. Each list contains the regular component pin on a net or a subnet, its corresponding component, and an indication whether the pin is part of a synthesized scan chain.

Related Information

Set by this command: [read_def](#)

def_name

`def_name string`

Read-only `pnet` attribute. Returns the name of the DEF net.

Related Information

Set by this command: [read_def](#)

fixedbump

`fixedbump {false | true}`

Default: false

Read-only `pnet` attribute. Indicates whether the bump net in the net can be reassigned to a different pin.

Related Information

Set by this command: [read_def](#)

frequency

`frequency float`

Default: no_value

Read-only `pnet` attribute. Returns the frequency of the net in Hertz. The frequency value is used by the router to choose the correct number of via cuts required for a given net.

Related Information

Set by this command: [read_def](#)

name

`name string`

Read-only `pnet` attribute. Returns the same value as basename for this `pnet` object.

nondefaultrule

`nondefaultrule string`

Read-only pnet attribute. Returns the LEF-defined nondefaultrule name that is used when creating the net and wiring.

Related Information

Set by this command: [read_def](#)

original_name

`original_name string`

Read-only pnet attribute. Returns the name of the original net that was partitioned and that includes this net. This name was specified with the ORIGINAL keyword in the DEF file.

Related Information

Set by this command: [read_def](#)

path_count

`path_count integer`

Default: 0

Read-only pnet attribute. Returns the number of paths for this net.

Related Information

Set by this command: [read_def](#)

path_index

`path_index integer`

Default: 0

Read-write pnet attribute. Specifies the index of the path of the net for which you want to get more information through the path_value attribute.

Related Information

Sets this attribute: [path_value](#) on page 466

path_value

path_value string

Read-only pnet attribute. Returns the information for the path identified through the `path_index` attribute.

Related Information

Set by this attribute: [path_index](#) on page 465

pattern

pattern string

Read-only pnet attribute. Returns the routing pattern used for this net. The routing pattern can be one of the following:

- BALANCED—Used to minimize skews in timing delays for clock nets.
- STEINER—Used to minimize net length.
- TRUNK—Used to minimize delay for global nets.
- WIREDLOGIC—Used in ECL designs to connect output and mustjoin pins before routing to the remaining pins.

Related Information

Set by this command: [read_def](#)

properties

properties {propertyName propertyName}...

Read-only pnet attribute. Returns one or more lists. Each list contains a property defined for this net, that is, a property name followed by its value.

Attribute Reference for Encounter RTL Compiler

Physical—Pnet Attributes

Related Information

Set by this command: [read_def](#)

rc_name

`rc_name string`

Read-only pnet attribute. Returns the name that RTL Compiler gave to this net.

shieldnet

`shieldnet string`

Read-only pnet attribute. Returns the name of a special net that shields this net.

Related Information

Set by this command: [read_def](#)

source

`source string`

Read-only pnet attribute. Returns how the net was created.

- DIST—Net is the result of adding physical components (that is, components that only connect to power or ground nets), such as filler cells, well-taps, tie-high and tie-low cells, and decoupling caps.
- NETLIST—Net is defined in the original netlist. This is the default value, and is not normally written out in the DEF file.
- TEST—Net is part of a scan chain.
- TIMING—Net represents a logical rather than physical change to netlist, and is used typically as a buffer for a clock-tree, or to improve timing on long nets.
- USER—Net is user defined.

Related Information

Set by this command: [read_def](#)

use

`use string`

Read-only pnet attribute. Returns the use of the net.

- ANALOG—Used as an analog signal net.
- CLOCK—Used as a clock net.
- GROUND—Used as a ground net.
- POWER—Used as a power net.
- RESET—Used as a reset net.
- SCAN—Used as a scan net.
- SIGNAL—Used as a digital signal net.
- TIEOFF—Used as a tie-high or tie-low net.

Related Information

Set by this command: [read_def](#)

visible

`visible {false | true}`

Default: false

Read-only pnet attribute. Indicates whether this net is visible in the GUI.

weight

`weight integer`

Default: no_value

Read-only pnet attribute. Returns the weight assigned to the net. Automatic layout tools attempt to shorten the lengths of nets with high weights. A value of 0 indicates that the net length for that net can be ignored. The value of 1 specifies that the net should be treated normally. A larger weight specifies that the tool should try harder to minimize the net length of that net.

Attribute Reference for Encounter RTL Compiler

Physical—Pnet Attributes

Related Information

Set by this command: [read_def](#)

xtalk

`xtalk integer`

Default: no_value

Read-only pnet attribute. Returns the crosstalk class number for the net. The value ranges between 0 and 200.

Related Information

Set by this command: [read_def](#)

Power Domain Attributes

Contain physical information about the power domains specified in the DEF file. These attributes are read-only attributes, so you cannot set their values.

- To get a `power_domain` attribute value, type

```
get_attribute attribute_name [find /des*/*/power -power_domain power_domain]
```

Note: These attributes are located at

```
/designs/design/power/power_domains/power_domain
```

box_list

`box_list string`

Read-only `power_domain` attribute. Returns a Tcl list of boxes that define the power domain boundary.

Related Information

Set by this command: [modify_power_domain_attr](#)

disjoint_hinst_box_list

`disjoint_hinst_box_list string`

Read-only `power_domain` attribute. Returns a list of hierachcial instances and disjoint boxes that define the power domain boundary.

Related Information

Set by this command: [modify_power_domain_attr](#)

min_gaps

`min_gaps string`

Read-only `power_domain` attribute. Returns the top, bottom, left, and right distance, in microns, that must be reserved from the power domain boundary edges for power routing.

Attribute Reference for Encounter RTL Compiler

Physical—Power Domain Attributes

Example

```
rc:/> get_attr min_gaps [find / -power_domain pd_08v]  
10.1 20.2 30.5 40
```

Related Information

Set by this command: [modify_power_domain_attr](#)

rs_ext

rs_ext string

Read-only [power_domain](#) attribute. Returns the top, bottom, left, and right boundary for legal targets to be used by the power planning and routing commands, in conjunction with the power domain boundary.

Example

```
rc:/> get_attr rs_ext [find / -power_domain pd_08v]  
1.1 2.3 3.5 4.7
```

Related Information

Set by this command: [modify_power_domain_attr](#)

Region Attributes

Contain information about the specified DEF region. These attributes are read-only attributes, so you cannot set their values.

- To get a region attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -region region]
```

Note: These attributes are located at

/designs/design/physical/regions/region

boxes

```
boxes {llx lly urx ury}
```

Read-only region attribute. Returns one or more lists. Each list contains the lower left and upper right coordinates of a rectangular area in the region. The coordinates can be floating numbers.

Example

```
rc:/designs/DTMF_CHIP> get_att boxes [find / -region REGION_ONE]
{120.0 120.0 180.0 180.0} {240.0 240.0 300.0 300.0}
```

Related Information

Set by these commands: [create_region](#)
[read_def](#)

def_name

```
def_name string
```

Read-only region attribute. Returns either the original DEF region name if the region was defined in the DEF file, or the user-defined name in case the region was created in RTL Compiler.

Related Information

Set by these commands: [create_region](#)
[read_def](#)

derived_from_power_domain

`derived_from_power_domain {false | true}`

Read-only region attribute. Indicates whether this region was created when the power domain physical boundary was created.

Related Information

Set by this command: [modify_power_domain_attr](#)

group

`group string`

Read-only region attribute. Returns the group associated with this region.

location_x

`location_x float`

Read-only region attribute. Returns the physical X-coordinate of the lower left hand corner of the region in microns. The `location_x` value is derived from the DEF.

Related Information

Related attribute: [location_y on page 473](#)

location_y

`location_y float`

Read-only region attribute. Returns the physical Y-coordinate of the lower left hand corner of the region in microns. The `location_x` value is derived from the DEF.

Related Information

Related attribute: [location_x on page 418](#)

Attribute Reference for Encounter RTL Compiler

Physical—Region Attributes

properties

properties string

Read-only region attribute. Lists the properties that are associated with the region.

Related Information

Set by this command: [read_def](#)

type

type string

Read-write region attribute. Specifies the type of the region.

`fence` indicates that all instances assigned to this region must be exclusively placed inside the region boundaries. No other instances are allowed inside this region.

`guide` indicates that all instances assigned to this region should be placed inside this region; however, it is a preference, not a hard constraint.

Example

```
rc:/designs/DTMF_CHIP> get_att type [find / -region DTMF*]  
guide
```

Related Information

Set by these commands: [create_region](#)
 [read_def](#)

user_created

user_created {false | true}

Read-only region attribute. Indicates whether the region was created by the user in RTL Compiler. The attribute returns `false` if the region was defined in the DEF file.

Related Information

Set by these commands: [create_region](#)
 [read_def](#)

Row Attributes

Contain information about the specified DEF row. These attributes are read-only attributes, so you cannot set their values.

- To get a `row` attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -row row]
```

Note: These attributes are located at

`/designs/design/physical/rows/row`

height

`height float`

Default: no_value

Read-only `row` attribute. Returns the height of the row.

Example

```
rc:/designs/fifo> get_att height [find / -row CORE_ROW_7]
3.690
```

Related Information

Set by these commands:

[create_row](#)

[read_def](#)

Related attribute:

[width](#) on page 478

is_horizontal

`is_horizontal {true | false }`

Default: true

Read-only `row` attribute. Indicates whether the row is horizontal or vertical.

Example

```
rc:/designs/fifo> get_att is_horizontal [find / -row CORE_ROW_7]
true
```

Attribute Reference for Encounter RTL Compiler

Physical—Row Attributes

Related Information

Set by this command: [read_def](#)

location_x

`location_x float`

Default: no_value

Read-only [row](#) attribute. Returns the x-coordinate of the lower left corner of the row.

Example

```
rc:/designs/fifo/> get_att location_x [find / -row CORE_ROW_7]  
0.000
```

Related Information

Set by these commands: [create_row](#)
[read_def](#)

Related attribute: [location_y](#) on page 476

location_y

`location_y float`

Default: no_value

Read-only [row](#) attribute. Returns the y-coordinate of the lower left corner of the row.

Example

```
rc:/designs/fifo/> get_att location_y [find / -row CORE_ROW_7]  
25.830
```

Related Information

Set by these commands: [create_row](#)
[read_def](#)

Related attribute: [location_x](#) on page 476

Attribute Reference for Encounter RTL Compiler

Physical—Row Attributes

macro

`macro string`

Read-only row attribute. Retruns the name of the LEF site used for the row.

Example

```
rc:/designs/fifo> get_att macro [find / -row CORE_ROW_7]
tsml2site
```

Related Information

Set by these commands: [create_row](#)
 [read_def](#)

orientation

`orientation string`

Read-only row attribute. Returns the orientation of all sites in the row. Following are the possible orientations: N, S, E, W, FN, FS, FE, or FW.

Example

```
rc:/designs/fifo> get_att orientation [find / -row CORE_ROW_7]
N
```

Related Information

Set by this command: [read_def](#)

user_created

`user_created {false | true}`

Read-only row attribute. Indicates whether the row was created by the user in RTL Compiler. The attribute returns `false` if the row was defined in the DEF file.

Related Information

Set by these commands: [create_row](#)
 [read_def](#)

visible

`visible {true | false}`

Default: true

Read-write [row](#) attribute. Indicates whether the row is displayed in the GUI.

Related Information

Set by these commands:

[create_row](#)

[read_def](#)

width

`width float`

Default: no_value

Read-only [row](#) attribute. Returns the physical width of the row.

Example

```
rc:/designs/fifo> get_att width [find / -row CORE_ROW_7]  
93.380
```

Related Information

Set by these commands:

[create_row](#)

[read_def](#)

Related attribute:

[height](#) on page 475

Site Attributes

Contain information about the placement sites in the design. These attributes are created when the LEF library is read in. The information is based on the SITE statement in the LEF file.

These attributes are read-only attributes, so you cannot set their values.

- To get a site attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -site site]
```

Note: These attributes are located at

```
/designs/design/physical/sites/site
```

class

```
class string
```

Read-only site attribute. Returns the site class name.

Related Information

Set by this attribute: [lef_library on page 272](#)

height

```
height float
```

Default: 0.000

Read-only site attribute. Returns the site height in microns.

Related Information

Set by this attribute: [lef_library on page 272](#)

symmetry

```
symmetry {x | y | R90}
```

Read-only site attribute. Returns the site symmetry.

Attribute Reference for Encounter RTL Compiler

Physical—Site Attributes

Related Information

Set by this attribute: [lef_library](#) on page 272

width

`width float`

Default: 0.000

Read-only [site](#) attribute. Returns the site width in microns.

Related Information

Set by this attribute: [lef_library](#) on page 272

Slot Attributes

Contain information about the slotting of the wires in the design. These attributes are created when the DEF file is read in. The information is based on the SLOTS statement in the DEF file.

These attributes are read-only attributes, so you cannot set their values.

- To get a slot attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -slot slot]
```

Note: These attributes are located at

```
/designs/design/physical/slots/slot
```

boxes

```
boxes {{layer {llx lly urx ury} ...}}
```

Read-only slot attribute. Returns one or more lists. Each list contains the lower left and upper right coordinates of a rectangular-shaped slot.

Related Information

Set by this command: [read_def](#)

Related attributes: [layer](#) on page 481

[polygons](#) on page 482

layer

```
layer string
```

Read-only slot attribute. Returns the layer associated with this slot.

Related Information

Set by this command: [read_def](#)

Related attributes: [boxes](#) on page 481

[polygons](#) on page 482

polygons

`polygons {{pt pt pt [pt]} ...}`

Read-only slot attribute. Returns one or more lists. Each list contains a list of coordinates of at least three points of a polygon that defines a slot. The polygon is generated by connecting each successive point, and then the first and last points.

Related Information

Set by this command: [read_def](#)

Related attributes: [boxes](#) on page 481

[layer](#) on page 481

Special Net Attributes

Contain information about netlist connectivity for nets containing special pins. These attributes are created when the DEF file is read in. The information is based on the SPECIALNETS statement in the DEF file.

These attributes are read-only attributes, so you cannot set their values.

- To get a specialnet attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -specialnet net]
```

Note: These attributes are located at

```
/designs/design/physical/specialnet/net
```

components

```
components {componentName pinName {0|1} }...
```

Read-only specialnet attribute. Returns one or more lists. Each list contains the special pin on the net, its corresponding component, and an indication whether the pin is part of a synthesized scan chain.

Related Information

Set by this command: [read_def](#)

def_name

```
def_name string
```

Read-only specialnet attribute. Returns the DEF name of the special net.

Related Information

Set by this command: [read_def](#)

fixedbump

```
fixedbump {false | true}
```

Read-only specialnet attribute. Indicates whether the bump net in the special net can be reassigned.

Attribute Reference for Encounter RTL Compiler

Physical—Special Net Attributes

Related Information

Set by this command: [read_def](#)

name

`name string`

Read-only specialnet attribute. Returns the name of the special net.

Related Information

Set by this command: [read_def](#)

original_name

`original_name string`

Read-only specialnet attribute. Returns the name of the original net that was partitioned and that includes this special net.

Related Information

Set by this command: [read_def](#)

path_count

`path_count integer`

Read-only specialnet attribute. Returns the number of paths for this special net. Paths contain the routing point data in the special wiring statement of the special net that are not part of the POLYGON or RECT statements.

Related Information

Set by this command: [read_def](#)

path_index

`path_index integer`

Default: 0

Read-write specialnet attribute. Specifies the index of the path of the special net for which you want to get more information through the path_value attribute.

Related Information

Sets this attribute: [path_value on page 485](#)

path_value

`path_value string`

Read-only specialnet attribute. Returns the information for the path identified though the path_index attribute.

Related Information

Set by this attribute: [path_index on page 485](#)

pattern

`pattern string`

Read-only specialnet attribute. Returns the routing pattern used for this special net. The routing pattern can be one of the following:

- BALANCED—Used to minimize skews in timing delays for clock nets.
- STEINER—Used to minimize net length.
- TRUNK—Used to minimize delay for global nets.
- WIREDLOGIC—Used in ECL designs to connect output and mustjoin pins before routing to the remaining pins.

Related Information

Set by this command: [read_def](#)

polygons

`polygons {{layer {pt pt pt [pt]} ...}}`

Read-only specialnet attribute. Returns one or more lists. Each list defines a polygon on the specified layer that is part of the routing of this special net.

Related Information

Set by this command: [read_def](#)

properties

`properties {propertyName propertyName}...`

Read-only specialnet attribute. Returns one or more lists. Each list contains a property defined for this specialnet, that is, a property name followed by its value.

Related Information

Set by this command: [read_def](#)

rc_name

`rc_name string`

Read-only specialnet attribute. Returns the name that RTL Compiler gave to this specialnet.

rectangles

`rectangles {layer l1x l1y urx ury} ...`

Read-only specialnet attribute. Returns one or more lists. Each list defines a rectangle on the specified layer that is part of the routing of this special net.

Related Information

Set by this command: [read_def](#)

source

`source string`

Read-only specialnet attribute. Returns how the net was created. The source can be one of the following:

- DIST—Net is the result of adding physical components (that is, components that only connect to power or ground nets), such as filler cells, well-taps, tie-high and tie-low cells, and decoupling caps.
- NETLIST—Net is defined in the original netlist. This is the default value, and is not normally written out in the DEF file.
- TEST—Net is part of a scan chain.
- TIMING—Net represents a logical rather than physical change to netlist, and is used typically as a buffer for a clock-tree, or to improve timing on long nets.
- USER—Net is user defined.

Related Information

Set by this command: [read_def](#)

style

`style {no_value | integer }`

Read-only specialnet attribute. Returns the index of the style that defines the outer boundary for this special net wire. The no_value value indicates that the special net does not use a style.

Related Information

Set by this command: [read_def](#)

type

type string

Read-only specialnet attribute. Returns the routing wiring type (cover, fixed, or routed) of the special net.

Related Information

Set by this command: [read_def](#)

use

use string

Read-only specialnet attribute. Returns the use of the specialnet. Following are the possible values:

- ANALOG—Used as an analog signal net.
- CLOCK—Used as a clock net.
- GROUND—Used as a ground net.
- POWER—Used as a power net.
- RESET—Used as a reset net.
- SCAN—Used as a scan net.
- SIGNAL—Used as a digital signal net.
- TIEOFF—Used as a tie-high or tie-low net.

Related Information

Set by this command: [read_def](#)

voltage

voltage *float*

Read-only specialnet attribute. Returns the voltage of the specialnet.

Related Information

Set by this command: [read_def](#)

weight

weight *integer*

Read-only specialnet attribute. Returns the weight assigned to the special net.

Related Information

Set by this command: [read_def](#)

Style Attributes

Contain information about styles. A style polygon defines a wire's outer boundary. These attributes are created when the DEF file is read in. The information is based on the STYLES statement in the DEF file.

These attributes are read-only attributes, so you cannot set their values.

- To get a `style` attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -style name]
```

Note: These attributes are located at

`/designs/design/physical/styles/name`

index

`index integer`

Read-only `style` attribute. Returns the style index value (number following a STYLE statement).

Related Information

Set by this command: [read_def](#)

polygon

`polygon {x y} {x y}...`

Read-only `style` attribute. Returns a list of coordinates from which the style polygon can be generated.

Related Information

Set by this command: [read_def](#)

Track Attributes

Contain track (or routing grid) information for each layer. These attributes are created when the DEF file is read in. The information is based on the TRACKS statements in the DEF file. The names for the objects in the tracks directory are generated based on the layer name and the routing direction. Y indicates that the track information is for the horizontal direction. X indicates that the track information is for the vertical direction.

These attributes are read-only attributes, so you cannot set their values.

- To get a track attribute value, type

```
get_attribute attribute_name [find /des*/*/phys* -track track]
```

Note: These attributes are located at

```
/designs/design/physical/tracks/track
```

count

```
count integer
```

Read-only track attribute. Returns the number of tracks for the layer identified by the layer attribute in the routing direction identified by the is_horizontal attribute. This number corresponds to the value specified for the DO keyword in the TRACKS statement in the DEF file.

Example

The following command shows the number of tracks in the horizontal direction for layer Metal2:

```
rc:/designs/DTMF_CHIP> get_attr count Metal2_Y_5  
2410
```

Related Information

Set by this command: [read_def](#)

is_horizontal

```
is_horizontal {false | true}
```

Read-only track attribute. Indicates for which direction the track information applies. The direction is determined by the X or Y specification following the TRACKS statement.

Attribute Reference for Encounter RTL Compiler

Physical—Track Attributes

Example

The following command shows the direction for which the track information applies:

```
rc:/designs/DTMF_CHIP> get_attr is_horizontal Metal2_Y_5  
true
```

Related Information

Set by this command: [read_def](#)

is_used

```
is_used {false | true}
```

Read-only [track](#) attribute. Indicates whether the track is used during the physical layout estimation.

layer

```
layer string
```

Read-only [track](#) attribute. Returns the layer to which the track information applies.

Example

The following command shows the layer for which the track information applies:

```
rc:/designs/DTMF_CHIP> get_attr layer Metal2_Y_5  
Metal2
```

Related Information

Set by this command: [read_def](#)

macro

```
macro string
```

Read-only [track](#) attribute. Returns the macro associated with these tracks.

Attribute Reference for Encounter RTL Compiler

Physical—Track Attributes

Example

The following command shows the macro for which the track information applies:

```
rc:/designs/DTMF_CHIP> get_attr macro Metal2_Y_5  
Y
```

Related Information

Set by this command: [read_def](#)

mask

```
mask integer
```

Read-only [track](#) attribute. Returns the mask number used for the first routing track.in case double- or triple-patterning lithography is used.

Related Information

Set by this command: [read_def](#)

same_mask

```
same_mask {false | true}
```

Read-only [track](#) attribute. Indicates whether all routing tracks use the same mask as the first track in case double- or triple-patterning lithography is used.

start

```
start float
```

Read-only [track](#) attribute. Returns the X or Y coordinate of the first line. The value will be an X (Y) coordinate if the `is_horizontal` attribute is set to `false` (`true`).

Example

The following command shows the coordinate for which the track information applies:

```
rc:/designs/DTMF_CHIP> get_attr start Metal2_Y_5  
0.280
```

Attribute Reference for Encounter RTL Compiler

Physical—Track Attributes

Related Information

Set by this command: [read_def](#)

step

step float

Read-only [track](#) attribute. Returns the spacing between the tracks.

Example

The following command shows the layer for which the track information applies:

```
rc:/designs/DTMF_CHIP> get_attr step Metal2_Y_5  
0.560
```

Related Information

Set by this command: [read_def](#)

Via Attributes

Contain information about fixed vias and generated vias. These attributes are created when the DEF file is read in. The information is based on the VIAS statement in the DEF file. The via names correspond to the via names specified in the VIAS statement. All vias consist of shapes on three layers: a cut layer and two routing layers that connect through the cut layer.

A fixed via is defined using rectangles or polygons, and does not use a VIARULE. A generated via is defined using VIARULE parameters that are derived from a VIARULE GENERATE statement in the LEF file.

These attributes are read-only attributes, so you cannot set their values.

- To get a via attribute value, type

```
get_attribute attribute_name [find /des*//*/phys* -via via]
```

Note: These attributes are located at

```
/designs/design/physical/vias/via
```

bottom_layer

```
bottom_layer string
```

Read-only via attribute. Returns the name of the bottom routing layer associated with the via.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

boxes

```
boxes {{layer mask {llx lly urx ury} ...}}
```

Read-only via attribute. Returns one or more lists. Each list defines the via geometry for the specified layer: it contains the layer name, which mask for double- or triple-patterning lithography is to be applied to the defined shape, and the lower left and upper right coordinates of the via shape. The coordinates are specified in microns and can be floating numbers.

Related Information

Set by this command: [read_def](#)

cut_cols

`cut_cols integer`

Read-only via attribute. Returns the number of cut columns that make up the via array.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

cut_layer

`cut_layer string`

Read-only via attribute. Returns the name of the cut layer associated with the via.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

cut_pattern

`cut_pattern string`

Read-only via attribute. Returns an ASCII string that represents the cut pattern associated with the via. A cut pattern is used when some of the cuts are missing from the array of cuts. When no cut pattern is available, all cuts are assumed to be present.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

cut_rows

`cut_rows integer`

Read-only via attribute. Returns the number of cut rows that make up the via array.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

polygons

`polygons {{layer mask {pt pt pt [pt]} ...}}`

Read-only via attribute. Returns one or more lists. Each list contains a layer name, which mask for double- or triple-patterning lithography is to be applied to the defined shape, and a list of coordinates of at least 3 points. A polygon is generated by connecting each successive point, and then the first and last points.

Related Information

Set by this command: [read_def](#)

top_layer

`top_layer string`

Read-only via attribute. Returns the name of the top routing layer associated with the via.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

viarule_name

`viarule_name string`

Read-only via attribute. Returns the name of the LEF VIARULE that produced this via.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

xbottom_enclosure

`xbottom_enclosure float`

Read-only via attribute. Returns the required x enclosure (in micron) for the bottom layer. This enclosure measures the distance in the horizontal direction from the edge of the cut array to the edge of the bottom metal layer that encloses the cut array.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

xbottom_offset

`xbottom_offset float`

Read-only via attribute. Returns the xoffset of the bottom layer (in micron).

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

xcut_size

`xcut_size float`

Read-only via attribute. Returns the required width of the cut layer rectangle (in micron).

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

xcut_spacing

xcut_spacing float

Read-only via attribute. Returns the required spacing between cuts in the horizontal direction. The spacing is measured from one cut edge to the next cut edge and is specified in microns.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

xorigin_offset

xorigin_offset float

Read-only via attribute. Returns the x offset of the origin of the via shapes (in micron). By default, the 0,0 origin of the via is the center of the cut array and the enclosing metal rectangles.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

xtop_enclosure

xtop_enclosure float

Read-only via attribute. Returns the required x enclosure for the top layer (in micron). This enclosure measures the distance in the horizontal direction from the edge of the cut array to the edge of the top metal layer that encloses the cut array.

Note: This attribute has no value for fixed vias.

Attribute Reference for Encounter RTL Compiler

Physical—Via Attributes

Related Information

Set by this command: [read_def](#)

xtop_offset

`xtop_offset float`

Read-only via attribute. Returns the x offset of the top layer (in micron).

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

ybottom_enclosure

`ybottom_enclosure float`

Read-only via attribute. Returns the required y enclosure for the bottom layer (in micron). This enclosure measures the distance in the vertical direction from the edge of the cut array to the edge of the bottom metal layer that encloses the cut array.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

ybottom_offset

`ybottom_offset float`

Read-only via attribute. Returns the y offset of the bottom layer (in micron).

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

ycut_size

`ycut_size float`

Read-only via attribute. Returns the required height of the cut layer rectangle (in micron).

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

ycut_spacing

`ycut_spacing float`

Read-only via attribute. Returns the required spacing between cuts in the vertical direction. The spacing is measured from one cut edge to the next cut edge and is specified in microns.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

yorigin_offset

`yorigin_offset float`

Read-only via attribute. Returns they offset of the origin of the via shapes (in micron). By default, the 0,0 origin of the via is the center of the cut array and the enclosing metal rectangles.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

ytop_enclosure

`ytop_enclosure float`

Read-only via attribute. Returns the required y enclosure for the top layer (in micron). This enclosure measures the distance in the vertical direction from the edge of the cut array to the edge of the top metal layer that encloses the cut array.

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

ytop_offset

`ytop_offset float`

Read-only via attribute. Returns the y offset of the top layer (in micron).

Note: This attribute has no value for fixed vias.

Related Information

Set by this command: [read_def](#)

SDP Column Attributes

Contains information about a column in the SDP relative placement file.

- To set an `sdp_column` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design/sdp_groups -sdp_column sdp_columns/column]
```

- To get an `sdp_column` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/sdp_groups -sdp_column sdp_columns/column]
```

These attributes are located in an `sdp_columns` directory below the following directory:

`/designs/design/sdp_groups/sdp_group/`

As the SDP file can be nested, the `sdp_columns` directory can be below an `sdp_group`, or below an `sdp_row`.

flip

`flip {x | y | xy}`

Read-only `sdp_column` attribute. Indicates whether the column is flipped. If no flip value was specified for the column in the SDP relative placement file, the attribute has no value.

Related Information

Set by this command: [read_sdp_file](#)

index

`index integer`

Default: 0

Read-only `sdp_column` attribute. Returns the index or position of this column in the row it belongs to.

Related Information

Set by this command: [read_sdp_file](#)

justify_by

`justify_by {SW | NW | SE | NE | W | E | N | S | MID}`

Default: SW

Read-only `sdp_column` attribute. Returns the justifyBY constraint of the column. If no constraint value was specified for the column in the SDP relative placement file, the value defaults to SW.

Related Information

Set by this command: [read_sdp_file](#)

orient

`orient {R0|R90|R180|R270|MX|MY|MY90|MX90}`

Default: R0

Read-only `sdp_column` attribute. Returns the orientation of the SDP column. If no orientation value was specified for the column in the SDP relative placement file, the value defaults to R0.

Related Information

Set by this command: [read_sdp_file](#)

size_same

`size_same {false | true}`

Default: false

Read-write `sdp_column` attribute. Specifies whether all instances in the column have the same width as the widest instance in the column.

Related Information

Set by this command: [read_sdp_file](#)

skip_value

`skip_value integer`

Default: 0

Read-only `sdp_column` attribute. Specifies the number of columns to skip. This attribute can only have a non-zero value for a column called `skip_column_x`.

Example

The following example lists the attributes for all columns in row `rb`. Row `rb` has two columns specified in the SDP relative placement file: `cb` and `ca`. RTL Compiler creates a column instance with name `skip_column_x` for each column in the column. Following column `cb` is column `skip_column_0` (both columns have index 0). Following column `ca` is `skip_column_1` (both columns have index 1). The `skip_value` for `column_skip_0` is 4, which indicates that there are four empty columns between columns `cb` and `ca`.

```
rc:/designs/test/sdp_groups/ga/sdp_rows/rb/sdp_columns> ls -al
Total: 5 items
./
ca/                      (sdp_column)
    All attributes:
        flip =
        index = 1
        justify_by = N
        orient =
        size_same = false
        skip_value = 0
cb/                      (sdp_column)
    All attributes:
        flip = x
        index = 0
        justify_by = SW
        orient =
        size_same = false
        skip_value = 0
skip_column_0/          (sdp_column)
    All attributes:
        flip =
        index = 0
        justify_by = SW
        orient =
        size_same = false
        skip_value = 4
skip_column_1/          (sdp_column)
    All attributes:
        flip =
        index = 1
        justify_by = SW
        orient =
        size_same = false
        skip_value = 0
```

Attribute Reference for Encounter RTL Compiler

Physical—SDP Column Attributes

Related Information

Set by this command: [read_sdp_file](#)

SDP Group Attributes

Contains information about a top SDP group (or datapath structure) in the SDP relative placement file.

These attributes are read-only attributes, so you cannot set their values.

- To get an `sdp_group` attribute value, type

```
get_attribute attribute_name \
/des*/design/sdp_groups/sdp_group
```

Note: These attributes are located at

`/designs/design/sdp_groups/sdp_group`

hier_path

`hier_path string`

Read-only `sdp_group` attribute. Returns the hierarchical path name of the SDP group (or datapath structure).

Related Information

Set by this command: [read_sdp_file](#)

orient

`orient {R0|R90|R180|R270|MX|MY|MY90|MX90}`

Default: RO

Read-only `sdp_group` attribute. Returns the orientation of the SDP group (or datapath structure). If no orientation value was specified for the SDP group in the SDP relative placement file, the value defaults to R0.

Related Information

Set by this command: [read_sdp_file](#)

origin

`origin x y`

Read-only sdp_group attribute. Returns the coordinates of the origin of the SDP group (or datapath structure).

Related Information

Set by this command: [read_sdp_file](#)

SDP Instance Attributes

Contain information about an instance in the SDP file.

- To set an `sdp_instance` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design/sdp_groups -sdp_instance sdp_instances/instance]
```

- To get an `sdp_instance` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/sdp_groups -sdp_instance sdp_instances/instance]
```

These attributes are located in the `sdp_instances` directory below the following directory:

`/designs/design/sdp_groups/sdp_group/`

As SDP instances can part of a column or row, the `instances` directory can be below an `sdp_column`, or below an `sdp_row`.

flip

`flip {x | y | xy}`

Read-only `sdp_instance` attribute. Indicates whether the instance is flipped. If no flip value was specified for the instance in the SDP file, the attribute has no value.

Related Information

Set by this command: [read_sdp_file](#)

index

`index integer`

Default: 0

Read-only `sdp_instance` attribute. Returns the index or position of the instance in the row or column to which the instance belongs.

Related Information

Set by this command: [read_sdp_file](#)

instance

instance string

Read-only sdp_instance attribute. Returns the full path name of the instance.

Related Information

Set by this command: [read_sdp_file](#)

justify_by

`justify_by {SW | NW | SE | NE | W | E | N | S | MID}`

Default: SW

Read-only sdp_instance attribute. Returns the justifyBY constraint of the instance. If no constraint value was specified for the instance in the SDP file, the value defaults to SW.

Related Information

Set by this command: [read_sdp_file](#)

orient

`orient {R0|R90|R180|R270|MX|MY|MY90|MX90}`

Default: R0

Read-only sdp_column attribute. Returns the orientation of the SDP instance. If no orientation value was specified for the instance in the SDP file, the value defaults to R0.

Related Information

Set by this command: [read_sdp_file](#)

size_fixed

```
size_fixed {false | true}
```

Default: false

Read-write sdp_instance attribute. Specifies whether the size of the instance can be modified during incremental optimization. By default, the size can be modified.

Related Information

Set by this command: [read_sdp_file](#)

skip_value

```
skip_value integer
```

Default: 0

Read-only sdp_instance attribute. Specifies the number of rows or columns to skip between two instances. This attribute can only have a non-zero value for an instance called `skip_instance_x`.

Example

The following example lists the attributes for all instances in row `ra`. Row `ra` has two instances specified in the SDP file: `st_box1_g1` and `st_box1_g2`. RTL Compiler creates an instance with name `skip_instance_x` for each instance in the row. Following instance `st_box1_g1` is instance `skip_instance_0` (both instances have index 0). Following instance `st_box1_g2` is `skip_instance_0` (both instances have index 1). The `skip_value` for `skip_instance_0` is 3, which indicates that there are three empty places between instances `st_box1_g1` and `st_box1_g2`.

```
rc:/designs/test/sdp_groups/ga/sdp_columns/cb/sdp_rows/ra/sdp_instances> ls -al
Total: 5 items
./
skip_instance_0          (sdp_instance)
  All attributes:
    index = 0
    instance =
    justify_by = SW
    size_fixed = false
    skip_value = 3

skip_instance_1          (sdp_instance)
  All attributes:
    index = 1
    instance =
    justify_by = SW
```

Attribute Reference for Encounter RTL Compiler

Physical—SDP Instance Attributes

```
    size_fixed = false
    skip_value = 0
st_box1_g1          (sdp_instance)
  All attributes:
    index = 0
    instance = /designs/test/instances_hier/st/instances_hier/box1
instances_comb/g1
  justify_by = SW
  size_fixed = false
  skip_value = 0
st_box2_g1          (sdp_instance)
  All attributes:
    index = 1
    instance = /designs/test/instances_hier/st/instances_hier/box2/
instances_comb/g1
  justify_by = SW
  size_fixed = false
  skip_value = 0
```

Related Information

Set by this command: [read_sdp_file](#)

SDP Row Attributes

Contains information about a row in the SDP file.

- To set an `sdp_row` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design/sdp_groups -sdp_row sdp_rows/row]
```

- To get an `sdp_row` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/sdp_groups -sdp_row sdp_rows/row]
```

These attributes are located in an `sdp_rows` directory below the following directory:

`/designs/design/sdp_groups/sdp_group/`

As the SDP file can be nested, the `sdp_rows` directory can be below an `sdp_group`, or below an `sdp_column`.

flip

`flip {x | y | xy}`

Default:

Read-only `sdp_row` attribute. Indicates whether the SDP row is flipped. If no flip value was specified for the row in the SDP file, the attribute has no value.

Related Information

Set by this command: [read_sdp_file](#)

index

`index integer`

Default: 0

Read-only `sdp_row` attribute. Returns the index or position of this row in the column it belongs to.

Related Information

Set by this command: [read_sdp_file](#)

justify_by

`justify_by {SW | NW | SE | NE | W | E | N | S | MID}`

Default: SW

Read-only `sdp_row` attribute. Returns the justifyBY constraint of the row. If no constraint value was specified for the row in the SDP file, the value defaults to SW.

Related Information

Set by this command: [read_sdp_file](#)

orient

`orient {R0|R90|R180|R270|MX|MY|MY90|MX90}`

Default: R0

Read-only `sdp_row` attribute. Returns the orientation of the SDP row. If no orientation value was specified for the row in the SDP file, the value defaults to R0.

Related Information

Set by this command: [read_sdp_file](#)

size_same

`size_same {false | true}`

Default: false

Read-write `sdp_row` attribute. Specifies whether all instances in the row have the same width as the widest instance in the row.

Related Information

Set by this command: [read_sdp_file](#)

skip_value

`skip_value integer`

Default: 0

Read-only `sdp_row` attribute. Specifies the number of rows to skip. This attribute can only have a non-zero value for a column called `skip_row_x`.

Example

The following example lists the attributes for all rows in column `cb`. Column `cb` has two rows specified in the SDP file: `rb` and `ra`. RTL Compiler creates a row instance with name `skip_row_x` for each row in the column. Following row `rb` is row `skip_row_0` (both rows have index 0). Following row `ra` is `skip_row_1` (both rows have index 1). The `skip_value` for `skip_row_0` is 4, which indicates that there are four empty rows between rows `rb` and `ra`.

```
rc:/designs/test/sdp_groups/ga/sdp_columns/cb/sdp_rows> ls -al
Total: 5 items
./
ra/                      (sdp_row)
    All attributes:
        flip =
        index = 1
        justify_by = N
        orient =
        size_same = false
        skip_value = 0
rb/                      (sdp_row)
    All attributes:
        flip = y
        index = 0
        justify_by = SW
        orient =
        size_same = false
        skip_value = 0
skip_row_0/              (sdp_row)
    All attributes:
        flip =
        index = 0
        justify_by = SW
        orient =
        size_same = false
        skip_value = 4
skip_row_1/              (sdp_row)
    All attributes:
        flip =
        index = 1
        justify_by = SW
        orient =
        size_same = false
        skip_value = 0
```

Attribute Reference for Encounter RTL Compiler

Physical—SDP Row Attributes

Related Information

Set by this command: [read_sdp_file](#)

Design for Manufacturing

Root Attributes

- [optimize_yield](#) on page 518

Design Attributes

- [yield](#) on page 519

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

optimize_yield

`optimize_yield {false | true}`

Default: false

Read-write `root` attribute. Sets RTL Compiler into yield optimization mode. You must set this attribute to `true` to use the design for manufacturing (DFM) flow.

Related Information

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

Affects these commands: [report gates -yield](#)

[report yield](#)

Affects this attribute: [yield on page 519](#)

Design Attributes

Contain information about the specified design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design
```

or

```
set_attribute attribute_name attribute_value [find /des* -design name]
```

- To get a design attribute value, type

```
get_attribute attribute_name design
```

yield

```
yield number
```

Read-only design attribute. Returns the total yield for the design.

Example

The following example finds the yield for the design penny:

```
rc:> get_attribute yield [find . -design penny]  
0.999983594076
```

Related Information

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

Affected by this command: [read_dfm](#)

Affects these commands: [report_gates -yield](#)
[report_yield](#)

Attribute Reference for Encounter RTL Compiler
Design for Manufacturing—Design Attributes

Constraint

Root Attributes

- [case analysis multi driver propagation](#) on page 527
- [case analysis sequential propagation](#) on page 528
- [define clock with new cost group](#) on page 529
- [drc first](#) on page 529
- [drc max cap first](#) on page 530
- [drc max fanout first](#) on page 531
- [drc max trans first](#) on page 533
- [enable break timing paths by mode](#) on page 534
- [enable data check](#) on page 535
- [fix min drcs](#) on page 536
- [ignore scan combinational arcs](#) on page 536
- [operating conditions](#) on page 537
- [override library max drc](#) on page 537
- [scale factor group path weights](#) on page 538
- [tim ignore data check for non endpoint pins](#) on page 538
- [time recovery arcs](#) on page 539
- [timing disable non sequential checks](#) on page 539
- [timing no path segmentation](#) on page 540
- [timing no path segmentation](#) on page 540
- [use multi clks latency uncertainty optimize](#) on page 541

Attribute Reference for Encounter RTL Compiler

Constraint—List

- [use multi clks latency uncertainty report](#) on page 541
- [wireload mode](#) on page 542
- [wireload selection](#) on page 543

Design Attributes

- [cell delay multiplier](#) on page 545
- [cell min delay multiplier](#) on page 546
- [force wireload](#) on page 547
- [ideal seq async pins](#) on page 548
- [ignore library drc](#) on page 549
- [ignore library max fanout](#) on page 550
- [latch borrow](#) on page 551
- [latch borrow by mode](#) on page 552
- [latch max borrow](#) on page 554
- [latch max borrow by mode](#) on page 555
- [max capacitance](#) on page 557
- [max fanout](#) on page 558
- [max transition](#) on page 559
- [timing disable internal inout net arcs](#) on page 560

Mode Attributes

- [default](#) on page 561

Instance Attributes

- [cell delay multiplier](#) on page 562
- [cell min delay multiplier](#) on page 563
- [disabled arcs](#) on page 564
- [disabled arcs by mode](#) on page 565

Attribute Reference for Encounter RTL Compiler

Constraint—List

- [hard_region](#) on page 567
- [latch_borrow](#) on page 568
- [latch_borrow_by_mode](#) on page 569
- [latch_max_borrow](#) on page 571
- [latch_max_borrow_by_mode](#) on page 572

Pin Attributes

- [break_timing_paths](#) on page 574
- [break_timing_paths_by_mode](#) on page 576
- [clock_hold_uncertainty](#) on page 578
- [clock_hold_uncertainty_by_mode](#) on page 579
- [clock_network_early_latency](#) on page 580
- [clock_network_early_latency_by_mode](#) on page 582
- [clock_network_late_latency](#) on page 584
- [clock_network_late_latency_by_mode](#) on page 586
- [clock_setup_uncertainty](#) on page 588
- [clock_setup_uncertainty_by_mode](#) on page 589
- [clock_source_early_latency](#) on page 590
- [clock_source_early_latency_by_mode](#) on page 592
- [clock_source_late_latency](#) on page 594
- [clock_source_late_latency_by_mode](#) on page 596
- [hold_uncertainty_by_clock](#) on page 598
- [ideal_driver](#) on page 599
- [ideal_network](#) on page 600
- [latch_max_borrow](#) on page 601
- [latch_max_borrow_by_mode](#) on page 602
- [network_early_latency_by_clock](#) on page 604

Attribute Reference for Encounter RTL Compiler

Constraint—List

- [network late latency by clock](#) on page 605
- [setup uncertainty by clock](#) on page 607
- [source early latency by clock](#) on page 609
- [source late latency by clock](#) on page 610
- [timing case logic value](#) on page 612
- [timing case logic value by mode](#) on page 613

Port Attributes

- [break timing paths](#) on page 616
- [break timing paths by mode](#) on page 617
- [clock hold uncertainty](#) on page 619
- [clock hold uncertainty by mode](#) on page 620
- [clock network early latency](#) on page 621
- [clock network early latency by mode](#) on page 623
- [clock network late latency](#) on page 625
- [clock network late latency by mode](#) on page 627
- [clock setup uncertainty](#) on page 629
- [clock setup uncertainty by mode](#) on page 631
- [clock source early latency](#) on page 632
- [clock source early latency by mode](#) on page 634
- [clock source late latency](#) on page 636
- [clock source late latency by mode](#) on page 638
- [external driven pin fall](#) on page 640
- [external driven pin rise](#) on page 640
- [external driver](#) on page 641
- [external driver from pin](#) on page 641
- [external driver input slew](#) on page 642

Attribute Reference for Encounter RTL Compiler

Constraint—List

- [external_fanout_load](#) on page 643
- [external_non_tristate_drivers](#) on page 643
- [external_pin_cap](#) on page 644
- [external_resistance](#) on page 644
- [external_wire_cap](#) on page 645
- [external_wire_res](#) on page 645
- [external_wireload_fanout](#) on page 646
- [external_wireload_model](#) on page 646
- [fixed_slew](#) on page 647
- [hold_uncertainty_by_clock](#) on page 648
- [ideal_driver](#) on page 649
- [ideal_network](#) on page 649
- [ignore_external_driver_drc](#) on page 650
- [max_capacitance](#) on page 651
- [max_fanout](#) on page 652
- [max_transition](#) on page 653
- [network_early_latency_by_clock](#) on page 654
- [network_late_latency_by_clock](#) on page 655
- [setup_uncertainty_by_clock](#) on page 657
- [source_early_latency_by_clock](#) on page 659
- [source_late_latency_by_clock](#) on page 660
- [timing_case_logic_value](#) on page 663
- [timing_case_logic_value_by_mode](#) on page 664

Subdesign Attributes

- [force_wireload](#) on page 666
- [hard_region](#) on page 667

Attribute Reference for Encounter RTL Compiler

Constraint—List

Clock Attributes

- [clock_hold_uncertainty](#) on page 668
- [clock_network_early_latency](#) on page 669
- [clock_network_late_latency](#) on page 670
- [clock_setup_uncertainty](#) on page 672
- [clock_source_early_latency](#) on page 673
- [clock_source_late_latency](#) on page 675
- [inverted_sources](#) on page 677
- [latch_max_borrow](#) on page 677
- [non_inverted_sources](#) on page 677
- [slew](#) on page 678

Cost Group Attributes

- [weight](#) on page 679

Exception Attributes

- [max](#) on page 681
- [user_priority](#) on page 681

External Delay Attributes

- [clock_network_latency_included](#) on page 682
- [clock_source_latency_included](#) on page 683

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

case_analysis_multi_driver_propagation

```
case_analysis_multi_driver_propagation {favor_neither | favor_0 | favor_1 | none}
```

Default: favor_neither

Read-write `root` attribute. Controls how case analysis values are propagated across nets that have multiple drivers. There are four possible settings:

- `none`: If one or more of the drivers of a multi-driven net are determined to be logic constants for the purposes of timing case analysis, then the constant value is not propagated to the loads of the net.

The following three settings will cause RTL Compiler to propagate a timing case logic constant value to the loads of a multi-driven net. The following settings affect how RTL Compiler behaves when one or more drivers has a value of 1 and one or more drivers has a value of 0 (conflicting case values).

- `favor_0`: The conflict is resolved by propagating a case value of 0 to the loads.
- `favor_1`: The conflict is resolved by propagating a case value of 1 to the loads.
- `favor_neither`: The conflict is resolved by not propagating a case value to the loads.

Example

The following example shows how the various settings will affect the computation of a timing case value at the loads of the net in a case with a net with four drivers:

Attribute Reference for Encounter RTL Compiler

Constraint—Root Attributes

■ none

```
driver case values    load case value
-----
x  1  x  x          x
1  1  x  x          x
0  x  x  x          x
1  x  0  0          x
```

■ favor_0

```
driver case values    load case value
-----
x  1  x  x          1
1  1  x  x          1
0  x  x  x          0
1  x  0  0          0  <--
```

■ favor_1

```
driver case values    load case value
-----
x  1  x  x          1
1  1  x  x          1
0  x  x  x          0
1  x  0  0          1  <--
```

■ favor_neither

```
driver case values    load case value
-----
x  1  x  x          1
1  1  x  x          1
0  x  x  x          0
1  x  0  0          x  <--
```

case_analysis_sequential_propagation

```
case_analysis_sequential_propagation {false | true}
```

Default: false

Read-write root attribute. Indicates whether timing case analysis should propagate logic constants through sequential cells.

Example

```
rc:/> set_attribute case_analysis_sequential_propagation false /
```

define_clock_with_new_cost_group

```
define_clock_with_new_cost_group {false | true}
```

Default: false

Read-write root attribute. Controls whether a new cost group can be created for each clock defined with the `define_clock` command. Set this attribute to `true` to create cost groups.

Related Information

Affects these commands:

[define_clock](#)

[report_timing](#)

[synthesize](#)

drc_first

```
drc_first {false | true}
```

Default: false

Read-write root attribute. Specifies whether to give *all* design rule constraints higher priority than the timing constraints.

The design rule constraints are optimized in the following order:

1. `max_transition`
2. `max_capacitance`
3. `max_fanout`

Related Information

[Making Design Rule Constraints \(DRC\) the Highest Priority in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affects these commands:

[report_design_rules](#)

[report_timing](#)

[synthesize -incremental](#)

Related attributes:

[drc_max_cap_first](#) on page 530
[drc_max_fanout_first](#) on page 531
[drc_max_trans_first](#) on page 533
[ignore_library_drc](#) on page 549
[map_drc_first](#) on page 791
(design) [max_capacitance](#) on page 557
(libpin) [max_capacitance](#) on page 220
(port) [max_capacitance](#) on page 651
(design) [max_fanout](#) on page 558
(libpin) [max_fanout](#) on page 220
(port) [max_fanout](#) on page 652
(design) [max_transition](#) on page 653
(libpin) [max_transition](#) on page 220
(port) [max_transition](#) on page 653

drc_max_cap_first

`drc_max_cap_first {false | true}`

Default: `false`

Read-write [root](#) attribute. Specifies whether to give the maximum capacitance cost higher priority than the timing constraints.

To optimize the maximum capacitance cost before the timing constraints, set the `drc_first` attribute to `false` and this attribute to `true`. In this case, the two other design rule constraints are optimized after the timing constraints have been taken into account.

Note: If you set the `drc_max_cap_first`, `drc_max_fanout_first`, and `drc_max_trans_first` attributes to `true`, all design rule constraints are optimized before the timing constraints and they are optimized in the following order:

1. `max_transition`
2. `max_capacitance`
3. `max_fanout`

Related Information

[Making Design Rule Constraints \(DRC\) the Highest Priority in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affects these commands:	report design rules report timing synthesize -incremental
Affected by this attribute:	drc_first on page 529
Related attributes:	drc_max_fanout_first on page 531 drc_max_trans_first on page 533 ignore_library_drc on page 549 (design) max_capacitance on page 557 (libpin) max_capacitance on page 220 (port) max_capacitance on page 651 (design) max_fanout on page 558 (libpin) max_fanout on page 220 (port) max_fanout on page 652 (design) max_transition on page 653 (libpin) max_transition on page 220 (port) max_transition on page 653

drc_max_fanout_first

`drc_max_fanout_first {false | true}`

Default: false

Read-write [root](#) attribute. Specifies whether to give the maximum fanout cost higher priority than the timing constraints.

To optimize the maximum fanout cost before the timing constraints, set the `drc_first` attribute to `false` and this attribute to `true`. In this case, the two other design rule constraints are optimized after the timing constraints have been taken into account.

Attribute Reference for Encounter RTL Compiler

Constraint—Root Attributes

Note: If you set the `drc_max_cap_first`, `drc_max_fanout_first`, and `drc_max_trans_first` attributes to `true`, all design rule constraints are optimized before the timing constraints and in the following order:

1. `max_transition`
2. `max_capacitance`
3. `max_fanout`

Related Information

[Making Design Rule Constraints \(DRC\) the Highest Priority in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affects these commands:

[report design_rules](#)

[report timing](#)

[synthesize -incremental](#)

Affected by this attribute:

[drc_first](#) on page 529

Related attributes:

[drc_max_cap_first](#) on page 530

[drc_max_trans_first](#) on page 533

[ignore_library_drc](#) on page 549

(design) [max_capacitance](#) on page 557

(libpin) [max_capacitance](#) on page 220

(port) [max_capacitance](#) on page 651

(design) [max_fanout](#) on page 558

(libpin) [max_fanout](#) on page 220

(port) [max_fanout](#) on page 652

(design) [max_transition](#) on page 653

(libpin) [max_transition](#) on page 220

(port) [max_transition](#) on page 653

drc_max_trans_first

`drc_max_trans_first {false | true}`

Default: false

Read-write root attribute. Specifies whether to consider the maximum transition cost before the timing constraints.

To optimize the maximum transition cost before the timing constraints, set the `drc_first` attribute to `false` and this attribute to `true`. In this case, the two other design rule constraints are optimized after the timing constraints have been taken into account.

Note: If you set the `drc_max_cap_first`, `drc_max_fanout_first`, and `drc_max_trans_first` attributes to `true`, the design rule constraints are optimized in the following order:

1. `max_transition`
2. `max_capacitance`
3. `max_fanout`

Related Information

Making Design Rule Constraints (DRC) the Highest Priority in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler

Affects these commands:

[report design rules](#)

[report timing](#)

[synthesize -incremental](#)

Affected by this attribute:

[drc_first](#) on page 529

Related attributes:

[drc_max_cap_first](#) on page 530

[drc_max_fanout_first](#) on page 531

[ignore_library_drc](#) on page 549

(design) [max_capacitance](#) on page 557

(libpin) [max_capacitance](#) on page 220

(port) [max_capacitance](#) on page 651

(design) [max_fanout](#) on page 558

(libpin) [max_fanout](#) on page 220

- (port) [max_fanout](#) on page 652
- (design) [max_transition](#) on page 653
- (libpin) [max_transition](#) on page 220
- (port) [max_transition](#) on page 653

enable_break_timing_paths_by_mode

`enable_break_timing_paths_by_mode {true | false}`

Default: true

Read-write [root](#) attribute. When in multi mode, indicates whether timing paths can be broken by mode to honour different timing constraints by mode, such as clock gating checks.



Do not set this attribute to `false` unless the tool issued a warning message that instructs you to change the setting.

Related Information

Affects these commands:	report_clocks report_qor read_sdc report_timing write_sdc write_script
Related commands:	multi_cycle path_adjust path_delay path_disable specify_paths

Related attributes:	(pin) break_timing_paths on page 574 (port) break_timing_paths on page 616 (pin) break_timing_paths_by_mode on page 576 (port) break_timing_paths_by_mode on page 617 (instance) disabled_arcs on page 564 (instance) disabled_arcs_by_mode on page 565 (pin) timing_case_computed_value_by_mode on page 985 (instance) timing_case_disabled_arcs on page 953 (instance) timing_case_disabled_arcs_by_mode on page 954 (pin) timing_case_logic_value on page 612 (port) timing_case_logic_value on page 663 (pin) timing_case_logic_value_by_mode on page 613 (port) timing_case_logic_value_by_mode on page 664
---------------------	--

enable_data_check

`enable_data_check {true | false}`

Default: true

Read-write [root](#) attribute. When set to true, the data-to-data checks (specified through the `set_data_check` SDC command or in the library) are enabled.

Note: You should set this attribute before reading in the design.

Related Information

[Setting Data-to-Data Checks](#) in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affects these commands:	report_timing synthesize
-------------------------	---

Attribute Reference for Encounter RTL Compiler

Constraint—Root Attributes

Affects these attributes:	(pin) has_min_delay on page 971 (port) has_min_delay on page 1009 (pin) min_slew on page 972 (port) min_slew on page 1010 (pin) min_timing_arcs on page 973
Related attributes:	timing_disable_non_sequential_checks on page 539 (libarc) type on page 230

fix_min_drcs

`fix_min_drcs {false | true}`

Default: false

Read-write [root](#) attribute. When set to true, fixes the minimum design rule costs based on calculations in the library. Specifically, RTL Compiler fixes the minimum capacitance, minimum transition, and minimum fanout design rule violations by resizing the driver and its loads.

Related Information

Affects this command: [synthesize -incremental](#)

ignore_scan_combinational_arcs

`ignore_scan_combinational_arcs {true | false}`

Default: true

Read-write [root](#) attribute. Controls whether to ignore the combinational arcs from scan input pins to output pins of scan flip-flops during timing analysis and optimization.

Example

```
rc:/> set_attribute ignore_scan_combinational_arcs false /
```

Related Information

Affects these commands: [report_timing](#)
[synthesize](#)

operating_conditions

`operating_conditions string`

Read-write root attribute. Specifies the operating conditions to use for timing. You must specify the path to an `operating_conditions` object in the library. The `operating_conditions` attribute does not need to have a value if you want to use the default operating conditions.

Specify this attribute before you use the libraries in the design.

Example

The following command sets the operating conditions to worst_case.

```
rc:/> set_attribute operating_conditions worst_case /
```

Related Information

Specifying Operating Conditions in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.

Related attribute: (library_domain) **operating_conditions** on page 1429

override_library_max_drc

```
override_library_max_drc {false | true}
```

Default: false

Read-write `root` attribute. Specifies whether you can relax the design rule constraints set on the library. Set this attribute to `true` to relax the `max_capacitance`, `max_fanout`, and `max_transition` constraints. By default, you can only tighten the constraint values.

Related Information

Affects these commands:

report design rules

synthesize -incremental

Affects these attributes:

(libpin) max capacitance on page 220

(libpin) max_fanout on page 220

(libpin) max transition on page 220

scale_factor_group_path_weights

`scale_factor_group_path_weights integer`

Default: 1

Read-write root attribute. Scales the weights specified in any `group_path` SDC command(s) .

Example

Consider the following line in the SDC file:

```
group_path -weight 10 -name m -from in
```

The following command scales the weight of cost group `m` by 6.

```
set_attribute scale_factor_group_path_weights 6 /
```

As a result the weight of cost group `m` is now 60.

Related Information

[Cost Group Examples in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Related command: [dc::group_path](#)

tim_ignore_data_check_for_non_endpoint_pins

```
tim_ignore_data_check_for_non_endpoint_pins {true | false  
| sdc_set_data_check_only | lib_non_seq_setup_only}
```

Default: true

Read-write root attribute. Controls which data-to-data timing constraints are applied if the constrained pins (`to_pin`) are *not* endpoints. You can specify the following values:

<code>false</code>	Takes all data-to-data constraints into account, even when the constrained pin is not an endpoint.
<code>lib_non_seq_setup_only</code>	Specifies to only ignore constraints on instances whose library cells have timing arcs of type <code>non_seq_setup_rising</code> or <code>non_seq_setup_falling</code> .

Attribute Reference for Encounter RTL Compiler

Constraint—Root Attributes

`sdc_set_data_check_only` Specifies to only ignore constraints specified with the `set_data_check` SDC command.

`true` Ignores all data-to-data constraints into account.

This requirement avoids breaking timing paths at the *to_pin*.

time_recovery_arcs

`time_recovery_arcs {false | true}`

Default: `false`

Read-write root attribute. When set to `true`, all paths to the asynchronous pin of a flip-flop become constrained to the recovery arc of the flip-flop. This attribute has no affect on those asynchronous inputs of flip-flops that are not modeled with recovery arcs. This attribute does not change the ideal setting (the value of `causes_ideal_net`) of the asynchronous pin's net.

Example

```
rc:/> set_attribute time_recovery_arcs true /
```

Related Information

Affects these commands: [report_timing](#)
 [synthesize](#)

Related attributes: (pin) [causes_ideal_net](#) on page 962
 (port) [causes_ideal_net](#) on page 1003
 (subport) [causes_ideal_net](#) on page 1039

timing_disable_non_sequential_checks

`timing_disable_non_sequential_checks {true|false}`

Default: `true`

Read-write root attribute. Controls whether the `non_seq_setup_rising` and `non_seq_setup_falling` pin timing attributes specified in the library are taken into account. Set this attribute to `false` to take these timing arcs into account.

Related Information

[Setting Data-to-Data Checks](#) in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affects these commands:	report_timing synthesize
Affects these attributes:	(pin) has_min_delay on page 971 (port) has_min_delay on page 1009 (pin) min_slew on page 972 (port) min_slew on page 1010 (pin) min_timing_arcs on page 973
Affected by this attribute:	enable_data_check on page 535
Related attribute:	(libarc) type on page 230

timing_no_path_segmentation

`timing_no_path_segmentation list_of_timing_checks`

Read-write [root](#) attribute. Prevents breaking of the timing path at the `-to` pin of the specified timing checks. You can specify a combination of the following checks: `set_max_delay`, `clock_gating`, and `set_data_check`. Set this attribute before you read in the SDC constraints.

Example

The following command prevents breaking of the timing path for the `set_max_delay` and `set_data_check` timing checks:

```
set_attribute timing_no_path_segmentation {set_max_delay set_data_check} /
```

Related Information

[Setting Data-to-Data Checks](#) in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affects these commands:	report_timing synthesize
-------------------------	---

use_multi_clks_latency_uncertainty_optimize

`use_multi_clks_latency_uncertainty_optimize {false | true}`

Default: false

Read-write [root](#) attribute. Enables optimization to take into account clock-specific setup uncertainties, and network and source latencies on the pins and ports.

Related Information

Affects these commands:	report_timing synthesize
Affects these attributes:	(pin) setup uncertainty by clock on page 607 (port) setup uncertainty by clock on page 657 (pin) network late latency by clock on page 605 (port) network late latency by clock on page 655 (pin) source late latency by clock on page 610 (port) source late latency by clock on page 660

use_multi_clks_latency_uncertainty_report

`use_multi_clks_latency_uncertainty_report {false | true}`

Default: false

Read-write [root](#) attribute. Enables the timing analysis engine to take into account clock-specific setup uncertainties, and clock-specific network and source latencies on the pins and ports.

Related Information

Affects these commands:	report_timing synthesize
Affects these attributes:	(pin) setup uncertainty by clock on page 607 (port) setup uncertainty by clock on page 657 (pin) network late latency by clock on page 605

(port) [network late latency by clock](#) on page 655

(pin) [source late latency by clock](#) on page 610

(port) [source late latency by clock](#) on page 660

wireload_mode

wireload_mode {enclosed| segmented | top | none}

Read-write [root](#) attribute. Allows you to override the computing net load method that is normally specified in the technology library. The wire-load modes are:

enclosed	Uses the wire-load model of the smallest block that fully encloses the net to compute the load of the net. Hierarchical boundary pins are not counted as fanouts.
segmented	Divides nets that cross hierarchical boundaries into segments with one segment for each level of hierarchy. Separate load values are computed for each segment (counting the hierarchical boundary pins as individual fanouts) and the load values are added together.
top	Uses the wire-load model of the top-level design for all nets in all subdesigns. Hierarchical boundary pins are not counted as fanouts.

By default, this attribute is set to the value of `default_wire_load_mode` Liberty attribute of the library that was read in. If the `default_wire_load_mode` attribute was not set in the library, the `wireload_mode` attribute defaults to `enclosed`.

If multiple libraries are read in, the tool checks for a library that has the `default_wire_load_mode` attribute set and the `wireload_mode` attribute is set to that value. If multiple libraries have this attribute set, the `wireload_mode` attribute is set to the value defined in the library that was read in first. If the `default_wire_load_mode` attribute is not found in any of the libraries, the `wireload_mode` attribute defaults to `enclosed`.

Note: When the `interconnect_mode` attribute is set to `ple`, the `wireload_mode` attribute is set to `none`. In this case, the tool uses physical layout estimators instead of wire-load models during synthesis.

Related Information

[Specifying Wire-load Models in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affected by this attribute: [interconnect mode](#) on page 377

Related attributes: [force_wireload](#) on page 547

[wireload_selection](#) on page 543

wireload_selection

wireload_selection {default | table | none}

Default: default

Read-write [root](#) attribute. Indicates whether to use a wire-load selection table to choose default wire-load models for blocks based on their cell areas.

default RTL Compiler behaves as if the attribute had never been set.
 The environment reverts to the default settings.

none Specifies not to perform automatic wire-load selection by area.
 The only wire-load models that will be used are the ones that
 are set with the [force_wireload](#) attribute on individual
 modules or the default wireload model specified in the library.

table Specifies the wire-load selection table to use. Specify the
 hierarchical path to the wire-load selection table to be used. You
 can obtain the path using the [find](#) command.

Example

- Some libraries contain multiple selection tables, such as for different numbers of metal layers), and in such cases you can indicate which [wireload_selection](#) table should be used:

```
rc:/> set_attribute wireload_selection \
      ==> [find / -wireload_selection "4_layer"]
```

- The following example chooses a different selection group:

```
rc:/> set_attribute wireload_selection wc_group /
```

Attribute Reference for Encounter RTL Compiler

Constraint—Root Attributes

Related Information

[Specifying Wire-load Models](#) in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*.

Related attributes:

- [force_wireload](#) on page 547
- [wireload_mode](#) on page 542
- (library_domain) [wireload_selection](#) on page 1430

Design Attributes

Contain information about the specified design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design  
or
```

```
set_attribute attribute_name attribute_value [find /des* -design name ]
```

- To get a design attribute value, type

```
get_attribute attribute_name design
```

cell_delay_multiplier

`cell_delay_multiplier float`

Default: 1.0

Read-write `design` attribute. Scales the delay of the design by the specified value. If a scaling factor is specified for an instance, that value will be used to scale the delay of that instance. If no instance attribute was specified, but a libcell attribute was specified, that value will be used to scale the delay of the corresponding instances.

This attribute can also be set by the following SDC command:

```
dc::set_timing_derate -late -data -cell_delay design
```

Example

Assume the `cell_delay_multiplier` attribute for the design is set to 1.5, the `cell_delay_multiplier` attribute for libcell `INVX4` is set to 1.4, and the corresponding attribute for instance `I54` of that libcell is set to 1.0. All instances of the design will be derated by a value of 1.5, except for the instances of libcell `INVX4` which will have a value of 1.4. Instance `I54` will have no derating with a value of 1.0.

Related Information

Affects these commands:

[report cell_delay_calculation](#)

[report slew_calculation](#)

[report timing](#)

Related attributes:

(libcell) [cell_delay_multiplier](#) on page 191

(instance) [cell_delay_multiplier](#) on page 562

cell_min_delay_multiplier

`cell_min_delay_multiplier float`

Default: 1.0

Read-write design attribute. Scales the minimum delay of the design by the specified value. This scaled delay is used to compute the input delay to the from_pin during data-to-data checking. If a scaling factor is specified for an instance, that value will be used to scale the delay of that instance. If no instance attribute was specified, but a libcell attribute was specified, that value will be used to scale the delay of the corresponding instances.

This attribute can also be set by the following SDC command:

```
dc::set_timing_derate -early -data -cell_delay design
```

Example

Assume the `cell_min_delay_multiplier` attribute for the design is set to 1.5, the `cell_min_delay_multiplier` attribute for libcell INVX4 is set to 1.4, and the corresponding attribute for instance I55 of that libcell is set to 1.2. The minimum delay of all instances of the design will be derated by a value of 1.5, except for the instances of libcell INVX4 which will have a value of 1.4. The minimum delay of instance I55 will be scaled with a value of 1.2.

Related Information

[Setting Data-to-Data Checks](#) in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affects these commands: [report cell_delay calculation](#)

[report slew calculation](#)

[report timing](#)

Related attributes: (libcell) [cell_min_delay_multiplier](#) on page 193

 (instanc) [cell_min_delay_multiplier](#) on page 563

force_wireload

```
force_wireload {auto_select | custom_wireload | none | inherit}
```

Default: auto_select

Read-write design attribute. Forces RTL Compiler to use the specified wire-load model.

auto_select	Automatically selects wire-load models according to the wire-load selection table or default wire-load model in the technology library.
custom_wireload	Forces RTL Compiler to use the specified custom wire-load model. Specify the hierarchical path to the wire-load model to be used. You can obtain the path using the <u>find</u> command.
inherit	Causes the same behavior as <code>auto_select</code> .
none	Prevents use of any wire-load models.

Note: When you set this attribute on the design it does not affect any subdesigns on which this attribute was set, unless the value on the subdesign was set to `inherit`.

Examples

- The following example specifies a wire-load model from a library when multiple libraries are loaded:

```
rc:/> set_attribute force_wireload [find [find / -library lib_name] \
-wireload "10x10"] [find / -design test_top]
```

- The following example specifies the 1x1 wire-load model on the chekov design:

```
rc:/> set_attribute force_wireload 1x1 chekov
```

Related Information

Specifying a Wire-load Model in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*.

Affects these attributes: [\(subdesign\) force_wireload](#) on page 666

[wireload](#) on page 945

Related attributes: [wireload_mode](#) on page 542

[wireload_selection](#) on page 543

ideal_seq_async_pins

```
ideal_seq_async_pins {true | false}
```

Default: true

Read-write design attribute. This attribute affects those nets that have both asynchronous and synchronous loads. If this attribute is set to `true`, RTL Compiler will only treat the asynchronous loads as ideal (0 capacitance) and not the entire net. Only the synchronous loads will be used to compute the total load capacitance of these nets during timing and optimization. For example, if there are 20 asynchronous reset loads on a net and each has a load capacitance of 100 and there is also two D-pin loads and each has a load capacitance of 150, the total load capacitance of the net is:

$$2 * 150 = 300$$

When a net has an inverter or buffer tree that in turn drives the synchronous or asynchronous loads, then any asynchronous pin of a register is considered to be an asynchronous load. Any inverter or buffer that drives **only** other asynchronous loads is itself considered to be an asynchronous load for its driver net. Thus, if a buffer or inverter tree drives asynchronous or synchronous loads, the asynchronous selection process is propagated backwards through the chain of buffers or inverter.

Example

```
rc:/> set_attribute ideal_seq_async_pins false /designs/designname
```

Related Information

[Nets with Asynchronous and Synchronous Loads in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:

- (pin) [causes_ideal_net](#) on page 962
- (port) [causes_ideal_net](#) on page 1003
- (subport) [causes_ideal_net](#) on page 1039
- (pin) [ideal_driver](#) on page 599
- (port) [ideal_driver](#) on page 649

ignore_library_drc

`ignore_library_drc {false | true}`

Default: false

Read-write [design](#) attribute. Controls the use of design rule constraints from the technology library. When set to `true`, forces RTL Compiler to ignore any design rule constraints specified in the technology library. You can use this attribute in conjunction with the `max_capacitance`, `max_fanout`, and `max_transition` attributes to set looser design rule constraints than the ones appearing in the technology library. Use the `ignore_library_drc` attribute with care. The `ignore_library_drc` attribute applies to the top-level module and all subdesigns.

Example

```
rc:/> set_attribute ignore_library_drc true /designs/designname
```

Related Information

[Controlling the Use of Design Rule Constraints From the Technology Library in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affects these commands:

[report design_rules](#)

[synthesize -incremental](#)

Affects these attributes:

[drc_first](#) on page 529

[ignore_library_max_fanout](#) on page 550

(design) [max_capacitance](#) on page 557

(port) [max_capacitance](#) on page 651

(design) [max_fanout](#) on page 558

(port) [max_fanout](#) on page 652

(design) [max_transition](#) on page 653

(port) [max_transition](#) on page 653

ignore_library_max_fanout

`ignore_library_max_fanout {false | true}`

Default: false

Read-write design attribute. Determines whether to use the maximum fanout design rule limits from the technology library.

Related Information

Affects these commands:	report design rules synthesize -incremental
Affected by this attribute:	ignore_library_drc on page 549
Affects this attribute:	(design) max_fanout on page 558 (port) max_fanout on page 652
Related attributes:	drc_first on page 529 (design) max_capacitance on page 557 (port) max_capacitance on page 651 (design) max_fanout on page 558 (port) max_fanout on page 652 (design) max_transition on page 653 (port) max_transition on page 653

latch_borrow

`latch_borrow {no_value | float}`

Default: no_value

Read-write design attribute. Specifies the time (in ps) borrowed from the next clock cycle. If set to no_value, the latch borrow values are computed dynamically.

Example

The following example sets a `latch_borrow` value of 300 p.s:

```
rc:/> set_attr latch_borrow 300 [find ./ -design test_top]
```

Related Information

Specifying Latch Time Borrow Values in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*.

Affected by this attribute: [latch_borrow_by_mode](#) on page 552

Related attribute: (instance) [latch_borrow](#) on page 568

latch_borrow_by_mode

```
latch_borrow_by_mode {mode_name_1 delay_value}
[ {mode_name_2 delay_value} ] ...
```

Read-write [design](#) attribute. Specifies the time borrowed from the next clock cycle for latches in the netlist for all the timing modes. Valid only for latches and timing models. The value must be non-negative. Specify a Tcl list of modes and corresponding delay values.

When the `latch_borrow_by_mode` attribute is set, the `latch_borrow` attribute will be reset to `no_value`. However, when all the values for the `latch_borrow_by_mode` attribute are the same, the `latch_borrow` attribute is set to that value.

When the `latch_borrow` attribute is set, the `latch_borrow_by_mode` attribute reflects that value for each mode.

Note: If this attribute is set on a design and on an instance, the smaller time borrow value will have priority.

Examples

- The following example sets this attribute for the `my_top` design in modes `a` and `b`:

```
rc:/> set_attribute latch_borrow_by_mode {{a 20} {b 5}} /designs/my_top
rc:/> get_attribute latch_borrow_by_mode /designs/my_top
{/designs/latch/modes/b 5.0} {/designs/latch/modes/a 20.0}
```

- The following example shows that the value of this attribute is a list of lists that gets overwritten each time it is set:

```
rc:/> get_attribute latch_borrow_by_mode /designs/my_top
{/designs/latch/modes/b 20.0}
rc:/> set_attribute latch_borrow_by_mode {{a 20}} /designs/my_top
rc:/> get_attribute latch_borrow_by_mode /designs/my_top
{/designs/latch/modes/a 20.0}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:	report clocks report qor report timing write encounter write sdc
-------------------------	--

Attribute Reference for Encounter RTL Compiler

Constraint—Design Attributes

[write_script](#)

Related commands:

[create_mode](#)

[read_sdc](#)

Affects this attribute

(design) [latch_borrow](#) on page 551

Related attributes:

[disabled_arcs_by_mode](#) on page 565

(pin) [external_delays_by_mode](#) on page 969

(port) [external_delays_by_mode](#) on page 1007

(design) [latch_borrow](#) on page 551

(instance) [latch_borrow](#) on page 568

(instance) [latch_borrow_by_mode](#) on page 569

(instance) [latch_max_borrow](#) on page 571

(pin) [latch_max_borrow](#) on page 601

(clock) [latch_max_borrow](#) on page 677

(instance) [latch_max_borrow_by_mode](#) on page 572

(pin) [latch_max_borrow_by_mode](#) on page 602

(pin) [propagated_clocks_by_mode](#) on page 976

(port) [propagated_clocks_by_mode](#) on page 1015

(design) [slack_by_mode](#) on page 943

(pin) [slack_by_mode](#) on page 980

(port) [slack_by_mode](#) on page 1018

(cost group) [slack_by_mode](#) on page 1060

(pin) [timing_case_computed_value_by_mode](#) on page 985

(port) [timing_case_computed_value_by_mode](#) on page 1021

instance) [timing_case_disabled_arcs](#) on page 953

(instance) [timing_case_disabled_arcs_by_mode](#) on page 954

(pin) [timing_case_logic_value](#) on page 612

(port) [timing_case_logic_value_by_mode](#) on page 664

latch_max_borrow

`latch_max_borrow {no_value | integer}`

Default: no_value

Read-write [design](#) attribute. Specifies the maximum amount of time that can be borrowed, in picoseconds, from the next clock cycle. The specified value must be an integer greater or equal to 0 (decimal values are rounded to the nearest integer) or no_value. This attribute is available on latch cells, clocks, clock (enable) pin, data pin, or on a design. If the attribute is set on multiple objects that overlap each other, for example a latch instance and its data pin, the minimum value will be taken. If the [latch_borrow](#) attribute has already been set, then the [latch_max_borrow](#) attribute is ignored.

Related Information

[Specifying Latch Time Borrow Values in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affected by these attributes:	(design) latch_borrow on page 551 (design) latch_max_borrow_by_mode on page 555 (clock) latch_max_borrow on page 677 (instance) latch_max_borrow on page 571 (pin) latch_max_borrow on page 601
Related attributes:	(instance) latch_borrow on page 568 (instance) latch_max_borrow_by_mode on page 572 (pin) latch_max_borrow_by_mode on page 602 (design) latch_borrow_by_mode on page 552 (instance) latch_borrow_by_mode on page 569

latch_max_borrow_by_mode

```
latch_max_borrow_by_mode {mode_name_1 delay_value}  
  [{mode_name_2 delay_value}]...
```

Read-write design attribute. Specifies the maximum time borrowed, for a design, from the next clock cycle for latches in a netlist for all timing modes. Valid only for latches and timing models. The value must be non-negative. Specify a Tcl list of modes and corresponding delays.

When the `latch_max_borrow_by_mode` attribute is set, the `latch_max_borrow` attribute will be reset to `no_value`. However, when all the values for the `latch_max_borrow_by_mode` attribute are the same, the `latch_max_borrow` attribute is set to that value.

When the `latch_max_borrow` attribute is set, the `latch_max_borrow_by_mode` attribute reflects that value for each mode.

Examples

- The following example shows that the value of this attribute is a list of lists that gets overwritten each time it is set:

```
rc:/> get_attribute latch_max_borrow_by_mode /designs/my_top/A  
{/designs/latch/modes/a 20000.0}  
rc:/> set_attribute latch_max_borrow_by_mode [list [list b 10000]] A  
{/designs/latch/modes/b 10000.0}
```

- The following example sets this attribute in modes a and b:

```
rc:/> set_attribute latch_max_borrow_by_mode [list [list b 10000] \  
[list a 20000]] A  
rc:/> get_attribute latch_max_borrow_by_mode A  
{/designs/latch/modes/b 10000.0} {/designs/latch/modes/a 20000.0}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands: [report clocks](#)
[report qor](#)
[report timing](#)
[write encounter](#)

Attribute Reference for Encounter RTL Compiler

Constraint—Design Attributes

[write_sdc](#)

[write_script](#)

Related commands:

[create_mode](#)

[read_sdc](#)

Affects these attributes

(design) [latch_max_borrow](#) on page 554

Related attributes:

[disabled_arcs_by_mode](#) on page 565

(pin) [external_delays_by_mode](#) on page 969

(port) [external_delays_by_mode](#) on page 1007

(design) [latch_borrow](#) on page 551

(instance) [latch_borrow](#) on page 568

(instance) [latch_borrow_by_mode](#) on page 569

(instance) [latch_max_borrow](#) on page 571

(pin) [latch_max_borrow](#) on page 601

(clock) [latch_max_borrow](#) on page 677

(instance) [latch_max_borrow_by_mode](#) on page 572

(pin) [latch_max_borrow_by_mode](#) on page 602

(pin) [propagated_clocks_by_mode](#) on page 976

(port) [propagated_clocks_by_mode](#) on page 1015

(design) [slack_by_mode](#) on page 943

(pin) [slack_by_mode](#) on page 980

(port) [slack_by_mode](#) on page 1018

(cost group) [slack_by_mode](#) on page 1060

(pin) [timing_case_computed_value_by_mode](#) on page 985

(port) [timing_case_computed_value_by_mode](#) on page 1021

instance) [timing_case_disabled_arcs](#) on page 953

(instance) [timing_case_disabled_arcs_by_mode](#) on page 954

(pin) [timing_case_logic_value](#) on page 612

(port) [timing_case_logic_value_by_mode](#) on page 664

max_capacitance

```
max_capacitance {no_value | float}
```

Default: no_value

Read-write [design](#) attribute. Specifies the maximum capacitance design rule limit in femtofarads for all nets in a design. The resolution is 1/10. When optimizing a design, RTL Compiler attempts to satisfy all design rule constraints. These constraints can come from attributes on a module or port, or from the technology library.

If set to no_value, no design-level constraint is applied, although design rules may still be inferred from port attributes or from the technology library.

Example

- The following example specifies a maximum capacitance limit for all nets in a design:

```
rc:/> set_attribute max_capacitance value /designs/my_top
```

Related Information

[Specifying the Maximum Capacitance Limit in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:

[report design_rules](#)

[report timing](#)

[synthesize -incremental](#)

Related attributes:

[drc_first](#) on page 529

[drc_max_cap_first](#) on page 530

[ignore_library_drc](#) on page 549

(libpin) [max_capacitance](#) on page 220

(port) [max_capacitance](#) on page 651

(design) [max_fanout](#) on page 558

(libpin) [max_fanout](#) on page 220

(port) [max_fanout](#) on page 652

(design) [max_transition](#) on page 653

(libpin) [max_transition](#) on page 220

(port) [max_transition](#) on page 653

max_fanout

```
max_fanout {no_value | float}
```

Default: no_value

Read-write [design](#) attribute. Specifies the maximum fanout design rule limit for all nets in a design. The resolution is 1/1000. When optimizing a design, RTL Compiler attempts to satisfy all design rule constraints. These constraints can come from attributes on a module or port, or from the technology library.

If set to no_value, no design-level constraint is applied, although design rules may still be inferred from port attributes or from the technology library.

Example

The following example specifies a maximum fanout design rule limit for all nets in design test:

```
rc:/> set_attribute max_fanout value /designs/test
```

Related information

[Specifying the Maximum Fanout Limit in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:

[report design rules](#)

[report timing](#)

[synthesize -incremental](#)

Related attributes:

[drc first](#) on page 529

[drc_max_fanout_first](#) on page 531

[ignore_library_drc](#) on page 549

[ignore_library_max_fanout](#) on page 550

(design) [max_capacitance](#) on page 557

(libpin) [max_capacitance](#) on page 220

(port) [max_capacitance](#) on page 651

(libpin) [max_fanout](#) on page 220

(port) [max_fanout](#) on page 652

(design) [max_transition](#) on page 653

(libpin) [max_transition](#) on page 220
(port) [max_transition](#) on page 653

max_transition

`max_transition {no_value | integer}`

Default: no_value

Read-write [design](#) attribute. Specifies a maximum transition design rule limit for all nets in a design. The resolution is 1. When optimizing a design, RTL Compiler attempts to satisfy all design rule constraints. These constraints can come from attributes on a module or port, or from the technology library.

If set to no_value, no design-level constraint is applied, although design rules may still be inferred from port attributes or from the technology library.

Example

- The following example specifies a maximum transition design rule limit for all nets in a design:

```
rc:/> set_attribute max_transition 5 /designs/my_top
```

Related Information

[Specifying the Maximum Transition Limit in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:

[report design rules](#)

[report timing](#)

[synthesize -incremental](#)

Related attributes:

[drc_first](#) on page 529

[drc_max_fanout_first](#) on page 531

[ignore_library_drc](#) on page 549

(design) [max_capacitance](#) on page 557

(libpin) [max_capacitance](#) on page 220

(port) [max_capacitance](#) on page 651

(design) [max_fanout](#) on page 558

(libpin) [max_fanout](#) on page 220

(port) [max_fanout](#) on page 652

(libpin) [max_transition](#) on page 220

(port) [max_transition](#) on page 653

timing_disable_internal_inout_net_arcs

`timing_disable_internal_inout_net_arcs {false | true}`

Default: `false`

Read-write [design](#) attribute. Determines whether to disable the path from internal drivers of multi-driven nets when the net is connected to a bidirectional I/O port.

If this attribute is set to `true`, the following applies to multi-driven nets connected to an I/O port:

- All timing paths starting from drivers other than the I/O port to loads of the net are disabled.
- The only valid paths are from drivers other than the I/O port to the I/O and from the I/O to the loads.
- The slew value at the input pin of the loads is propagated from the I/O ports.
- In case of multiple I/O ports, the worst slew among the I/O ports gets propagated to the input pin of the loads.

Related Information

Affects these commands: [synthesize](#)
[report timing](#)

Mode Attributes

Contain information about the design modes of the design. Mode objects are found in the mode directory in the design directory. These attributes are read-only attributes, so you cannot set their values.

- To get a mode attribute value, type

```
get_attribute attribute_name [find /des*/design -mode name]
```

default

default {false | true}

Default: false

Read-only mode attribute. Indicates whether the mode is the default mode.

Related Information

Set by this command: [create_mode](#)

Related attribute: (library_domain) [library](#) on page 1427

Instance Attributes

Contain information about a combinational, sequential, or hierarchical instance in the specified design. Instance objects are found in the `instances_comb`, `instances_seq`, and `instances_hier` directories.

- To set an instance attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -instance name]
```

- To get an instance attribute value, type

```
get_attribute attribute_name [find /des*/design -instance name]
```

cell_delay_multiplier

`cell_delay_multiplier float`

Default: 1.0

Read-write `instance` attribute. Scales the delay of the instance by the specified value. If both the libcell and instance attribute values differ from 1.0, the value of the instance attribute will be used to scale the delay. If the value on the instance is the default value, but the value on the libcell was changed, then the value on the libcell is used to scale the delay.

Note: You can only set this attribute on leaf-level instances.

This attribute can also be set by the following SDC command:

```
dc::set_timing_derate -late -data -cell_delay instance
```

Related Information

Affects these commands: [report cell_delay_calculation](#)

[report slew_calculation](#)

[report timing](#)

Related attributes: [\(design\) cell_delay_multiplier](#) on page 545

[\(libcell\) cell_delay_multiplier](#) on page 191

cell_min_delay_multiplier

`cell_min_delay_multiplier float`

Default: 1.0

Read-write instance attribute. Scales the minimum delay of the instance by the specified value. This scaled delay is used to compute the input delay to the from_pin during data-to-data checking. If both the libcell and instance attribute values differ from 1.0, the value of the instance attribute will be used to scale the delay. If the value on the instance is the default value, but the value on the libcell was changed, then the value on the libcell is used to scale the delay.

Note: You can only set this attribute on leaf-level instances.

This attribute can also be set by the following SDC command:

`dc::set_timing_derate -early -data -cell_delay instance`

Related Information

Setting Data-to-Data Checks in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affects these commands:

[report cell_delay_calculation](#)

[report slew_calculation](#)

[report timing](#)

Related attributes:

(design) [cell_min_delay_multiplier](#) on page 546

(libcell) [cell_min_delay_multiplier](#) on page 193

disabled_arcs

`disabled_arcs Tcl_list`

Read-write [instance](#) attribute. Disables or breaks timing arcs (path delay) of library cells for synthesis and static timing analysis. This attribute affects only the specified instance and not all references of the library cell in the top level of the design. A disable on the specific arc requests the static timing analysis engine not to consider the path under consideration for timing analysis. You can specify a Tcl list of timing arcs.

Note: Only arcs of mapped instances can be disabled.

Example

The following example disables all the timing arcs from pin A1 to Z in instance p0553A which is an AND2D1 cell in the design after mapping to the technology library.

```
set_attribute disabled_arcs [find ./lib*/*/libcells/AND2D1 -libarc A1_n90]  
[find ./ -instance p0553A]
```

Related Information

[Disabling Timing Arcs for a Specified Instance in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Related command: [report timing](#)

Affected by this attribute: [\(instance\) disabled_arcs_by_mode](#) on page 565

Related attributes: [\(instance\) timing_case_disabled_arcs_by_mode](#) on page 954

disabled_arcs_by_mode

```
disabled_arcs_by_mode {mode_name_1 libarcs} [{mode libarcs}]...
```

Read-write instance attribute. Specifies the libarc objects to be disabled in each timing mode. Specify a Tcl list of modes and a corresponding list of libarcs.

When the `disabled_arcs_by_mode` attribute is set, the `disabled_arcs` attribute will be reset to " ". However, when all the values for the `disabled_arcs_by_mode` attribute are the same, the `disabled_arcs` attribute will be set to that value.

When the `disabled_arcs` attribute is set, the `disabled_arcs_by_mode` attribute will reflect that value for each mode.

Example

- The following example shows that this attribute displays a list of lists, which gets overwritten each time it is set:

```
rc:/> get_attribute disabled_arcs_by_mode nand_inst
{/designs/top/modes/a /libraries/typical/libcells/NAND2X1/Y/inarcs/B_Y_n60}
rc:/> set_attribute disabled_arcs_by_mode [list [list b \
[find NAND2x1/Y -libarc B_Y*]]] nand_inst
rc:/> get_attribute disabled_arcs_by_mode nand_inst
{/designs/top/modes/b /libraries/typical/libcells/NAND2X1/Y/inarcs/B_Y_n60}
```

- The following example shows how to set this attribute in different modes:

```
rc:/> set_attribute disabled_arcs_by_mode [list [list b \
[find NAND2X1/Y -libarc B_Y*]] [list a \
[find NAND2X1/Y -libarc B_Y*]]] nand_inst
rc:/> get_attribute disabled_arcs_by_mode nand_inst
{/designs/top/modes/b /libraries/typical/libcells/NAND2X1/Y/inarcs/B_Y_n60} \
{/designs/top/modes/a /libraries/typical/libcells/NAND2X1/Y/inarcs/B_Y_n60}
rc:/> get_attribute disabled_arcs_by_mode nand_inst
/libraries/typical/libcells/NAND2X1/Y/inarcs/B_Y_n60
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands:

[report clocks](#)

[report qor](#)

[report timing](#)

Attribute Reference for Encounter RTL Compiler

Constraint—Instance Attributes

	write encounter
	write sdc
	write script
Related commands:	create mode
	read sdc
Affects this attribute:	disabled arcs on page 564
Related attributes:	(pin) external delays by mode on page 969 (port) external delays by mode on page 1007 (design) latch borrow by mode on page 552 (instance) latch borrow by mode on page 569 (design) latch max borrow by mode on page 555 (instance) latch max borrow by mode on page 572 (pin) latch max borrow by mode on page 602 (pin) propagated clocks by mode on page 976 (port) propagated clocks by mode on page 1015 (design) slack by mode on page 943 (pin) slack by mode on page 980 (port) slack by mode on page 1018 (cost group) slack by mode on page 1060 (pin) timing case computed value by mode on page 985 (port) timing case computed value by mode on page 1021 instance) timing case disabled arcs on page 953 (instance) timing case disabled arcs by mode on page 954 (pin) timing case logic value on page 612 (port) timing case logic value by mode on page 664

hard_region

```
hard_region {false | true}
```

Default: false

Read-write instance attribute. Specifies hierarchical instances that are recognized as hard regions in your floorplan during logic synthesis and preserves pins and subports.

Note: Some place and route tools operate better if your design has no buffers between regions at the top level. To accommodate this, specify hard regions before mapping.

Example

The following example specifies the hard region on the pbu_ctl instance:

```
rc:/> set_attribute hard_region 1 [find / -instance pbu_ctl]
```

Related Information

[Creating Hard Regions in Using Encounter RTL Compiler](#)

Related attribute: (subdesign) hard_region on page 667

latch_borrow

`latch_borrow {no_value | float}`

Default: no_value

Read-write [instance](#) attribute. Specifies the time (in ps) borrowed from the next clock cycle. The resolution is 1.

Note: A `latch_borrow` definition is only valid when inferencing a latch or a blackbox.

Example

The following example sets time borrowing to 300 picoseconds for all latches inferred in the design top:

```
rc:/> set_attr latch_borrow 300 [find ./ -design test_top]
```

Related Information

[Using Latch Time Borrowing in the Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affected by this attribute: (instance) [latch_borrow_by_mode](#) on page 569

Related attribute: (design) [latch_borrow](#) on page 551

latch_borrow_by_mode

```
latch_borrow_by_mode {mode_name_1 delay_value} [{mode_name_2 delay_value}]...
```

Read-write instance attribute. Specifies the time borrowed from the next clock cycle for latches in the netlist for all the timing modes. Valid only for latches and timing models. The value must be non-negative. Specify a Tcl list of modes and corresponding delays.

When the `latch_borrow_by_mode` attribute is set, the `latch_borrow` attribute will be reset to `no_value`. However, when all the values for the `latch_borrow_by_mode` attribute are the same, the `latch_borrow` attribute is set to that value.

When the `latch_borrow` attribute is set, the `latch_borrow_by_mode` attribute reflects that value for each mode.

Note: If this attribute is set on a design and on an instance, the smaller time borrow value will have priority.

Examples

- The following example sets this attribute in modes a and b:

```
rc:/> set_attribute latch_borrow_by_mode {{a 20} {b 5}} i0
rc:/> get_attribute latch_borrow_by_mode i0
{/designs/latch/modes/b 5.0} {/designs/latch/modes/a 20.0}
```

- The following example shows that the value of this attribute is a list of lists that gets overwritten each time it is set:

```
rc:/> get_attribute latch_borrow_by_mode i0
{/designs/latch/modes/b 20.0}
rc:/> set_attribute latch_borrow_by_mode {{a 20}} i0
rc:/> get_attribute latch_borrow_by_mode i0
{/designs/latch/modes/a 20.0}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands: [report clocks](#)
 [report qor](#)
 [report timing](#)
 [write encounter](#)

Attribute Reference for Encounter RTL Compiler

Constraint—Instance Attributes

[write_sdc](#)

[write_script](#)

Related commands: [create_mode](#)

[read_sdc](#)

Affects this attribute [\(instance\) latch_borrow](#) on page 568

Related attributes: [disabled_arcs_by_mode](#) on page 565

(pin) [external_delays_by_mode](#) on page 969

(port) [external_delays_by_mode](#) on page 1007

(design) [latch_borrow](#) on page 551

(instance) [latch_borrow](#) on page 568

(design) [latch_borrow_by_mode](#) on page 552

(clock) [latch_max_borrow](#) on page 677

(instance) [latch_max_borrow](#) on page 571

(pin) [latch_max_borrow](#) on page 601

(design) [latch_max_borrow_by_mode](#) on page 555

(instance) [latch_max_borrow_by_mode](#) on page 572

(pin) [latch_max_borrow_by_mode](#) on page 602

(pin) [propagated_clocks_by_mode](#) on page 976

(port) [propagated_clocks_by_mode](#) on page 1015

(design) [slack_by_mode](#) on page 943

(pin) [slack_by_mode](#) on page 980

(port) [slack_by_mode](#) on page 1018

(cost group) [slack_by_mode](#) on page 1060

(pin) [timing_case_computed_value_by_mode](#) on page 985

(port) [timing_case_computed_value_by_mode](#) on page 1021

instance) [timing_case_disabled_arcs](#) on page 953

(instance) [timing_case_disabled_arcs_by_mode](#) on page 954

(pin) [timing_case_logic_value](#) on page 612

(port) [timing_case_logic_value_by_mode](#) on page 664

latch_max_borrow

latch_max_borrow {no_value | *integer*}

Default: no_value

Read-write [instance](#) attribute. Specifies the maximum amount of time that can be borrowed, in picoseconds, from the next clock cycle. The specified value must be an integer greater or equal to 0 (decimal values are rounded to the nearest integer) or no_value. This attribute is available on latch cells, clocks, or on a design. If it is set on a latch cell, it replaces the value set on the design. If the [latch_borrow](#) attribute has already been set, then the [latch_max_borrow](#) attribute is ignored.

Example

```
rc:/> set_attribute latch_max_borrow no_value \
    [find /des*/design -instance name]
```

Related Information

Affected by these attributes:	(design) latch_borrow on page 551 (instance) latch_borrow on page 568 (instance) latch_max_borrow_by_mode on page 602
Related attributes:	(clock) latch_max_borrow on page 677 (design) latch_max_borrow on page 554 (pin) latch_max_borrow on page 601 (design) latch_max_borrow_by_mode on page 555 (pin) latch_max_borrow_by_mode on page 602

latch_max_borrow_by_mode

```
latch_max_borrow_by_mode {mode_name_1 delay_value}  
[ {mode_name_2 delay_value}]...
```

Read-write instance attribute. Specifies the maximum time borrowed, for an instance, from the next clock cycle for latches in a netlist for all timing modes. Valid only for latches and timing models. The value must be non-negative. Specify a Tcl list of modes and corresponding delays.

When the `latch_max_borrow_by_mode` attribute is set, the `latch_max_borrow` attribute will be reset to `no_value`. However, when all the values for the `latch_max_borrow_by_mode` attribute are the same, the `latch_max_borrow` attribute is set to that value.

When the `latch_max_borrow` attribute is set, the `latch_max_borrow_by_mode` attribute reflects that value for each mode.

Example

- The following example shows that the value of this attribute is a list of lists that gets overwritten each time it is set:

```
rc:/> get_attribute latch_max_borrow_by_mode io  
{/designs/latch/modes/a 20000.0}  
rc:/> set_attribute latch_max_borrow_by_mode [list [list b 10000]] io  
{/designs/latch/modes/b 10000.0}
```

- The following example sets this attribute in modes a and b:

```
rc:/> set_attribute latch_max_borrow_by_mode [list [list b 10000] \  
[list a 20000]] io  
rc:/> get_attribute latch_max_borrow_by_mode io  
{/designs/latch/modes/b 10000.0} {/designs/latch/modes/a 20000.0}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:

[report clocks](#)

[report qor](#)

[report timing](#)

[write encounter](#)

Attribute Reference for Encounter RTL Compiler

Constraint—Instance Attributes

	write_sdc
	write_script
Related commands:	create_mode
	read_sdc
Affects this attributes	instance) latch_max_borrow on page 571
Related attributes:	disabled_arcs_by_mode on page 565 (pin) external_delays_by_mode on page 969 (port) external_delays_by_mode on page 1007 (design) latch_borrow on page 551 (instance) latch_borrow on page 568 (design) latch_borrow_by_mode on page 552 (clock) latch_max_borrow on page 677 (pin) latch_max_borrow on page 601 (design) latch_max_borrow_by_mode on page 555 (pin) latch_max_borrow_by_mode on page 602 (pin) propagated_clocks_by_mode on page 976 (port) propagated_clocks_by_mode on page 1015 (design) slack_by_mode on page 943 (pin) slack_by_mode on page 980 (port) slack_by_mode on page 1018 (cost group) slack_by_mode on page 1060 (pin) timing_case_computed_value_by_mode on page 985 (port) timing_case_computed_value_by_mode on page 1021 instance) timing_case_disabled_arcs on page 953 (instance) timing_case_disabled_arcs_by_mode on page 954 (pin) timing_case_logic_value on page 612 (port) timing_case_logic_value_by_mode on page 664

Pin Attributes

Contain information about the specified instance pin. A pin is a single bit connection point on an instance. Pin objects are stored in `pins_in` and `pins_out` directories.

- To set a pin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pin instance/pin]
```

- To get a pin attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pin instance/pin]
```

Note: You may need to specify more than one instance to serve as the search path that will uniquely identify the relevant pin.

break_timing_paths

```
break_timing_paths {false | true | propagate_slews}
```

Default: false

Read-write `pin` attribute. Controls whether to break the timing path at the specified pin. By default the timing path is not broken.

If the attribute is set to `true`, the timing path is broken and external delays can be set on the specified pin and the pin can function as `-to`, `-through`, or `-from` point in path exception commands.

If the attribute is set to `false`, the timing path is not broken and the slew values will propagate through the specified pin.

If the attribute is set to `propagate_slews`, the slew will propagate from the driver. The load on the net is unaffected by the `break_timing_paths` attribute.

Example

```
rc:/> set_attribute break_timing_path true \
[find /des*/design -pin instance/pin]
```

Attribute Reference for Encounter RTL Compiler

Constraint—Pin Attributes

Related Information

[Breaking the Path of a Specified Pin in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affects these commands:

[report clocks](#)

[report qor](#)

[write_sdc](#)

[report timing](#)

[write_sdc](#)

[write_script](#)

Related commands:

[multi_cycle](#)

[path_adjust](#)

[path_delay](#)

[path_disable](#)

[specify_paths](#)

Related attributes:

(port) [break_timing_paths](#) on page 616

(pin) [break_timing_paths_by_mode](#) on page 576

(port) [break_timing_paths_by_mode](#) on page 617

[external_driver](#) on page 641

[fixed_slew](#) on page 647

[slew](#) on page 678

break_timing_paths_by_mode

```
break_timing_paths_by_mode {mode_1 {0 | 1 | propagate_slews}  
    {mode_2 {0 | 1 | propagate_slews}} ...}
```

Read-write attribute. Controls whether to break the timing path at the pin for particular timing modes. Setting the value to 1 for a mode will break the timing path in that mode.

Note: The `break_timing_paths` attribute must be set to `false` for the `break_timing_paths_by_mode` attribute to take effect.



If you set the value to `propagate_slews` or `false` in one or more modes, but not in all modes, the slews will be propagated in all modes.

Only if you set the value to `true` or `1` in all modes will the slew not be propagated. The timing paths will be broken.

Related Information

Affects these commands: [report clocks](#)

[report qor](#)

[write_sdc](#)

[report timing](#)

[write_sdc](#)

[write_script](#)

Related commands: [multi_cycle](#)

[path_adjust](#)

[path_delay](#)

[path_disable](#)

[specify_paths](#)

Related attributes: (pin) [break_timing_paths](#) on page 574

(port) [break_timing_paths](#) on page 616

(port) [break_timing_paths_by_mode](#) on page 617

(instance) [disabled_arcs](#) on page 564

(instance) [disabled_arcs_by_mode](#) on page 565

Attribute Reference for Encounter RTL Compiler

Constraint—Pin Attributes

(pin) [timing_case_computed_value_by_mode](#) on page 985

instance) [timing_case_disabled_arcs](#) on page 953

(instance) [timing_case_disabled_arcs_by_mode](#) on page 954

(pin) [timing_case_logic_value](#) on page 612

(port) [timing_case_logic_value](#) on page 663

(pin) [timing_case_logic_value_by_mode](#) on page 613

(port) [timing_case_logic_value_by_mode](#) on page 664

clock_hold_uncertainty

```
clock_hold_uncertainty { {no_value no_value} | Tcl_list }
```

Default: no_value no_value

Read-write [pin](#) attribute. Specifies the uncertainty in the arrival times of capturing edges (in picoseconds) for the clock in early-mode (hold) timing analysis. RTL Compiler ignores (does not use) this value in optimization and timing analysis, but can pass it to downstream tools.

If you specify a single value, then both the rising and falling edge are set to that single value. If you specify a Tcl list containing two values, then the two values are interpreted as the rising and falling edge.

Example

The following commands set the rising and falling clock edge to 50 picoseconds on pin c.

```
rc:/> set_attribute clock_hold_uncertainty 50 a/b/c
rc:/> set_attribute clock_hold_uncertainty {50 50} a/b/c
```

Related Information

[Specifying Clock Skew in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:

- (clock) [clock_hold_uncertainty](#) on page 668
- (port) [clock_hold_uncertainty](#) on page 619
- (pin) [clock_hold_uncertainty_by_mode](#) on page 579
- (port) [clock_hold_uncertainty_by_mode](#) on page 620
- (clock) [clock_setup_uncertainty](#) on page 672
- (pin) [clock_setup_uncertainty](#) on page 588
- (port) [clock_setup_uncertainty](#) on page 629

clock_hold_uncertainty_by_mode

`clock_hold_uncertainty_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }`

Default: {no_value no_value} (for each mode)

Read-write [pin](#) attribute. Specifies the uncertainty in the arrival times of capturing edges (in picoseconds) for the clock in early-mode (hold) timing analysis for particular timing modes. RTL Compiler ignores (does not use) these values in optimization and timing analysis, but can pass them to downstream tools.

If you specify a single value, then both the rising and falling edge are set to that single value. If you specify a Tcl list containing two values, then the two values are interpreted as the rising and falling edge.

Note: When you set this attribute, it overrides the `clock_hold_uncertainty` attribute on this pin.

Example

The following command set the rising and falling clock edges for modes A and B on pin CK.

```
rc:/> set_attr clock_hold_uncertainty_by_mode {{ A {1 2}} {B 3}} [find / -pin CK]
Setting attribute of pin 'CK': 'clock_hold_uncertainty_by_mode' =
{/designs/top/modes/A {1.0 2.0}} {/designs/top/modes/B {3.0 3.0}}
```

Related Information

[Specifying Clock Skew in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (pin) [hold uncertainty by clock](#) on page 598

Affects this attribute: (pin) [clock hold uncertainty](#) on page 578

Related attributes: (clock) [clock hold uncertainty](#) on page 668

(port) [clock hold uncertainty](#) on page 619

(port) [clock hold uncertainty by mode](#) on page 620

(clock) [clock setup uncertainty](#) on page 672

(pin) [clock setup uncertainty](#) on page 588

(port) [clock setup uncertainty](#) on page 629

(pin) [clock setup uncertainty by mode](#) on page 589

(port) [clock setup uncertainty by mode](#) on page 631

clock_network_early_latency

`clock_network_early_latency { {no_value no_value no_value no_value} | Tcl_list }`

Default: no_value no_value no_value no_value

Read-write [pin](#) attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for early-mode (hold checking) timing analysis. You can specify a Tcl list of up to four delay values.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and an actual sequential element.

The values set on the pin both override the latency values for all clock signals that propagate through the pin and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: RTL Compiler ignores this value in optimization and timing analysis, but can pass it to downstream tools.

Example

The following commands sets the minimum and maximum rise latencies to 150, and the minim and maximum fall latencies to 30.

```
rc:/> set_attribute clock_network_early_latency {150 30} a/b/c
rc:/> set_attribute clock_network_early_latency {150 30 150 30} a/b/c
```

Attribute Reference for Encounter RTL Compiler

Constraint—Pin Attributes

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

- Related attributes:
- (clock) [clock network early latency](#) on page 669
 - (port) [clock network early latency](#) on page 621
 - (pin) [clock network early latency by mode](#) on page 582
 - (port) [clock network early latency by mode](#) on page 623
 - (clock) [clock network late latency](#) on page 670
 - (pin) [clock network late latency](#) on page 584
 - (port) [clock network late latency](#) on page 625
 - (clock) [clock source early latency](#) on page 673
 - (pin) [clock source early latency](#) on page 590
 - (port) [clock source early latency](#) on page 632
 - (clock) [clock source late latency](#) on page 675
 - (pin) [clock source late latency](#) on page 594
 - (port) [clock source late latency](#) on page 636

clock_network_early_latency_by_mode

```
clock_network_early_latency_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }
```

Default: {no_value no_value no_value no_value} (for each mode)

Read-write [pin](#) attribute. Specifies the delay at the pin between the ideal waveform edges and the sequential elements in the circuit for early-mode (hold checking) timing analysis. For each timing mode, you can specify a Tcl list of up to four delay values.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and an actual sequential element.

The values set on the pin both override the latency values for all clock signals that propagate through the pin and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: RTL Compiler ignores this value in optimization and timing analysis, but can pass it to downstream tools.

Note: When you set this attribute, it overrides the `clock_network_early_latency` attribute set on the pin.

Example

The following command sets the network early latencies for modes A and B on pin CK.

```
rc:/> set_attr clock_network_early_latency_by_mode {{ A {150 30}} \
==> {B {75 20 60 15}}} flop/CK
Setting attribute of pin 'CK': 'clock_network_early_latency_by_mode' =
{/designs/top/modes/A {150.0 30.0 150.0 30.0}}{/designs/top/modes/B {75.0 20.0
60.0 150.0}}
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (pin) [network early latency by clock](#) on page 604

Affects this attribute: (pin) [clock network early latency](#) on page 580

Related attributes: (clock) [clock network early latency](#) on page 669

(port) [clock network early latency](#) on page 621

(port) [clock network early latency by mode](#) on page 623

(clock) [clock network late latency](#) on page 670

(pin) [clock network late latency](#) on page 584

(port) [clock network late latency](#) on page 625

(pin) [clock network late latency by mode](#) on page 586

(port) [clock network late latency by mode](#) on page 627

(clock) [clock source early latency](#) on page 673

(pin) [clock source early latency](#) on page 590

(port) [clock source early latency](#) on page 632

(pin) [clock source early latency by mode](#) on page 592

(port) [clock source early latency by mode](#) on page 634

(clock) [clock source late latency](#) on page 675

(pin) [clock source late latency](#) on page 594

(port) [clock source late latency](#) on page 636

(pin) [clock source late latency by mode](#) on page 596

(port) [clock source late latency by mode](#) on page 638

clock_network_late_latency

```
clock_network_late_latency { {no_value no_value no_value no_value} | Tcl_list }
```

Default: no_value no_value no_value no_value

Read-write [pin](#) attribute. Specifies the delay (in picoseconds) between the ideal waveform edges and the sequential elements in the circuit for late-mode (setup checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the pin both override the latency values for all clock signals that propagate through the pin and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

In addition to numeric delay values, this attribute accepts the special string `no_value` to indicate that no change should be made to the latency. The default value of the attribute is `{no_value no_value no_value no_value}`, which indicates that all four latency values should remain unchanged.

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Examples

- Both of the following examples have the same effect:

```
rc:/> set_attribute clock_network_late_latency {150 30} a/b/c
rc:/> set_attribute clock_network_late_latency {150 30 150 30} a/b/c
```
- The following example sets the network fall latency to 50 picoseconds but leaves the network rise latency unchanged:

```
rc:/> set_attribute clock_network_late_latency {no_value 50} a/b/c
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:	(clock) <u>clock_network_early_latency</u> on page 669 (pin) <u>clock_network_early_latency</u> on page 580 (port) <u>clock_network_early_latency</u> on page 621 (clock) <u>clock_network_late_latency</u> on page 670 (port) <u>clock_network_late_latency</u> on page 625 (pin) <u>clock_network_late_latency_by_mode</u> on page 586 (port) <u>clock_network_late_latency_by_mode</u> on page 627 (clock) <u>clock_source_early_latency</u> on page 673 (pin) <u>clock_source_early_latency</u> on page 590 (port) <u>clock_source_early_latency</u> on page 632 (clock) <u>clock_source_late_latency</u> on page 675 (pin) <u>clock_source_late_latency</u> on page 594 (port) <u>clock_source_late_latency</u> on page 636
---------------------	--

clock_network_late_latency_by_mode

`clock_network_late_latency_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }`

Default: {no_value no_value no_value no_value} (for each mode)

Read-write [pin](#) attribute. Specifies the delay at the pin (in picoseconds) between the ideal waveform edges and the sequential elements in the circuit for late-mode (setup checking) timing analysis. For each timing mode, you can specify a Tcl list of up to four delay values.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the pin both override the latency values for all clock signals that propagate through the pin and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

In addition to numeric delay values, this attribute accepts the special string `no_value` to indicate that no change should be made to the latency. The default value of the attribute is `{no_value no_value no_value no_value}`, which indicates that all four latency values should remain unchanged.

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: When you set this attribute, it overrides the `clock_network_late_latency` attribute set on this pin.

Example

The following command sets the network late latencies for modes A and B on pin CK.

```
rc:/> set_attr clock_network_late_latency_by_mode {{ A {150 30}} \n==> {B {75 20 60 15}}} flop/CK\nSetting attribute of pin 'CK': 'clock_network_late_latency_by_mode' =\n{/designs/top/modes/A {150.0 30.0 150.0 30.0}}{/designs/top/modes/B {75.0 20.0\n60.0 15.0}}
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (pin) [network late latency by clock](#) on page 655

Affects this attribute: (pin) [clock network late latency](#) on page 584

Related attributes: (clock) [clock network early latency](#) on page 669

(pin) [clock network early latency](#) on page 580

(port) [clock network early latency](#) on page 621

(pin) [clock network early latency by mode](#) on page 582

(port) [clock network early latency by mode](#) on page 623

(clock) [clock network late latency](#) on page 670

(port) [clock network late latency](#) on page 625

(port) [clock network late latency by mode](#) on page 627

(clock) [clock source early latency](#) on page 673

(pin) [clock source early latency](#) on page 590

(port) [clock source early latency](#) on page 632

(pin) [clock source early latency by mode](#) on page 592

(port) [clock source early latency by mode](#) on page 634

(clock) [clock source late latency](#) on page 675

(pin) [clock source late latency](#) on page 594

(port) [clock source late latency](#) on page 636

(pin) [clock source late latency by mode](#) on page 596

(port) [clock source late latency by mode](#) on page 638

clock_setup_uncertainty

```
clock_setup_uncertainty { {no_value no_value} | Tcl_list }
```

Default: no_value no_value

Read-write pin attribute. Specifies the uncertainty in the arrival times of capturing edges for clocks in late-mode timing analysis. The values set on a pin both override the uncertainty values for all clock signals that propagate through the pin and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list specifying the uncertainty in the rising and falling edges of the ideal clock waveform. Positive values indicate that the clock may arrive earlier in time, which causes timing constraints to be tighter.

In addition to numeric delay values, this attribute accepts the special string no_value to indicate that no change should be made to the uncertainty times. The default value of this attribute is {no_value no_value}, indicating that both rise and fall uncertainties should be left unchanged.

If you specify a single value (instead of a Tcl list representing the rise and fall values), both rise and fall values are set to the single value.

Examples

- Both of the following commands set the rise and fall setup uncertainty arrival times for pin a/b/c to 100 picoseconds:

```
rc:/> set_attribute clock_setup_uncertainty {100 100} a/b/c  
rc:/> set_attribute clock_setup_uncertainty 100 a/b/c
```

- The following command sets the rise setup uncertainty to 100 picoseconds and the fall setup uncertainty to 50 picoseconds:

```
rc:/> set_attribute clock_setup_uncertainty {100 50} a/b/c
```

- The following command sets the rise setup uncertainty to 50 picoseconds but leaves the fall uncertainty unchanged:

```
rc:/> set_attribute clock_setup_uncertainty {50 no_value} a/b/c
```

Related Information

[Specifying Clock Skew in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:	(clock) <u>clock_hold_uncertainty</u> on page 668 (pin) <u>clock_hold_uncertainty</u> on page 578 (port) <u>clock_hold_uncertainty</u> on page 619 (clock) <u>clock_setup_uncertainty</u> on page 672 (port) <u>clock_setup_uncertainty</u> on page 629 (pin) <u>clock_setup_uncertainty_by_mode</u> on page 589 (port) <u>clock_setup_uncertainty_by_mode</u> on page 631
---------------------	--

clock_setup_uncertainty_by_mode

`clock_setup_uncertainty_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }`

Default: {no_value no_value} (for each mode)

Read-write [pin](#) attribute. Specifies the uncertainty in the arrival times of capturing edges for clocks in late-mode timing analysis. The values set on a pin both override the uncertainty values for all clock signals that propagate through the pin and apply to all sequential elements in the transitive fanout. For each timing mode, you can specify a Tcl list of up to two values.

The attribute value is a Tcl list specifying the uncertainty in the rising and falling edges of the ideal clock waveform. Positive values indicate that the clock may arrive earlier in time, which causes timing constraints to be tighter.

In addition to numeric delay values, this attribute accepts the special string no_value to indicate that no change should be made to the uncertainty times. The default value of this attribute is {no_value no_value}, indicating that both rise and fall uncertainties should be left unchanged.

If you specify a single value (instead of a Tcl list representing the rise and fall values), both rise and fall values are set to the single value.

Note: When you set this attribute, it overrides the `clock_setup_uncertainty` attribute set on this pin.

Examples

- The following command sets the setup uncertainties for modes A and B:

```
rc:/> set_attr clock_setup_uncertainty_by_mode {{ A {100 50}} {B 50}} \
==> [find / -pin CK]
      Setting attribute of pin 'CK': 'clock_setup_uncertainty_by_mode' =
{/designs/top/modes/A {100.0 50.0}} {/designs/top/modes/B {50.0 50.0}}
```

Related Information

[Specifying Clock Skew in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (pin) [setup uncertainty by clock](#) on page 607

Affects this attribute: (pin) [clock setup uncertainty](#) on page 588

Related attributes: (clock) [clock hold uncertainty](#) on page 668

(pin) [clock hold uncertainty](#) on page 578

(port) [clock hold uncertainty](#) on page 619

(pin) [clock hold uncertainty by mode](#) on page 579

(port) [clock hold uncertainty by mode](#) on page 620

(clock) [clock setup uncertainty](#) on page 672

(port) [clock setup uncertainty](#) on page 629

(port) [clock setup uncertainty by mode](#) on page 631

clock_source_early_latency

```
clock_source_early_latency { {no_value no_value no_value no_value} | Tcl_list}
```

Default: no_value no_value no_value no_value

Read-write [pin](#) attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for early-mode (hold checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the pin both override the latency values for all clock signals that propagate through the pin and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

The RTL Compiler tool does not use these attribute values because the tool does not perform any hold timing analysis. However, when you execute the `write_sdc` command, the values specified for the `clock_source_early_latency` attribute are written out to the SDC constraints file.

Example

```
rc:/> set_attribute clock_source_early_latency 150 a/b/c
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:

- (clock) [clock network early latency](#) on page 669
- (pin) [clock network early latency](#) on page 580
- (port) [clock network early latency](#) on page 621
- (clock) [clock network late latency](#) on page 670
- (pin) [clock network late latency](#) on page 584
- (port) [clock network late latency](#) on page 625
- (clock) [clock source early latency](#) on page 673
- (port) [clock source early latency](#) on page 632
- (pin) [clock source early latency by mode](#) on page 592
- (port) [clock source early latency by mode](#) on page 634

(clock) [clock_source_late_latency](#) on page 675

(pin) [clock_source_late_latency](#) on page 594

(port) [clock_source_late_latency](#) on page 636

clock_source_early_latency_by_mode

`clock_source_early_latency_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }`

Default: {no_value no_value no_value no_value} (for each mode)

Read-write [pin](#) attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for early-mode (hold checking) timing analysis. For each timing mode, you can specify a Tcl list of up to four delay values.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the pin both override the latency values for all clock signals that propagate through the pin and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

The RTL Compiler tool does not use these attribute values because the tool does not perform any hold timing analysis. However, when you execute the `write_sdc` command, the values specified for the `clock_source_early_latency_by_mode` attribute are written out to the SDC constraints file.

Note: When you set this attribute, it overrides the `clock_source_early_latency` attribute set on this pin.

Example

```
rc:/> set_attr clock_source_early_latency_by_mode {{ A {100 50 150 60} } \  
==> {B {110 70}} } [find / -pin CK]  
Setting attribute of pin 'CK': 'clock_source_early_latency_by_mode' =  
{/designs/top/modes/A {100.0 50.0 150.0 60.0}}{/designs/top/modes/B {110.0 70.0  
110.0 70.0}}
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (pin) [source early latency by clock](#) on page 609

Affects this attribute: (pin) [clock source early latency](#) on page 590

Related attributes: (clock) [clock network early latency](#) on page 669

(pin) [clock network early latency](#) on page 580

(port) [clock network early latency by mode](#) on page 621

(pin) [clock network early latency by mode](#) on page 582

(port) [clock network early latency by mode](#) on page 623

(clock) [clock network late latency](#) on page 670

(pin) [clock network late latency](#) on page 584

(port) [clock network late latency](#) on page 625

(pin) [clock network late latency by mode](#) on page 586

(port) [clock network late latency by mode](#) on page 627

(clock) [clock source early latency](#) on page 673

(port) [clock source early latency](#) on page 632

(port) [clock source early latency by mode](#) on page 634

(clock) [clock source late latency](#) on page 675

(pin) [clock source late latency](#) on page 594

(port) [clock source late latency](#) on page 636

(pin) [clock source late latency by mode](#) on page 596

(port) [clock source late latency by mode](#) on page 638

clock_source_late_latency

```
clock_source_late_latency { {no_value no_value no_value no_value} | Tcl_list }
```

Default: no_value no_value no_value no_value

Read-write [pin](#) attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for late-mode (setup checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the pin both override the latency values for all clock signals that propagate through the pin and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

In addition to numeric delay values, this attribute accepts the special string `no_value` to indicate that no change should be made to the latency. The default value of the attribute is `{no_value no_value no_value no_value}`, which indicates that all four latency values should remain unchanged.

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Examples

All of the following examples have the same effect:

```
rc:/> set_attribute clock_source_late_latency 150 a/b/c
rc:/> set_attribute clock_source_late_latency {150 150} a/b/c
rc:/> set_attribute clock_source_late_latency {150 150 150 150} a/b/c
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:

- (clock) [clock network early latency](#) on page 669
- (pin) [clock network early latency](#) on page 580
- (port) [clock network early latency](#) on page 621
- (clock) [clock network late latency](#) on page 670
- (pin) [clock network late latency](#) on page 584
- (port) [clock network late latency](#) on page 625
- (clock) [clock source early latency](#) on page 673
- (pin) [clock source early latency](#) on page 590
- (port) [clock source early latency](#) on page 632
- (clock) [clock source late latency](#) on page 675
- (port) [clock source late latency](#) on page 636
- (pin) [clock source late latency by mode](#) on page 596
- (port) [clock source late latency by mode](#) on page 638

clock_source_late_latency_by_mode

`clock_source_late_latency_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }`

Default: {no_value no_value no_value no_value} (for each mode)

Read-write [pin](#) attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for late-mode (setup checking) timing analysis. For each timing mode, you can specify a Tcl list of up to four delay values.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the pin both override the latency values for all clock signals that propagate through the pin and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

In addition to numeric delay values, this attribute accepts the special string `no_value` to indicate that no change should be made to the latency. The default value of the attribute is `{no_value no_value no_value no_value}`, which indicates that all four latency values should remain unchanged.

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: When you set this attribute, it overrides the `clock_source_late_latency` attribute on this pin.

Examples

```
rc:/> set_attr clock_source_late_latency_by_mode {{ A {100 50 150 60} } \ 
==> {B {110 70}} } [find / -pin CK]
Setting attribute of pin 'CK': 'clock_source_late_latency_by_mode' =
{/designs/top/modes/A {100.0 50.0 150.0 60.0} }{/designs/top/modes/B {110.0 70.0
110.0 70.0}}
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (pin) [source_late_latency_by_clock](#) on page 610

Affects this attribute: (pin) [clock_source_late_latency](#) on page 594

Related attributes: (clock) [clock_network_early_latency](#) on page 669

(pin) [clock_network_early_latency](#) on page 580

(port) [clock_network_early_latency_by_mode](#) on page 621

(pin) [clock_network_early_latency_by_mode](#) on page 582

(port) [clock_network_early_latency_by_mode](#) on page 623

(clock) [clock_network_late_latency](#) on page 670

(pin) [clock_network_late_latency](#) on page 584

(port) [clock_network_late_latency](#) on page 625

(pin) [clock_network_late_latency_by_mode](#) on page 586

(port) [clock_network_late_latency_by_mode](#) on page 627

(clock) [clock_source_early_latency](#) on page 673

(pin) [clock_source_early_latency](#) on page 590

(port) [clock_source_early_latency](#) on page 632

(pin) [clock_source_early_latency_by_mode](#) on page 592

(port) [clock_source_early_latency_by_mode](#) on page 634

(clock) [clock_source_late_latency](#) on page 675

(port) [clock_source_late_latency](#) on page 636

(port) [clock_source_late_latency_by_mode](#) on page 638

hold_uncertainty_by_clock

```
hold_uncertainty_by_clock {{clock {rise fall}}...}
```

Read-write [pin](#) attribute. Specifies the uncertainty in the arrival times of the capturing edges of the specified clocks in early-mode timing analysis.

The attribute value is a Tcl list of Tcl lists for a pin. There can be as many Tcl lists as there are clocks propagating to this pin. The Tcl list for a clock contains

- The name of the clock
- The rise and fall uncertainties, respectively

If a single value is specified for a clock, then the rise and fall uncertainties are set to that single value.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this pin, the value of the [clock_hold_uncertainty](#) pin attribute applies.

The RTL Compiler tool does not use these attribute values because the tool does not perform any hold timing analysis. However, when you execute the [write_sdc](#) command, the values specified for the [hold_uncertainty_by_clock](#) attribute are written out to the SDC constraints file.

- If only one uncertainty value is specified for a clock, the following SDC command is written:

```
set_clock_uncertainty -clock clock uncertainty -hold pin
```

- If rise and fall uncertainties were specified, the following SDC commands are written:

```
set_clock_uncertainty -clock clock -rise uncertainty -hold pin
set_clock_uncertainty -clock clock -fall uncertainty -hold pin
```

Example

The following command specifies the rise and fall uncertainty numbers for clock abc at pin clk to be 17 and 27 ps.

```
set_attr hold_uncertainty_by_clock { {abc {17 27}} } [find /des*/top -pin clk]
```

Related Information

Related attributes:

(clock) [clock_hold_uncertainty](#) on page 668

(pin) [clock_hold_uncertainty](#) on page 578

(port) [clock_hold_uncertainty](#) on page 619

(port) [hold_uncertainty_by_clock](#) on page 648

ideal_driver

`ideal_driver {false | true}`

Default: false

Read-write pin attribute. Indicates if the pin is to be considered an ideal driver.

If the pin is an ideal driver, the timer and optimizer will not attempt to optimize any net driven by this pin. Therefore, transitions, connect delay, and design rule constraints of this pin are ignored. This attribute is available on both mapped and unmapped netlists.

Example

```
rc:/> set_attribute ideal_driver true [find /des*/design -pin instance/pin]
```

Related Information

[Handling Ideal Nets in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: [ideal_seq_async_pins](#) on page 548

Related attribute: (port) [ideal_driver](#) on page 649

ideal_network

```
ideal_network {false | true}
```

Default: false

Read-write [pin](#) attribute. Sets the network of the specified driver pin to an ideal network. The pin must be a driving pin. This attribute propagates through combinational gates, and hierarchical boundaries, unlike the [ideal_driver](#) attribute.

Propagation Rules

The propagation of the [ideal_network](#) attribute follows these rules:

- A pin is treated as ideal if it is either a:
 - Pin specified in the object list of the [ideal_network](#) attribute.
 - Driver pin and its cell is ideal.
 - Load pin attached to an ideal net.
- A net is treated as ideal if all its driving cells are ideal.
- A combinational cell is treated as ideal if all its input pins are ideal.

Note: A hierarchical pin can propagate the [ideal_network](#) attribute.

Propagation stops at the pins where these conditions are not met. These pins are referred to as network boundary pins, and they are ideal pins.

Example

The following example sets the network to which the `foo` pin is connected, to an ideal network:

```
rc:/> set_attribute ideal_network true \
{/designs/example/instances_comb/inst2/pins_out/foo}
```

Related Information

Affects this attribute: [propagated_ideal_network](#) on page 978

Related attributes: [\(pin\) ideal_driver](#) on page 599

[\(port\) ideal_driver](#) on page 649

[\(port\) ideal_network](#) on page 649

latch_max_borrow

```
latch_max_borrow {no_value | integer}
```

Default: no_value

Read-write [pin](#) attribute. Specifies the maximum amount of time that can be borrowed, in picoseconds, from the next clock cycle. The specified value must be an integer greater or equal to 0 (decimal values are rounded to the nearest integer) or no_value. This attribute is available on latch cells, clocks, clock (enable) pin, data pin, or on a design. If the attribute is set on multiple objects that overlap each other, for example a latch instance and its data pin, the minimum value will be taken. If the [latch_borrow](#) attribute has already been set, then the [latch_max_borrow](#) attribute is ignored.

Example

```
rc:/> set_attribute latch_max_borrow 20000.0 \
[find /des*/design -pin instance_A/B]
```

Related Information

[Specifying Latch Time Borrow Values in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by these attributes:	(design) latch_borrow on page 551 (instance) latch_borrow on page 568 (pin) latch_max_borrow_by_mode on page 602
Related attributes:	(clock) latch_max_borrow on page 677 (design) latch_max_borrow on page 554 (pin) latch_max_borrow on page 601 (design) latch_max_borrow_by_mode on page 555 (instance) latch_max_borrow_by_mode on page 572

latch_max_borrow_by_mode

```
latch_max_borrow_by_mode {mode_name_1 delay_value}  
[ {mode_name_2 delay_value}]...
```

Read-write [pin](#) attribute. Specifies the maximum time borrowed from the next clock cycle for latches in a netlist for all timing modes. Valid only for data and enable pins of latches and timing models. The value must be non-negative. Specify a Tcl list of modes and corresponding delays.

When the `latch_max_borrow_by_mode` attribute is set, the `latch_max_borrow` attribute will be reset to `no_value`. However, when all the values for the `latch_max_borrow_by_mode` attribute are the same, the `latch_max_borrow` attribute is set to that value.

When the `latch_max_borrow` attribute is set, the `latch_max_borrow_by_mode` attribute reflects that value for each mode.

Example

- The following example shows that the value of this attribute is a list of lists that gets overwritten each time it is set:

```
rc:/> get_attribute latch_max_borrow_by_mode i0/A  
{/designs/latch/modes/a 20000.0}  
rc:/> set_attribute latch_max_borrow_by_mode [list [list b 10000]] i0/A  
{/designs/latch/modes/b 10000.0}
```

- The following example sets this attribute in modes a and b:

```
rc:/> set_attribute latch_max_borrow_by_mode [list [list b 10000] \  
[list a 20000]] i0/A  
rc:/> get_attribute latch_max_borrow_by_mode i0/A  
{/designs/latch/modes/b 10000.0} {/designs/latch/modes/a 20000.0}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:

[report clocks](#)
[report qor](#)
[report timing](#)
[write encounter](#)

Attribute Reference for Encounter RTL Compiler

Constraint—Pin Attributes

	write_sdc
	write_script
Related commands:	create_mode
	read_sdc
Affects this attributes	instance) latch_max_borrow on page 571
Related attributes:	disabled_arcs_by_mode on page 565 (pin) external_delays_by_mode on page 969 (port) external_delays_by_mode on page 1007 (design) latch_borrow on page 551 (instance) latch_borrow on page 568 (design) latch_borrow_by_mode on page 552 (clock) latch_max_borrow on page 677 (pin) latch_max_borrow on page 601 (design) latch_max_borrow_by_mode on page 555 (instance) latch_max_borrow_by_mode on page 572 (pin) propagated_clocks_by_mode on page 976 (port) propagated_clocks_by_mode on page 1015 (design) slack_by_mode on page 943 (pin) slack_by_mode on page 980 (port) slack_by_mode on page 1018 (cost group) slack_by_mode on page 1060 (pin) timing_case_computed_value_by_mode on page 985 (port) timing_case_computed_value_by_mode on page 1021 instance) timing_case_disabled_arcs on page 953 (instance) timing_case_disabled_arcs_by_mode on page 954 (pin) timing_case_logic_value on page 612 (port) timing_case_logic_value_by_mode on page 664

network_early_latency_by_clock

```
network_early_latency_by_clock {{clock {min_rise min_fall max_rise  
max_fall}}...}
```

Read-write [pin](#) attribute. Specifies the delay between the ideal waveform edges of the specified clocks and the sequential elements in the circuit for early-mode (hold checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The attribute value is a Tcl list of Tcl lists for a pin. There can be as many Tcl lists as there are clocks propagating to this pin. The Tcl list for a clock contains

- The name of the clock
- The minimum (capture) latency of the rise edge of the ideal waveform of the clock
- The minimum (capture) latency of the fall edge of the ideal waveform of the clock
- The maximum (launch) latency of the rise edge of the ideal waveform of the clock
- The maximum (launch) latency of the fall edge of the ideal waveform of the clock

If a single value is specified for a clock, then all four edge latencies are set to that single value. If the specified value for a clock is a Tcl list containing two values, then the two values are interpreted as rise latency and fall latency.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this pin, the value of the `clock_network_early_latency` pin attribute applies.

The RTL Compiler tool does not use these attribute values because the tool does not perform any hold timing analysis. However, when you execute the `write_sdc` command, the values specified for the `network_early_latency_by_clock` attribute are written out to the SDC constraints file.

- If only one value was specified for a clock, the following SDC command is written:
`set_clock_latency -clock clock uncertainty -early pin`
- If two values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min uncertainty -early pin  
set_clock_latency -clock clock -max uncertainty -early pin
```

- If four values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min -rise uncertainty -early pin
set_clock_latency -clock clock -min -fall uncertainty -early pin
set_clock_latency -clock clock -max -rise uncertainty -early pin
set_clock_latency -clock clock -max -fall uncertainty -early pin
```

Example

The following command specifies four latency values for each of the following clocks propagating to pin clk: clk1, clk2, clk3, and clk4 on.

```
set_att network_early_latency_by_clock { {clk1 {10 20 30 40}} {clk2 {50 60 70 80}} \
{clk3 {90 100 110 120}} {clk4 {130 140 150 160}} } \
[find /des*/design -pin clk]
```

Related Information

Related attributes:

- (clock) [clock network early latency](#) on page 669
- (pin) [clock network early latency](#) on page 580
- (port) [clock network early latency](#) on page 621
- (port) [network early latency by clock](#) on page 654

network_late_latency_by_clock

```
network_late_latency_by_clock {{clock {min_rise min_fall max_rise
max_fall}}} ... }
```

Read-write [pin](#) attribute. Specifies the delay between the ideal waveform edges of the specified clocks and the sequential elements in the circuit for late-mode (setup checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The attribute value is a Tcl list of Tcl lists for a pin. There can be as many Tcl lists as there are clocks propagating to this pin. The Tcl list for a clock contains

- The name of the clock
- The minimum (capture) latency of the rise edge of the ideal waveform
- The minimum (capture) latency of the fall edge of the ideal waveform

Attribute Reference for Encounter RTL Compiler

Constraint—Pin Attributes

- The maximum (launch) latency of the rise edge of the ideal waveform
- The maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

If a single value is specified for a clock, then all four edge latencies are set to that single value. If the specified value for a clock is a Tcl list containing two values, then the two values are interpreted as rise latency and fall latency.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this pin, the value of the `clock_network_late_latency` pin attribute applies.

To take this attribute into account during timing analysis and optimization, set the following root attributes to `true`:

```
set_attribute use_multi_clks_latency_uncertainty_report true /  
set_attribute use_multi_clks_latency_uncertainty_optimize true /
```

When you execute the `write_sdc` command, the values specified for the `network_late_latency_by_clock` attribute are written out to the SDC constraints file.

- If only one value was specified for a clock, the following SDC command is written:

```
set_clock_latency -clock clock uncertainty -late pin
```

- If two values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min uncertainty -late pin  
set_clock_latency -clock clock -max uncertainty -late pin
```

- If four values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min -rise uncertainty -late pin  
set_clock_latency -clock clock -min -fall uncertainty -late pin  
set_clock_latency -clock clock -max -rise uncertainty -late pin  
set_clock_latency -clock clock -max -fall uncertainty -late pin
```

Example

The following command sets the rise and fall capture, and rise and fall launch network latency numbers for clock `clk3` to 90, 100, 100 and 120 ps at pin `clk` for setup analysis.

```
set_attr network_late_latency_by_clock { {clk3 {90 100 110 120}} } \  
[find /des*/design -pin clk]
```

Related Information

Affects these commands: [report timing](#)
[synthesize](#)

Affected by these attributes: [use multi_clks latency uncertainty optimize](#) on page 541

[use multi_clks latency uncertainty report](#) on page 541

Related attributes: (clock) [clock network late latency](#) on page 670
(pin) [clock network late latency](#) on page 584
(port) [clock network late latency](#) on page 625
(clock) [clock source late latency](#) on page 675
(pin) [clock source late latency](#) on page 594
(port) [clock source late latency](#) on page 636
(port) [network late latency by clock](#) on page 655

setup_uncertainty_by_clock

`setup_uncertainty_by_clock {{clock {rise fall}}...}`

Read-write [pin](#) attribute. Specifies the uncertainty in the arrival times of capturing edges for particular clocks in late-mode timing analysis.

The attribute value is a Tcl list of Tcl lists for a pin. There can be as many Tcl lists as there are clocks propagating to this pin. The Tcl list for a clock contains

- The name of the clock
- The rise and fall uncertainties, respectively. Positive values indicate that the clock may arrive earlier in time, which causes timing constraints to be tighter.

If a single value is specified for a clock, the rise and fall uncertainties are set to that single value.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this pin, the value of the `clock_setup_uncertainty` pin attribute applies.

Attribute Reference for Encounter RTL Compiler

Constraint—Pin Attributes

To take this attribute into account during timing analysis and optimization, set the following root attributes to true:

```
set_attribute use_multi_clks_latency_uncertainty_report true /  
set_attribute use_multi_clks_latency_uncertainty_optimize true /
```

When you execute the `write_sdc` command, the values specified for the `setup_uncertainty_by_clock` attribute are written out to the SDC constraints file.

- If only one uncertainty value is specified for a clock, the following SDC command is written:

```
set_clock_uncertainty -clock clock uncertainty -setup pin
```

- If rise and fall uncertainties were specified, the following SDC commands are written:

```
set_clock_uncertainty -clock clock -rise uncertainty -setup pin  
set_clock_uncertainty -clock clock -fall uncertainty -setup pin
```

Example

The following command specifies the rise and fall uncertainty numbers for clock abc at pin clk.

```
set_attr setup_uncertainty_by_clock { {abc {17 27}} } [find /des*/top -pin clk]
```

Related Information

Affects these commands:

[report_timing](#)

[synthesize](#)

Affected by these attributes:

[use_multi_clks_latency_uncertainty_optimize](#) on page 541

[use_multi_clks_latency_uncertainty_report](#) on page 541

Related attributes:

(pin) [clock_setup_uncertainty](#) on page 588

(port) [clock_setup_uncertainty](#) on page 629

(port) [setup_uncertainty_by_clock](#) on page 657

source_early_latency_by_clock

```
source_early_latency_by_clock {{clock {min_rise min_fall max_rise  
max_fall}}...}
```

Read-write pin attribute. Specifies the delay between the ideal waveform edges of the specified clocks and the sequential elements in the circuit for early-mode (hold checking) timing analysis.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The attribute value is a Tcl list of Tcl lists for a pin. There can be as many Tcl lists as there are clocks propagating to this pin. The Tcl list for a clock contains

- The name of the clock
- The minimum (capture) latency of the rise edge of the ideal waveform of the clock
- The minimum (capture) latency of the fall edge of the ideal waveform of the clock
- The maximum (launch) latency of the rise edge of the ideal waveform of the clock
- The maximum (launch) latency of the fall edge of the ideal waveform of the clock

If a single value is specified for a clock, then all four edge latencies are set to that single value. If the specified value for a clock is a Tcl list containing two values, then the two values are interpreted as rise latency and fall latency.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this pin, the value of the `clock_source_early_latency` pin attribute applies.

The RTL Compiler tool does not use these attribute values because the tool does not perform any hold timing analysis. However, when you execute the `write_sdc` command, the values specified for the `source_early_latency_by_clock` attribute are written out to the SDC constraints file.

- If only one value was specified for a clock, the following SDC command is written:
`set_clock_latency -clock clock uncertainty -source -early pin`

- If two values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min uncertainty -source -early pin  
set_clock_latency -clock clock -max uncertainty -source -early pin
```

- If four values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min -rise uncertainty -source -early pin
set_clock_latency -clock clock -min -fall uncertainty -source -early pin
set_clock_latency -clock clock -max -rise uncertainty -source -early pin
set_clock_latency -clock clock -max -fall uncertainty -source -early pin
```

Example

The following command specifies four source latency values for each of the following clocks propagating to pin *clk* for hold analysis: *clk1*, *clk2*, *clk3*, and *clk4*.

```
set_attr source_early_latency_by_clock { {clk1 {10 20 30 40}} {clk2 {50 60 70 80}} 
{clk3 {90 100 110 120}} {clk4 {130 140 150 160}} } [find /des*/top -pin clk]
```

Related Information

Related attributes:	(clock) clock network early latency on page 669
	(pin) clock network early latency on page 580
	(port) clock network early latency on page 621
	(clock) clock source late latency on page 675
	(pin) clock source late latency on page 594
	(port) clock source late latency on page 636
	(port) clock source early latency on page 673

source_late_latency_by_clock

```
source_late_latency_by_clock {{clock {min_rise min_fall max_rise max_fall}}}...
```

Read-write [pin](#) attribute. Specifies the delay between the ideal waveform edges of the specified clocks and the sequential elements in the circuit for late-mode (setup checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The attribute value is a Tcl list of Tcl lists for a pin. There can be as many Tcl lists as there are clocks propagating to this pin. The Tcl list for a clock contains

- The name of the clock

Attribute Reference for Encounter RTL Compiler

Constraint—Pin Attributes

- The minimum (capture) latency of the rise edge of the ideal waveform
- The minimum (capture) latency of the fall edge of the ideal waveform
- The maximum (launch) latency of the rise edge of the ideal waveform
- The maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

If the specified value for a clock is set to a single delay value, then all four edge latencies are set to that single value. If the attribute is set to a Tcl list containing two values, then the two values are interpreted as rise latency and fall latency.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this pin, the value of the `clock_source_late_latency` pin attribute applies.

To take this attribute into account during timing analysis and optimization, set the following root attributes to `true`:

```
set_attribute use_multi_clks_latency_uncertainty_report true /  
set_attribute use_multi_clks_latency_uncertainty_optimize true /
```

When you execute the `write_sdc` command, the values specified for the `source_late_latency_by_clock` attribute are written out to the SDC constraints file.

- If only one value was specified for a clock, the following SDC command is written:

```
set_clock_latency -clock clock uncertainty -source -late pin
```

- If two values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min uncertainty -source -late pin  
set_clock_latency -clock clock -max uncertainty -source -late pin
```

- If four values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock clock -min -rise uncertainty -source -late pin  
set_clock_latency -clock clock -min -fall uncertainty -source -late pin  
set_clock_latency -clock clock -max -rise uncertainty -source -late pin  
set_clock_latency -clock clock -max -fall uncertainty -source -late pin
```

Example

The following command sets the capture rise and fall, and launch rise and fall source latency numbers for clock `clk3` to 90, 100, 100 and 120 ps at pin `clk` for setup analysis.

```
set_attr source_late_latency_by_clock { {clk3 {90 100 110 120}} } \
[find /des*/design -pin clk]
```

Related Information

Affects these commands:	report timing synthesize
Affected by these attributes:	use multi clks latency uncertainty optimize on page 541 use multi clks latency uncertainty report on page 541
Related attributes:	(clock) clock network late latency on page 670 (pin) clock network late latency on page 584 (port) clock network late latency on page 625 (clock) clock source late latency on page 675 (pin) clock source late latency on page 594 (port) clock source late latency on page 636 (port) source late latency by clock on page 660

timing_case_logic_value

```
timing_case_logic_value {no_value | 0 | 1}
```

Default: no_value

Read-write [pin](#) attribute. Forces the pin to assume the specified logic value for timing analysis purposes. You can set this attribute on mapped leaf (combinational) instance pins, hierarchical boundary pins or outputs pins of sequential cells. The timer automatically sets the logic constants to their appropriate value.

Example

In the following example, the SDC `set_case_analysis` command was applied on the `A` pin of the `nand_inst` instance for mode `a`:

```
rc:/>get_attribute timing_case_logic_value_by_mode nand_inst/A
{/designs/top/modes/a 0}
```

In this case, the `timing_case_logic_value` attribute returns the default `no_value` for pin A:

```
rc:/>get_attribute timing_case_logic_value nand_inst/A  
no_value
```

Related Information

[Combinational Feedback Loops in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands:	<u>report_timing</u> <u>synthesize</u>
Affected by this attribute:	(pin) <u>timing_case_logic_value_by_mode</u> on page 613
Related attributes:	(port) <u>timing_case_logic_value</u> on page 663 (port) <u>timing_case_logic_value_by_mode</u> on page 664

timing_case_logic_value_by_mode

```
timing_case_logic_value_by_mode {mode_name_1 case_analysis_value}  
[ {mode_name_2 case_analysis_value} ]...
```

Read-write `pin` attribute. Forces the timer to assume logic values (0 or 1) for particular timing modes. Specify a Tcl list of modes and corresponding case analysis values.

When the `timing_case_logic_value_by_mode` attribute is set, the `timing_case_logic_value` attribute will be reset to `no_value`. However, when all the values for the `timing_case_logic_value_by_mode` attribute are the same, the `timing_case_logic_value` attribute will be set to that value.

When the `timing_case_logic_value` attribute is set, the `timing_case_logic_value_by_mode` attribute will reflect that value for each mode.

For example:

```
rc:/> dc::set_case_analysis -mode a 0 [dc::get_pins nand_inst/A]  
rc:/> get_attribute timing_case_logic_value_by_mode nand_inst/A  
{/designs/top/modes/a 0}  
rc:/> get_attribute timing_case_logic_value nand_inst/A  
no_value
```

Note: Specifying the `timing_case_logic_value_by_mode` attribute applies the `timing_case_computed_value_by_mode` attribute on the port where it is applied and on the relevant pins of logic downstream.

Example

The following example shows how to set this attribute in modes a and b:

```
rc:/> set_attribute timing_case_logic_value_by_mode {{a 1} {b 0}} \
    /designs/add/instances_comb/n0001D/I1
Setting attribute of pin 'I1': 'timing_case_logic_value_by_mode' =
{/designs/add/modes/b 0} {/designs/add/modes/a 1}
rc:/> get_attr timing_case_logic_value_by_mode n0001D/I1 \
    {/designs/add/modes/b 0} {/designs/add/modes/a 1}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:	report clocks report qor report timing write encounter write sdc write script
Related commands:	create mode read sdc
Affects these attributes:	(pin) timing case logic value on page 612 (port) timing case logic value on page 663
Related attributes:	disabled arcs by mode on page 565 (pin) external delays by mode on page 969 (port) external delays by mode on page 1007 (design) latch borrow by mode on page 552 (instance) latch max borrow by mode on page 572 (pin) propagated clocks by mode on page 976 (port) propagated clocks by mode on page 1015 (design) slack by mode on page 943 (pin) slack by mode on page 980

Attribute Reference for Encounter RTL Compiler

Constraint—Pin Attributes

(port) [slack_by_mode](#) on page 1018

(cost group) [slack_by_mode](#) on page 1060

(pin) [timing_case_computed_value_by_mode](#) on page 985

(port) [timing_case_computed_value_by_mode](#) on page 1021

instance) [timing_case_disabled_arcs](#) on page 953

(instance) [timing_case_disabled_arcs_by_mode](#) on page 954

(port) [timing_case_logic_value_by_mode](#) on page 664

Port Attributes

Contain information about single bit connection points in the specified top-level design. Port objects are stored in `ports_in` and `ports_out` directories.

- To set a port attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -port name]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -port name]
```

break_timing_paths

```
break_timing_paths {false | true | propagate_slews}
```

Default: false

Read-write `port` attribute. Controls whether to break the timing path at the specified port. By default the timing path is not broken.

If the attribute is set to `true`, the timing path is broken and external delays can be set on the specified port and the port can function as `-to`, `-through`, or `-from` point in path exception commands.

If the attribute is set to `false`, the timing path is not broken and the slew values will propagate through the specified port.

If the attribute is set to `propagate_slews`, the slew will propagate from the driver. The load on the net is unaffected by the `break_timing_paths` attribute.

Related Information

[Breaking the Path of a Specified Pin in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related commands:

[multi_cycle](#)
[path_adjust](#)
[path_delay](#)
[path_disable](#)
[specify_paths](#)

Related attributes:

(pin) [break_timing_paths](#) on page 574
(pin) [break_timing_paths_by_mode](#) on page 576
(port) [break_timing_paths_by_mode](#) on page 617
[external_driver](#) on page 641
[fixed_slew](#) on page 647
[slew](#) on page 678

break_timing_paths_by_mode

```
break_timing_paths_by_mode {mode_1 (0 | 1 | no_value)}  
{mode_2 (0 | 1 | no_value)} ...
```

Read-write attribute. Controls whether to break the timing path at the port for particular timing modes.

Note: The [break_timing_paths](#) attribute must be set to `false` for the [break_timing_paths_by_mode](#) attribute to take effect.



If you set the value to `propagate_slews` or `false` in one or more modes, but not in all modes, the slews will be propagated in all modes.

Only if you set the value to `true` or `1` in all modes will the slew not be propagated. The timing paths will be broken.

Related Information

Affects these commands:

[report_clocks](#)
[report_qor](#)
[read_sdc](#)
[report_timing](#)
[write_sdc](#)
[write_script](#)

Attribute Reference for Encounter RTL Compiler

Constraint—Port Attributes

Related attributes:

- (pin) [break_timing_paths](#) on page 574
- (port) [break_timing_paths](#) on page 616
- pin) [break_timing_paths_by_mode](#) on page 576
- (instance) [disabled_arcs](#) on page 564
- (instance) [disabled_arcs_by_mode](#) on page 565
- (pin) [timing_case_computed_value_by_mode](#) on page 985
- (port) [timing_case_computed_value_by_mode](#)
- instance) [timing_case_disabled_arcs](#) on page 953
- (instance) [timing_case_disabled_arcs_by_mode](#) on page 954
- (pin) [timing_case_logic_value](#) on page 612
- (port) [timing_case_logic_value](#) on page 663
- (pin) [timing_case_logic_value_by_mode](#) on page 613
- (port) [timing_case_logic_value_by_mode](#) on page 664

clock_hold_uncertainty

`clock_hold_uncertainty { {no_value no_value} | Tcl_list }`

Default: no_value no_value

Read-write `port` attribute. Specifies the uncertainty in the arrival times of capturing edges (in picoseconds) for the clock in early-mode (hold) timing analysis.

If you specify a single value, both the rising and falling edge are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as the rising and falling edge.

Note: RTL Compiler ignores this value in optimization and timing analysis, but can pass it to downstream tools.

Related Information

[Specifying Clock Skew in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:	(clock) clock_hold_uncertainty on page 668
	(pin) clock_hold_uncertainty on page 578
	(pin) clock_hold_uncertainty_by_mode on page 579
	(port) clock_hold_uncertainty_by_mode on page 620
	(clock) clock_setup_uncertainty on page 672
	(pin) clock_setup_uncertainty on page 588
	(port) clock_setup_uncertainty on page 629
	(pin) hold_uncertainty_by_clock on page 598

clock_hold_uncertainty_by_mode

```
clock_hold_uncertainty_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }
```

Default: {no_value no_value} (for each mode)

Read-write [port](#) attribute. Specifies the uncertainty in the arrival times of capturing edges (in picoseconds) for the clock in early-mode (hold) timing analysis for particular timing modes. RTL Compiler ignores (does not use) these values in optimization and timing analysis, but can pass them to downstream tools.

If you specify a single value, then both the rising and falling edge are set to that single value. If you specify a Tcl list containing two values, then the two values are interpreted as the rising and falling edge.

Note: When you set this attribute, it overrides the `clock_hold_uncertainty` attribute on this port.

Example

The following command sets the rising and falling clock edges for modes A and B on port `clk`.

```
rc:/> set_attr clock_hold_uncertainty_by_mode {{ A {1 2}} {B 3}} [find / -port clk]
Setting attribute of port 'clk': 'clock_hold_uncertainty_by_mode' =
{/designs/top/modes/A {1.0 2.0}}{/designs/top/modes/B {3.0 3.0}}
```

Related Information

[Specifying Clock Skew in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (port) [hold uncertainty by clock](#) on page 648

Affects this attribute: (port) [clock hold uncertainty](#) on page 619

Related attributes: (clock) [clock hold uncertainty](#) on page 668

(pin) [clock hold uncertainty](#) on page 578

(pin) [clock hold uncertainty by mode](#) on page 579

(clock) [clock setup uncertainty](#) on page 672

(pin) [clock setup uncertainty](#) on page 588

(port) [clock setup uncertainty](#) on page 629

(pin) [clock setup uncertainty by mode](#) on page 589

(port) [clock setup uncertainty by mode](#) on page 631

clock_network_early_latency

`clock_network_early_latency { {no_value no_value no_value no_value} | Tcl_list }`

Default: no_value no_value no_value no_value

Read-write port attribute. Specifies the delay (in picoseconds) between the ideal waveform edges and the sequential elements in the circuit for early-mode (hold checking) timing analysis.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and an actual sequential element.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: RTL Compiler ignores this value in optimization and timing analysis, but can pass it to downstream tools.

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:	(clock) <u>clock_network_early_latency</u> on page 669
	(pin) <u>clock_network_early_latency</u> on page 580
	(pin) <u>clock_network_early_latency_by_mode</u> on page 582
	(port) <u>clock_network_early_latency_by_mode</u> on page 623
	(clock) <u>clock_network_late_latency</u> on page 670

Attribute Reference for Encounter RTL Compiler

Constraint—Port Attributes

- (pin) [clock_network_late_latency](#) on page 584
- (port) [clock_network_late_latency](#) on page 625
- (clock) [clock_source_early_latency](#) on page 673
- (pin) [clock_source_early_latency](#) on page 590
- (port) [clock_source_early_latency](#) on page 632
- (clock) [clock_source_late_latency](#) on page 675
- (pin) [clock_source_late_latency](#) on page 594
- (port) [clock_source_late_latency](#) on page 636

clock_network_early_latency_by_mode

```
clock_network_early_latency_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }
```

Default: {no_value no_value no_value no_value} (for each mode)

Read-write port attribute. Specifies the delay at the port between the ideal waveform edges and the sequential elements in the circuit for early-mode (hold checking) timing analysis. For each timing mode, you can specify a Tcl list of up to four delay values.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and an actual sequential element.

The values set on the port both override the latency values for all clock signals that propagate through the port and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: RTL Compiler ignores this value in optimization and timing analysis, but can pass it to downstream tools.

Note: When you set this attribute, it overrides the `clock_network_early_latency` attribute set on the port.

Example

The following command sets the network early latencies for modes A and B on port clk.

```
rc:/> set_attr clock_network_early_latency_by_mode {{ A {150 30}} \
==> {B {75 20 60 15}}} [find / -port clk]
Setting attribute of port 'clk': 'clock_network_early_latency_by_mode' =
{/designs/top/modes/A {150.0 30.0 150.0 30.0}}{/designs/top/modes/B {75.0 20.0
60.0 150.0}}
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (port) [network early latency by clock](#) on page 654

Affects this attribute: (port) [clock network early latency](#) on page 621

Related attributes: (clock) [clock network early latency](#) on page 669

(pin) [clock network early latency](#) on page 580

(pin) [clock network early latency by mode](#) on page 582

(clock) [clock network late latency](#) on page 670

(pin) [clock network late latency](#) on page 584

(port) [clock network late latency](#) on page 625

(pin) [clock network late latency by mode](#) on page 586

(port) [clock network late latency by mode](#) on page 627

(clock) [clock source early latency](#) on page 673

(pin) [clock source early latency](#) on page 590

(port) [clock source early latency](#) on page 632

(pin) [clock source early latency by mode](#) on page 592

(port) [clock source early latency by mode](#) on page 634

(clock) [clock source late latency](#) on page 675

(pin) [clock source late latency](#) on page 594

(port) [clock source late latency](#) on page 636

(pin) [clock source late latency by mode](#) on page 596

(port) [clock source late latency by mode](#) on page 638

clock_network_late_latency

```
clock_network_late_latency { {no_value no_value no_value no_value} | Tcl_list }
```

Default: no_value no_value no_value no_value

Read-write port attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for late-mode (setup checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the port both override the latency values for all clock signals that propagate through the port and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

In addition to numeric delay values, this attribute accepts the special string `no_value` to indicate that no change should be made to the latency. The default value of the attribute is `{no_value no_value no_value no_value}`, which indicates that all four latency values should remain unchanged.

If this attribute is set to a single delay value, then all four edge latencies are set to that single value. If the attribute is set to a Tcl list containing two values, then the two values are interpreted as rise latency and fall latency.

Examples

- Both of the following commands have the same effect:

```
rc:/> set_attribute clock_network_late_latency {150 30} [find / -port a]  
rc:/> set_attribute clock_network_late_latency {150 30 150 30} [find / -port a]
```

- The following command sets the network fall latency to 50 picoseconds but leaves the network rise latency unchanged:

```
rc:/> set_attribute clock_network_late_latency {no_value 50} [find / -port a]
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:	(clock) clock_network_early_latency on page 669 (pin) clock_network_early_latency on page 580 (port) clock_network_early_latency on page 621 (clock) clock_network_late_latency on page 670 (pin) clock_network_late_latency on page 584 (pin) clock_network_late_latency_by_mode on page 586 (port) clock_network_late_latency_by_mode on page 627 (clock) clock_source_early_latency on page 673 (pin) clock_source_early_latency on page 590 (port) clock_source_early_latency on page 632 (clock) clock_source_late_latency on page 675 (pin) clock_source_late_latency on page 594 (port) clock_source_late_latency on page 636
---------------------	--

clock_network_late_latency_by_mode

```
clock_network_late_latency_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }
```

Default: {no_value no_value no_value no_value} (for each mode)

Read-write port attribute. Specifies the delay at the port (in picoseconds) between the ideal waveform edges and the sequential elements in the circuit for late-mode (setup checking) timing analysis. For each timing mode, you can specify a Tcl list of up to four delay values.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the port both override the latency values for all clock signals that propagate through the port and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

In addition to numeric delay values, this attribute accepts the special string `no_value` to indicate that no change should be made to the latency. The default value of the attribute is `{no_value no_value no_value no_value}`, which indicates that all four latency values should remain unchanged.

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: When you set this attribute, it overrides the `clock_network_late_latency` attribute set on this port.

Example

The following command sets the network late latencies for modes A and B on port CK.

```
rc:/> set_attr clock_network_late_latency_by_mode {{ A {150 30}} \
==> {B {75 20 60 15}}} [find / -port CK]
Setting attribute of port 'CK': 'clock_network_late_latency_by_mode' =
{/designs/top/modes/A {150.0 30.0 150.0 30.0}}{/designs/top/modes/B {75.0 20.0
60.0 15.0}}
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (port) [network late latency by clock](#) on page 605

Affects this attribute: (port) [clock network late latency](#) on page 625

Related attributes: (clock) [clock network early latency](#) on page 669

(pin) [clock network early latency](#) on page 580

(port) [clock network early latency](#) on page 621

(pin) [clock network early latency by mode](#) on page 582

(port) [clock network early latency by mode](#) on page 623

(clock) [clock network late latency](#) on page 670

(pin) [clock network late latency](#) on page 584

(pin) [clock network late latency by mode](#) on page 586

(clock) [clock source early latency](#) on page 673

(pin) [clock source early latency](#) on page 590

(port) [clock source early latency](#) on page 632

(pin) [clock source early latency by mode](#) on page 592

(port) [clock source early latency by mode](#) on page 634

(clock) [clock source late latency](#) on page 675

(pin) [clock source late latency](#) on page 594

(port) [clock source late latency](#) on page 636

(pin) [clock source late latency by mode](#) on page 596

(port) [clock source late latency by mode](#) on page 638

clock_setup_uncertainty

```
clock_setup_uncertainty { {no_value no_value} | Tcl_list }
```

Default: no_value no_value

Read-write port attribute. Specifies the uncertainty in the arrival times of capturing edges for clocks in late-mode timing analysis. The values set on the port both override the uncertainty values for all clock signals that propagate through the port and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list specifying the uncertainty in the rising and falling edges of the ideal clock waveform. Positive values indicate that the clock may arrive earlier in time, which causes timing constraints to be tighter.

In addition to numeric delay values, this attribute accepts the special string no_value to indicate that no change should be made to the uncertainty times. The default value of this attribute is {no_value no_value}, indicating that both rise and fall uncertainties should be left unchanged.

If this attribute is set to a single value (instead of a Tcl list representing the rise and fall values), then both rise and fall values are set to the single value.

Examples

- Both of the following commands set the rise and fall setup uncertainty arrival times for port a to 100 picoseconds:

```
rc:/> set_attribute clock_setup_uncertainty {100 100} [find / -port a]  
rc:/> set_attribute clock_setup_uncertainty 100 [find / -port a]
```

- The following command sets the rise setup uncertainty to 100 picoseconds and the fall setup uncertainty to 50 picoseconds:

```
rc:/> set_attribute clock_setup_uncertainty {100 50} [find / -port a]
```

- The following command sets the rise setup uncertainty to 50 picoseconds but leaves the fall uncertainty unchanged:

```
rc:/> set_attribute clock_setup_uncertainty {50 no_value} [find / -port a]
```

Attribute Reference for Encounter RTL Compiler

Constraint—Port Attributes

Related Information

[Specifying Clock Skew in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

- Related attributes:
- (clock) [clock_hold_uncertainty](#) on page 668
 - (pin) [clock_hold_uncertainty](#) on page 578
 - (port) [clock_hold_uncertainty](#) on page 619
 - (clock) [clock_setup_uncertainty](#) on page 672
 - (pin) [clock_setup_uncertainty](#) on page 588
 - (pin) [clock_setup_uncertainty_by_mode](#) on page 589
 - (port) [clock_setup_uncertainty_by_mode](#) on page 631

clock_setup_uncertainty_by_mode

```
clock_setup_uncertainty_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }
```

Default: {no_value no_value} (for each mode)

Read-write port attribute. Specifies the uncertainty in the arrival times of capturing edges for clocks in late-mode timing analysis. The values set on a port both override the uncertainty values for all clock signals that propagate through the port and apply to all sequential elements in the transitive fanout. For each timing mode, you can specify a Tcl list of up to two values.

The attribute value is a Tcl list specifying the uncertainty in the rising and falling edges of the ideal clock waveform. Positive values indicate that the clock may arrive earlier in time, which causes timing constraints to be tighter.

In addition to numeric delay values, this attribute accepts the special string no_value to indicate that no change should be made to the uncertainty times. The default value of this attribute is {no_value no_value}, indicating that both rise and fall uncertainties should be left unchanged.

If you specify a single value (instead of a Tcl list representing the rise and fall values), both rise and fall values are set to the single value.

Note: When you set this attribute, it overrides the `clock_setup_uncertainty` attribute set on this port.

Examples

- The following command sets the setup uncertainties for modes A and B:

```
rc:/> set_attr clock_setup_uncertainty_by_mode {{ A {100 50}} {B 50}} \
==> [find / -port CK]
      Setting attribute of port 'CK': 'clock_setup_uncertainty_by_mode' =
{/designs/top/modes/A {100.0 50.0}} {/designs/top/modes/B {50.0 50.0}}
```

Related Information

[Specifying Clock Skew in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (port) [setup uncertainty by clock](#) on page 657

Affects this attribute: (port) [clock setup uncertainty](#) on page 629

Related attributes: (clock) [clock hold uncertainty](#) on page 668

(pin) [clock hold uncertainty](#) on page 578

- (port) [clock_hold_uncertainty](#) on page 619
- (pin) [clock_hold_uncertainty_by_mode](#) on page 579
- (port) [clock_hold_uncertainty_by_mode](#) on page 620
- (clock) [clock_setup_uncertainty](#) on page 672
- (pin) [clock_setup_uncertainty](#) on page 588
- (pin) [clock_setup_uncertainty_by_mode](#) on page 589

clock_source_early_latency

`clock_source_early_latency { {no_value no_value no_value no_value} | Tcl_list }`

Default: no_value no_value no_value no_value

Read-write [port](#) attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for early-mode (hold checking) timing analysis.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and an actual sequential element.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: RTL Compiler ignores this value in optimization and timing analysis, but can pass it to downstream tools.

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

- Related attributes:
- (clock) [clock network early latency](#) on page 669
 - (pin) [clock network early latency](#) on page 580
 - (port) [clock network early latency](#) on page 621
 - (clock) [clock network late latency](#) on page 670
 - (pin) [clock network late latency](#) on page 584
 - (port) [clock network late latency](#) on page 625
 - (clock) [clock source early latency](#) on page 673
 - (pin) [clock source early latency](#) on page 590
 - (pin) [clock source early latency by mode](#) on page 592
 - (port) [clock source early latency by mode](#) on page 634
 - (clock) [clock source late latency](#) on page 675
 - (pin) [clock source late latency](#) on page 594
 - (port) [clock source late latency](#) on page 636

clock_source_early_latency_by_mode

```
clock_source_early_latency_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }
```

Default: {no_value no_value no_value no_value} (for each mode)

Read-write port attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for early-mode (hold checking) timing analysis. For each timing mode, you can specify a Tcl list of up to four delay values.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the port both override the latency values for all clock signals that propagate through the port and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

The RTL Compiler tool does not use these attribute values because the tool does not perform any hold timing analysis. However, when you execute the `write_sdc` command, the values specified for the `clock_source_early_latency_by_mode` attribute are written out to the SDC constraints file.

Note: When you set this attribute, it overrides the `clock_source_early_latency` attribute set on this port.

Example

```
rc:/> set_attr clock_source_early_latency_by_mode {{ A {100 50 150 60}} \
==> {B {110 70}}} [find / -port CK]
Setting attribute of port 'CK': 'clock_source_early_latency_by_mode' =
{/designs/top/modes/A {100.0 50.0 150.0 60.0}}{/designs/top/modes/B {110.0 70.0
110.0 70.0}}
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (port) [source_early_latency_by_clock](#) on page 659

Affects this attribute: (port) [clock_source_early_latency](#) on page 632

Related attributes: (clock) [clock_network_early_latency](#) on page 669

(pin) [clock_network_early_latency](#) on page 580

(port) [clock_network_early_latency](#) on page 621

(pin) [clock_network_early_latency_by_mode](#) on page 582

(port) [clock_network_early_latency_by_mode](#) on page 623

(clock) [clock_network_late_latency](#) on page 670

(pin) [clock_network_late_latency](#) on page 584

(port) [clock_network_late_latency](#) on page 625

(pin) [clock_network_late_latency_by_mode](#) on page 586

(port) [clock_network_late_latency_by_mode](#) on page 627

(clock) [clock_source_early_latency](#) on page 673

(pin) [clock_source_early_latency](#) on page 590

(pin) [clock_source_early_latency_by_mode](#) on page 592

(clock) [clock_source_late_latency](#) on page 675

(pin) [clock_source_late_latency](#) on page 594

(port) [clock_source_late_latency](#) on page 636

(pin) [clock_source_late_latency_by_mode](#) on page 596

(port) [clock_source_late_latency_by_mode](#) on page 638

clock_source_late_latency

```
clock_source_late_latency { {no_value no_value no_value no_value} | Tcl_list }
```

Default: no_value no_value no_value no_value

Read-write [port](#) attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for late-mode (setup checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the port both override the latency values for all clock signals that propagate through the port and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

In addition to numeric delay values, this attribute accepts the special string `no_value` to indicate that no change should be made to the latency. The default value of the attribute is `{no_value no_value no_value no_value}`, which indicates that all four latency values should remain unchanged.

If this attribute is set to a single delay value, then all four edge latencies are set to that single value. If the attribute is set to a Tcl list containing two values, then the two values are interpreted as rise latency and fall latency.

Examples

All of the following commands have the same effect:

```
rc:/> set_attribute clock_source_late_latency 150 [find / -port a]
rc:/> set_attribute clock_source_late_latency {150 150} [find / -port a]
rc:/> set_attribute clock_source_late_latency {150 150 150 150} [find / -port a]
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:

- (clock) [clock network early latency](#) on page 669
- (pin) [clock network early latency](#) on page 580
- (port) [clock network early latency](#) on page 621
- (clock) [clock network late latency](#) on page 670
- (pin) [clock network late latency](#) on page 584
- (port) [clock network late latency](#) on page 625
- (clock) [clock source early latency](#) on page 673
- (pin) [clock source early latency](#) on page 590
- (port) [clock source early latency](#) on page 632
- (clock) [clock source late latency](#) on page 675
- (pin) [clock source late latency](#) on page 594
- (pin) [clock source late latency by mode](#) on page 596
- (port) [clock source late latency by mode](#) on page 638

clock_source_late_latency_by_mode

```
clock_source_late_latency_by_mode { {mode_1 Tcl_list} {mode_2 Tcl_list} ... }
```

Default: {no_value no_value no_value no_value} (for each mode)

Read-write [port](#) attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for late-mode (setup checking) timing analysis. For each timing mode, you can specify a Tcl list of up to four delay values.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The values set on the port both override the latency values for all clock signals that propagate through the port and apply to all sequential elements in the transitive fanout.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

In addition to numeric delay values, this attribute accepts the special string `no_value` to indicate that no change should be made to the latency. The default value of the attribute is `{no_value no_value no_value no_value}`, which indicates that all four latency values should remain unchanged.

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: When you set this attribute, it overrides the `clock_source_late_latency` attribute on this port.

Attribute Reference for Encounter RTL Compiler

Constraint—Port Attributes

Examples

```
rc:/> set_attr clock_source_late_latency_by_mode {{ A {100 50 150 60}} \ 
==> {B {110 70}}} [find / -port CK]
Setting attribute of port 'CK': 'clock_source_late_latency_by_mode' =
{/designs/top/modes/A {100.0 50.0 150.0 60.0}}{/designs/top/modes/B {110.0 70.0
110.0 70.0}}
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: [\(port\) source late latency by clock](#) on page 660

Affects this attribute: [\(port\) clock source late latency](#) on page 636

Related attributes: [\(clock\) clock network early latency](#) on page 669

[\(pin\) clock network early latency](#) on page 580

[\(port\) clock network early latency](#) on page 621

[\(pin\) clock network early latency by mode](#) on page 582

[\(port\) clock network early latency by mode](#) on page 623

[\(clock\) clock network late latency](#) on page 670

[\(pin\) clock network late latency](#) on page 584

[\(port\) clock network late latency](#) on page 625

[\(pin\) clock network late latency by mode](#) on page 586

[\(port\) clock network late latency by mode](#) on page 627

[\(clock\) clock source early latency](#) on page 673

[\(pin\) clock source early latency](#) on page 590

[\(port\) clock source early latency](#) on page 632

[\(pin\) clock source early latency by mode](#) on page 592

[\(port\) clock source early latency by mode](#) on page 634

[\(clock\) clock source late latency](#) on page 675

[\(pin\) clock source late latency](#) on page 594

[\(pin\) clock source late latency by mode](#) on page 596

[\(port\) clock source late latency by mode](#) on page 638

external_driven_pin_fall

`external_driven_pin_fall string`

Read-write port attribute. Specifies the input pin of the external object that is driven in case of a fall transition. This information is useful for fall slope sensitivity modeling on output ports. You must specify a library pin object.

Example

The following command specifies to assume that the output ports are driving input A of cell AN2.

```
set_attribute external_driven_pin_fall [find [find / -libcell AN2] -libpin A] \
    /designs/des/ports_out/*
```

Related information

Related attribute: [external_driven_pin_rise](#) on page 640

external_driven_pin_rise

`external_driven_pin_rise string`

Read-write port attribute. Specifies the input pin of the external object that is driven in case of a rise transition. This information is useful for rise slope sensitivity modeling on output ports. You must specify a library pin object.

Example

The following command specifies to assume that the output ports are driving input A of cell AN2.

```
set_attribute external_driven_pin_rise [find [find / -libcell AN2] -libpin A] \
    /designs/des/ports_out/*
```

Related information

Related attribute: [external_driven_pin_fall](#) on page 640

external_driver

`external_driver Tcl_list`

Read-write `port` attribute. Specifies a string that corresponds to a library pin object or objects for the minimum rise, minimum fall, maximum rise, and maximum fall timing modes. The indicated library pin is assumed to be driving the port externally for the purposes of timing calculation and design rule checking in the given timing mode.

You can specify either 1, 2, or 4 values for the library pins. If you specify different values, you must do so as a Tcl list (within braces).

Note: This attribute applies only to input ports. RTL Compiler ignores (does not use) the minimum values in optimization and timing analysis, but can pass them to downstream tools.

Example

```
set_attribute external_driver [find [find / -libcell AN2] -libpin Z] \
    /designs/des/ports_in/*
```

Related information

[Setting External Driver and Load Constraints](#) in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affected by this attribute: [ignore external driver drc](#) on page 650

Related attributes: [external driver from pin](#) on page 641

[external driver input slew](#) on page 642

[external non tristate drivers](#) on page 643

external_driver_from_pin

`external_driver_from_pin Tcl_list`

Read-write `port` attribute. Indicates which input pin to use when modeling the transition on the port driven by the external driver. The `external_driver libpin` and the `external_driver_from_pin` attributes must be set to libpins from the same libcell.

Note: RTL Compiler ignores (does not use) the minimum values in optimization and timing analysis, but they can pass them to downstream tools.

Example

The following command specifies to assume that the input port `in1[0]` is driving input A of cell AN2.

```
set_attribute external_driver_from_pin [find [find / -libcell AN2] -libpin A] \
    /designs/des/ports_in/in1[0]
```

Related information

[Modifying the External Driver in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute:

[ignore_external_driver_drc](#) on page 650

Related attributes:

[external_driver](#) [on page 641](#)

[external_driver_input_slew](#) [on page 642](#)

[external_non_tristate_drivers](#) [on page 643](#)

external_driver_input_slew

```
external_driver_input_slew {no_value | integer}
```

Default: no_value

Read-write [port](#) attribute. Specifies the rise and fall values, respectively, for an input of an external driver in picoseconds.

If you specify different rise and fall slew values, you must do so as a Tcl list (within braces).

Example

The following specifies a rise slew value of 100 picoseconds and a fall slew value of 110 picoseconds for the input of an external driver `in1[0]`:

```
rc:/> set_attribute external_driver_input_slew {100 110} \
    /designs/example_design/ports_in/in1[0]
```

The following example specifies a rise slew and fall slew value of 100 picoseconds for the input of an external driver `in1[1]`:

```
rc:/> set_attribute external_driver_input_slew 100 \
    /designs/example_design/ports_in/in1[1]
```

Related information

[Modifying the External Driver in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attribute: [external_driver](#) on page 641

external_fanout_load

`external_fanout_load {no_value | float}`

Default: no_value

Read-write port attribute. Indicates the fanout load seen by the port outside the design. This information is used by the maximum and minimum fanout design rules. The resolution is 1/1000.

Related information

Related attributes: (design) [max_fanout](#) on page 558
(port) [max_fanout](#) on page 652

external_non_tristate_drivers

`external_non_tristate_drivers integer`

Default: 0

Read-write port attribute. Specifies the number of parallel driving pins.

Related information

[Modifying the External Driver in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attribute: [external_driver](#) on page 641

external_pin_cap

`external_pin_cap float`

Default: no_value

Read-write `port` attribute. Indicates the external capacitive load (in femtofarads) due to pins that are connected to this port.

Related information

[Modifying the External Driver in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:

[external_wire_cap](#) on page 645

[external_wire_res](#) on page 645

[external_wireload_fanout](#) on page 646

[external_wireload_model](#) on page 646

external_resistance

`external_resistance { {no_value no_value no_value no_value} | Tcl_list }`

Default: {no_value no_value no_value no_value}

Read-write `port` attribute. Specifies the resistance of the external driver in kilo ohm for a minimum rise, minimum fall, maximum rise and maximum fall transition. The resolution is 1/1000.

If you specify different resistance values, you must do so as a Tcl list (within braces).

Note: RTL Compiler ignores (does not use) the minimum values in optimization and timing analysis, but they can be passed to downstream tools.

Example

- The following example sets the fixed minimum rise value to be 1.0, minimum fall value to be 1.2, maximum rise value to be 1.1 and maximum fall value to be 1.2 on the port `in1[0]`:

```
rc:/> set_attribute external_resistance {1.0 1.2 1.1 1.2} \
      /designs/example_design/ports_in/in1[0]
```

Related information

Related attributes: [fixed_slew](#) on page 647

external_wire_cap

`external_wire_cap {no_value | float}`

Default: no_value

Read-write [port](#) attribute. Specifies the capacitance (in femtofarads) of the external wire connected to this port. The resolution is 1/10.

Related information

Related attributes: [external_pin_cap](#) on page 644
[external_wire_res](#) on page 645
[external_wireload_fanout](#) on page 646
[external_wireload_model](#) on page 646

external_wire_res

`external_wire_res {no_value | float}`

Default: no_value

Read-write [port](#) attribute. Specifies the resistance (in kilo ohm) of the external wire connected to this port. The resolution is 1/1000.

Related information

Related attributes: [external_pin_cap](#) on page 644
[external_wire_cap](#) on page 645
[external_wireload_fanout](#) on page 646
[external_wireload_model](#) on page 646

external_wireload_fanout

`external_wireload_fanout {no_value | integer}`

Default: no_value

Read-write `port` attribute. Specifies the number of fanouts for this port outside the design.

Related information

Related attributes:	external_pin_cap on page 644 external_wire_cap on page 645 external_wire_res on page 645 external_wireload_model on page 646
---------------------	---

external_wireload_model

`external_wireload_model string`

Read-write `port` attribute. Specifies the wire-load model to use for this port.

Related information

[Assigning a Different Wireload to the Input and Output Ports in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by these attributes:	(design) force_wireload on page 547 (subdesign) force_wireload on page 666
-------------------------------	---

Related attributes:	external_pin_cap on page 644 external_wire_cap on page 645 external_wire_res on page 645 external_wireload_fanout on page 646
---------------------	--

fixed_slew

```
fixed_slew { {no_value no_value no_value no_value} | Tcl_list }
```

Default: {no_value no_value no_value no_value}

Read-write port attribute. Specifies fixed minimum rise, minimum fall, maximum rise and maximum fall slew times, respectively, for the specified port in picoseconds.

If you specify different slew values, you must do so as a Tcl list (within braces).

Note: RTL Compiler ignores (does not use) the minimum values in optimization and timing analysis, but can pass them to downstream tools.

Example

- The following example sets the fixed minimum rise value to be 100, minimum fall value to be 110, maximum rise value to be 110 and maximum fall value to be 120 on the port in1[0]:

```
rc:/> set_attribute fixed_slew {100 110 110 120} \
      /designs/example_design/ports_in/in1[0]
```

- The following example specifies a maximum rise slew value of 100 and a maximum fall slew value of 110 on the input port, in1[0]:

```
rc:/> set_attribute fixed_slew {100 110} \
      /designs/example_design/ports_in/in1[0]
```

- The following example specifies a maximum rise slew and a maximum fall slew value of 100 on the input port, in1[0]:

```
rc:/> set_attribute fixed_slew 100 /designs/example_design/ports_in/in1[0]
```

Related information

Related attributes: [external_driver](#) on page 641

[external_resistance](#) on page 644

hold_uncertainty_by_clock

```
hold_uncertainty_by_clock {{clock {rise fall}}...}
```

Read-write port attribute. Specifies the uncertainty in the arrival times of the capturing edges of the specified clocks in early-mode timing analysis.

The attribute value is a Tcl list of Tcl lists for a port. There can be as many Tcl lists as there are clocks propagating to this port. The Tcl list for a clock contains

- The name of the clock
- The rise and fall uncertainties, respectively

If a single value is specified for a clock, then the rise and fall uncertainties are set to that single value.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this port, the value of the `clock_hold_uncertainty` port attribute applies.

The RTL Compiler tool does not use these attribute values because the tool does not perform any hold timing analysis. However, when you execute the `write_sdc` command, the values specified for the `hold_uncertainty_by_clock` attribute are written out to the SDC constraints file.

- If only one uncertainty value is specified for a clock, the following SDC command is written:

```
set_clock_uncertainty -clock clock uncertainty -hold port
```

- If rise and fall uncertainties were specified, the following SDC commands are written:

```
set_clock_uncertainty -clock clock -rise uncertainty -hold port
set_clock_uncertainty -clock clock -fall uncertainty -hold port
```

Example

The following command specifies the rise and fall uncertainty numbers for clock abc at port clk.

```
set_attr hold_uncertainty_by_clock { {abc {17 27}} } [find /des*/top -port clk]
```

Related Information

Related attributes:

(clock) [clock_hold_uncertainty](#) on page 668

(pin) [clock_hold_uncertainty](#) on page 578

(port) [clock_hold_uncertainty](#) on page 619

(pin) [hold_uncertainty_by_clock](#) on page 598

ideal_driver

```
ideal_driver {false | true}
```

Default: false

Read-write port attribute. Indicates if the port is to be considered an ideal driver. If the port is an ideal driver, the timer and optimizer will not attempt to optimize any net driven by this pin. Therefore, transitions, connect delay, and design rule constraints for this pin are ignored. This attribute is available on both mapped and unmapped netlists.

Note: This attribute does not propagate to the fanout drivers.

Related Information

[Handling Ideal Nets in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: [ideal_seq_async_pins](#) on page 548

Related attribute: (pin) [ideal_driver](#) on page 599

ideal_network

```
ideal_network {false | true}
```

Default: false

Read-write port attribute. Sets the network of the specified driver port to an ideal network. The port must be a driving port. This attribute propagates through combinational gates, and hierarchical boundaries, unlike the `ideal_driver` attribute.

If an ideal signal arrives at a multi-input instance, the output of that instance is considered ideal if all inputs of the instance are ideal.

Example

The following example sets the network to which the `foo` port is connected, to an ideal network:

```
rc:/> set_attribute ideal_network true{/designs/example/ports_in/foo}
```

Related Information

Affects this attribute:	propagated_ideal_network on page 978
Related attributes:	(pin) ideal_driver on page 599
	(port) ideal_driver on page 649
	(pin) ideal_network on page 600

ignore_external_driver_drc

`ignore_external_driver_drc {false | true}`

Default: false

Read-write [port](#) attribute. Indicates whether to use or ignore the design rule constraints specified on the external driver.

Related information

Affects this command:	synthesize -incremental
Affects this attribute:	external_driver on page 641

max_capacitance

```
max_capacitance {no_value | float}
```

Default: no_value

Read-write port attribute. Specifies the maximum capacitance design rule constraint in femtofarads for the net connected to the port. The resolution is 1/1000. When optimizing a design, RTL Compiler attempts to satisfy all design rule constraints. These constraints can come from attributes on a block or port, or from the technology library.

If the attribute is set to no_value, no port constraint is applied, although design rules may still be inferred from block attributes or from any driving pin that has been applied to the port (using the external_driver attribute).

Related Information

Modifying the External Driver in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affects these commands:

[report design rules](#)

[synthesize -incremental](#)

Affected by these attributes:

[ignore_library_drc](#) on page 549

(design) [max_capacitance](#) on page 557

Related attributes:

[drc_first](#) on page 529

(design) [max_capacitance](#) on page 557

(libpin) [max_capacitance](#) on page 220

(design) [max_fanout](#) on page 558

(libpin) [max_fanout](#) on page 220

(port) [max_fanout](#) on page 652

(design) [max_transition](#) on page 653

(libpin) [max_transition](#) on page 220

(port) [max_transition](#) on page 653

max_fanout

`max_fanout {no_value | float}`

Default: no_value

Read-write [port](#) attribute. Specifies the maximum fanout design rule limit for the net connected to the port. The resolution is 1/1000. When optimizing a design, RTL Compiler attempts to satisfy all design rule constraints. These constraints can come from attributes on a module or port, or from the technology library.

If set to no_value, no port constraint is applied, although design rules may still be inferred from module attributes or from any driving pin that has been applied to the port (using the [external_driver](#) attribute).

Related Information

[Specifying the Maximum Fanout](#) in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affects these commands:

[report design rules](#)

[synthesize -incremental](#)

Affected by these attributes:

[ignore_library_drc](#) on page 549

[ignore_library_max_fanout](#) on page 550

(design) [max_fanout](#) on page 558

Related attributes:

[drc_first](#) on page 529

(design) [max_capacitance](#) on page 557

(libpin) [max_capacitance](#) on page 220

(port) [max_capacitance](#) on page 651

(design) [max_fanout](#) on page 558

(libpin) [max_fanout](#) on page 220

(design) [max_transition](#) on page 653

(libpin) [max_transition](#) on page 220

(port) [max_transition](#) on page 653

max_transition

`max_transition {no_value | integer}`

Default: no_value

Read-write `port` attribute. Specifies the maximum transition design rule limit (in picoseconds) for the net connected to the port. The resolution is 1. When optimizing a design, RTL Compiler attempts to satisfy all design rule constraints. These constraints can come from attributes on a module or port, or from the technology library.

If set to no_value, no port constraint is applied, although design rules may still be inferred from module attributes or from any driving pin that has been applied to the port (using the `external_driver` attribute).

Note: The specified value for the `max_transition` attribute must be at least 55 ps.

Related Information

[Controlling the Use of Design Rule Constraints From the Technology Library in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affects these commands:	report design_rules synthesize -incremental
Affected by these attributes:	ignore_library_drc on page 549 (design) max_transition on page 653
Related attributes:	drc_first on page 529 (design) max_capacitance on page 557 (libpin) max_capacitance on page 220 (port) max_capacitance on page 651 (design) max_fanout on page 558 (libpin) max_fanout on page 220 (port) max_fanout on page 652 (design) max_transition on page 559 (libpin) max_transition on page 220

network_early_latency_by_clock

```
network_early_latency_by_clock {{clock {min_rise min_fall max_rise  
max_fall}}...}
```

Read-write port attribute. Specifies the delay between the ideal waveform edges of the specified clocks and the sequential elements in the circuit for early-mode (hold checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The attribute value is a Tcl list of Tcl lists for a port. There can be as many Tcl lists as there are clocks propagating to this port. The Tcl list for a clock contains

- The name of the clock
- The minimum (capture) latency of the rise edge of the ideal waveform of the clock
- The minimum (capture) latency of the fall edge of the ideal waveform of the clock
- The maximum (launch) latency of the rise edge of the ideal waveform of the clock
- The maximum (launch) latency of the fall edge of the ideal waveform of the clock

If a single value is specified for a clock, then all four edge latencies are set to that single value. If the specified value for a clock is a Tcl list containing two values, then the two values are interpreted as rise latency and fall latency.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this port, the value of the `clock_network_early_latency` port attribute applies.

The RTL Compiler tool does not use these attribute values because the tool does not perform any hold timing analysis. However, when you execute the `write_sdc` command, the values specified for the `network_early_latency_by_clock` attribute are written out to the SDC constraints file.

- If only one value was specified for a clock, the following SDC command is written:
`set_clock_latency -clock clock uncertainty -early port`
- If two values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock clock -min uncertainty -early port  
set_clock_latency -clock clock clock -max uncertainty -early port
```

- If four values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min -rise uncertainty -early port
set_clock_latency -clock clock -min -fall uncertainty -early port
set_clock_latency -clock clock -max -rise uncertainty -early port
set_clock_latency -clock clock -max -fall uncertainty -early port
```

Example

The following command specifies four latency values for each of the following clocks propagating to port clk: clk1, clk2, clk3, and clk4 on.

```
set_att network_early_latency_by_clock { {clk1 {10 20 30 40}} {clk2 {50 60 70 80}} \
{clk3 {90 100 110 120}} {clk4 {130 140 150 160}} } \
[find /des*/design -port clk]
```

Related Information

Related attributes:

- (clock) [clock network early latency](#) on page 669
- (pin) [clock network early latency](#) on page 580
- (port) [clock network early latency](#) on page 621
- (pin) [network early latency by clock](#) on page 604

network_late_latency_by_clock

```
network_late_latency_by_clock {{clock {min_rise min_fall max_rise
max_fall}}} ... }
```

Read-write port attribute. Specifies the delay between the ideal waveform edges of the specified clocks and the sequential elements in the circuit for late-mode (setup checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The attribute value is a Tcl list of Tcl lists for a pin. There can be as many Tcl lists as there are clocks propagating to this pin. The Tcl list for a clock contains

- The name of the clock
- The minimum (capture) latency of the rise edge of the ideal waveform
- The minimum (capture) latency of the fall edge of the ideal waveform

Attribute Reference for Encounter RTL Compiler

Constraint—Port Attributes

- The maximum (launch) latency of the rise edge of the ideal waveform
- The maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

If a single value is specified for a clock, then all four edge latencies are set to that single value. If the specified value for a clock is a Tcl list containing two values, then the two values are interpreted as rise latency and fall latency.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this port, the value of the `clock_network_late_latency` port attribute applies.

To take this attribute into account during timing analysis and optimization, set the following root attributes to `true`:

```
set_attribute use_multi_clks_latency_uncertainty_report true /  
set_attribute use_multi_clks_latency_uncertainty_optimize true /
```

When you execute the `write_sdc` command, the values specified for the `network_late_latency_by_clock` attribute are written out to the SDC constraints file.

- If only one value was specified for a clock, the following SDC command is written:

```
set_clock_latency -clock clock uncertainty -late port
```

- If two values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min uncertainty -late port  
set_clock_latency -clock clock -max uncertainty -late port
```

- If four values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min -rise uncertainty -late port  
set_clock_latency -clock clock -min -fall uncertainty -late port  
set_clock_latency -clock clock -max -rise uncertainty -late port  
set_clock_latency -clock clock -max -fall uncertainty -late port
```

Example

The following command sets the rise and fall capture, and rise and fall launch network latency numbers for clock `clk3` to 90, 100, 100 and 120 ps at port `clk` for setup analysis.

```
set_attr network_late_latency_by_clock { {clk3 {90 100 110 120}} } \  
[find /des*/design -port clk]
```

Related Information

Affects these commands: [report_timing](#)
 [synthesize](#)

Affected by these attributes: [use_multi_clks_latency_uncertainty_optimize](#) on page 541
 [use_multi_clks_latency_uncertainty_report](#) on page 541

Related attributes: (clock) [clock_network_late_latency](#) on page 670
 (pin) [clock_network_late_latency](#) on page 584
 (port) [clock_network_late_latency](#) on page 625
 (clock) [clock_source_late_latency](#) on page 675
 (pin) [clock_source_late_latency](#) on page 594
 (port) [clock_source_late_latency](#) on page 636
 (pin) [network_late_latency_by_clock](#) on page 605

setup_uncertainty_by_clock

`setup_uncertainty_by_clock {{clock {rise fall}}...}`

Read-write [port](#) attribute. Specifies the uncertainty in the arrival times of capturing edges for particular clocks in late-mode timing analysis.

The attribute value is a Tcl list of Tcl lists for a port. There can be as many Tcl lists as there are clocks propagating to this port. The Tcl list for a clock contains

- The name of the clock
- The rise and fall uncertainties, respectively. Positive values indicate that the clock may arrive earlier in time, which causes timing constraints to be tighter.

If a single value is specified for a clock, then the rise and fall uncertainties are set to that single value.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this port, the value of the `clock_setup_uncertainty` port attribute applies.

Attribute Reference for Encounter RTL Compiler

Constraint—Port Attributes

To take this attribute into account during timing analysis and optimization, set the following root attributes to true:

```
set_attribute use_multi_clks_latency_uncertainty_report true /  
set_attribute use_multi_clks_latency_uncertainty_optimize true /
```

When you execute the `write_sdc` command, the values specified for the `setup_uncertainty_by_clock` attribute are written out to the SDC constraints file.

- If only one uncertainty value is specified for a clock, the following SDC command is written:

```
set_clock_uncertainty -clock clock uncertainty -setup port
```

- If rise and fall uncertainties were specified, the following SDC commands are written:

```
set_clock_uncertainty -clock clock -rise uncertainty -setup port  
set_clock_uncertainty -clock clock -fall uncertainty -setup port
```

Example

The following command specifies the rise and fall uncertainty numbers for clock abc at port clk.

```
set_attr setup_uncertainty_by_clock { {abc {17 27}} } [find /des*/top -port clk]
```

Related Information

Affects these commands: [report timing](#)

[synthesize](#)

Affected by these attributes: [use_multi_clks_latency_uncertainty_optimize](#) on page 541

[use_multi_clks_latency_uncertainty_report](#) on page 541

Related attributes: (pin) [clock_setup_uncertainty](#) on page 588

 (port) [clock_setup_uncertainty](#) on page 629

 (pin) [setup_uncertainty_by_clock](#) on page 607

source_early_latency_by_clock

```
source_early_latency_by_clock {{clock {min_rise min_fall max_rise  
max_fall}}...}
```

Read-write [port](#) attribute. Specifies the delay between the ideal waveform edges of the specified clocks and the sequential elements in the circuit for early-mode (hold checking) timing analysis.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The attribute value is a Tcl list of Tcl lists for a port. There can be as many Tcl lists as there are clocks propagating to this port. The Tcl list for a clock contains

- The name of the clock
- The minimum (capture) latency of the rise edge of the ideal waveform of the clock
- The minimum (capture) latency of the fall edge of the ideal waveform of the clock
- The maximum (launch) latency of the rise edge of the ideal waveform of the clock
- The maximum (launch) latency of the fall edge of the ideal waveform of the clock

If a single value is specified for a clock, then all four edge latencies are set to that single value. If the specified value for a clock is a Tcl list containing two values, then the two values are interpreted as rise latency and fall latency.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this port, the value of the `clock_source_early_latency` port attribute applies.

The RTL Compiler tool does not use these attribute values because the tool does not perform any hold timing analysis. However, when you execute the `write_sdc` command, the values specified for the `source_early_latency_by_clock` attribute are written out to the SDC constraints file.

- If only one value was specified for a clock, the following SDC command is written:
`set_clock_latency -clock clock uncertainty -source -early port`
- If two values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min uncertainty -source -early port  
set_clock_latency -clock clock -max uncertainty -source -early port
```

Attribute Reference for Encounter RTL Compiler

Constraint—Port Attributes

- If four values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min -rise uncertainty -source -early port  
set_clock_latency -clock clock -min -fall uncertainty -source -early port  
set_clock_latency -clock clock -max -rise uncertainty -source -early port  
set_clock_latency -clock clock -max -fall uncertainty -source -early port
```

Example

The following command specifies four source latency values for each of the following clocks propagating to port clk: clk1, clk2, clk3, and clk4 for hold analysis.

```
set_attr source_early_latency_by_clock { {clk1 {10 20 30 40}} {clk2 {50 60 70 80}}  
{clk3 {90 100 110 120}} {clk4 {130 140 150 160}} } [find /des*/top -port clk]
```

Related Information

Related attributes:	(clock) clock network early latency on page 669
	(pin) clock network early latency on page 580
	(port) clock network early latency on page 621
	(clock) clock source late latency on page 675
	(pin) clock source late latency on page 594
	(port) clock source late latency on page 636
	(pin) source early latency by clock on page 609

source_late_latency_by_clock

```
source_late_latency_by_clock {{clock {min_rise min_fall max_rise max_fall}}...}
```

Read-write port attribute. Specifies the delay between the ideal waveform edges of the specified clocks and the sequential elements in the circuit for late-mode (setup checking) timing analysis.

The total latency, or delay, for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and the actual sequential elements.

The attribute value is a Tcl list of Tcl lists for a port. There can be as many Tcl lists as there are clocks propagating to this port. The Tcl list for a clock contains

- The name of the clock

Attribute Reference for Encounter RTL Compiler

Constraint—Port Attributes

- The minimum (capture) latency of the rise edge of the ideal waveform
- The minimum (capture) latency of the fall edge of the ideal waveform
- The maximum (launch) latency of the rise edge of the ideal waveform
- The maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive values produce looser timing constraints. For maximum (capture) latency edges, positive values produce tighter timing constraints.

If the specified value for a clock is set to a single delay value, then all four edge latencies are set to that single value. If the attribute is set to a Tcl list containing two values, then the two values are interpreted as rise latency and fall latency.

If no value is specified for this attribute, or if no specific value is specified for a clock propagating to this port, the value of the `clock_source_late_latency` port attribute applies.

To take this attribute into account during timing analysis and optimization, set the following root attributes to `true`:

```
set_attribute use_multi_clks_latency_uncertainty_report true /  
set_attribute use_multi_clks_latency_uncertainty_optimize true /
```

When you execute the `write_sdc` command, the values specified for the `source_late_latency_by_clock` attribute are written out to the SDC constraints file.

- If only one value was specified for a clock, the following SDC command is written:

```
set_clock_latency -clock clock uncertainty -source -late port
```

- If two values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock -min uncertainty -source -late port  
set_clock_latency -clock clock -max uncertainty -source -late port
```

- If four values were specified for a clock, the following SDC commands are written:

```
set_clock_latency -clock clock clock -min -rise uncertainty -source -late port  
set_clock_latency -clock clock -min -fall uncertainty -source -late port  
set_clock_latency -clock clock -max -rise uncertainty -source -late port  
set_clock_latency -clock clock -max -fall uncertainty -source -late port
```

Example

The following command sets the capture rise and fall, and launch rise and fall source latency numbers for clock `clk3` to 90, 100, 100 and 120 ps at port `clk` for setup analysis.

```
set_attr source_late_latency_by_clock { {clk3 {90 100 110 120}} } \  
[find /des*/design -port clk]
```

Related Information

Affects these commands: [report_timing](#)

[synthesize](#)

Affected by these attributes: [use_multi_clks_latency_uncertainty_optimize](#) on page 541

[use_multi_clks_latency_uncertainty_report](#) on page 541

Related attributes: [\(clock\) clock_network_late_latency](#) on page 670

[\(pin\) clock_network_late_latency](#) on page 584

[\(port\) clock_network_late_latency](#) on page 625

[\(clock\) clock_source_late_latency](#) on page 675

[\(pin\) clock_source_late_latency](#) on page 594

[\(port\) clock_source_late_latency](#) on page 636

[\(pin\) source_late_latency_by_clock](#) on page 610

timing_case_logic_value

`timing_case_logic_value {no_value | 0 | 1}`

Default: no_value

Read-write port attribute. Forces the port to assume the specified logic value for timing analysis purposes. You can set this attribute on input ports.

Related Information

Combinational Feedback Loops in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.

Affects these commands:	<u>report_timing</u> <u>synthesize</u>
Affected by this attribute:	(port) <u>timing_case_logic_value_by_mode</u> on page 664
Related attributes:	(pin) <u>timing_case_logic_value</u> on page 612 (pin) <u>timing_case_logic_value_by_mode</u> on page 613

timing_case_logic_value_by_mode

```
timing_case_logic_value_by_mode {mode_name_1 case_analysis_value}
[ {mode_name_2 case_analysis_value} ]...
```

Read-write port attribute. Forces the timer to assume the specified logic value for particular timing modes. Specify a Tcl list of modes and corresponding case analysis values.

When the `timing_case_logic_value_by_mode` attribute is set, the `timing_case_logic_value` attribute will be reset to `no_value`. However, when all the values for the `timing_case_logic_value_by_mode` attribute are the same, the `timing_case_logic_value` attribute will be set to that value.

- The following example show that when the `timing_case_logic_value` attribute is set, the `timing_case_logic_value_by_mode` attribute will reflect that value for each mode.

```
rc:/> dc::set_case_analysis -mode a 0 [dc::get_ports in
rc:/> get_attribute timing_case_logic_value_by_mode in
{/designs/top/modes/a 0}
rc:/> get_attribute timing_case_logic_value in
no_value
```

- The following example shows how to set this attribute in modes a and b:

```
rc:/> set_attribute timing_case_logic_value_by_mode {{a 1} {b 0}} \
[find / -port in]
Setting attribute of port 'in': 'timing_case_logic_value_by_mode' =
{/designs/add/modes/b 0} {/designs/add/modes/a 1}
rc:/> get_attr timing_case_logic_value_by_mode n0001D/I1 \
[find / -port in]
{/designs/add/modes/b 0} {/designs/add/modes/a 1}
```

Note: Specifying the `timing_case_logic_value_by_mode` attribute applies the `timing_case_computed_value_by_mode` attribute on the port where it is applied and on the relevant pins of logic downstream.

Example

- The following example shows how to set this attribute in modes a and b:

```
rc:/> set_attribute timing_case_logic_value_by_mode {{a 1} {b 0}}
/designs/add/instances_comb/n0001D/I1
Setting attribute of pin 'I1': 'timing_case_logic_value_by_mode' =
{/designs/add/modes/b 0} {/designs/add/modes/a 1}
rc:/> get_attr timing_case_logic_value_by_mode n0001D/I1
{/designs/add/modes/b 0} {/designs/add/modes/a 1}
```

Attribute Reference for Encounter RTL Compiler

Constraint—Port Attributes

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:	report clocks report qor report timing write encounter write sdc write script
Related commands:	create mode read sdc
Affects this attribute:	(port) timing case logic value on page 663
Related attributes:	disabled arcs by mode on page 565 (pin) external delays by mode on page 969 (port) external delays by mode on page 1007 (design) latch borrow by mode on page 552 (instance) latch max borrow by mode on page 572 (pin) propagated clocks by mode on page 976 (port) propagated clocks by mode on page 1015 (design) slack by mode on page 943 (pin) slack by mode on page 980 (port) slack by mode on page 1018 (cost group) slack by mode on page 1060 (pin) timing case computed value by mode on page 985 (port) timing case computed value by mode on page 1021 instance) timing case disabled arcs on page 953 (instance) timing case disabled arcs by mode on page 954 (pin) timing case logic value by mode on page 613

Subdesign Attributes

Contain information about the subdesigns in the specified design. Subdesigns correspond to Verilog modules or VHDL entities instantiated in the top-level Verilog module or top-level VHDL entity.

- To set a subdesign attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subdesign name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subdesign name]
```

force_wireload

```
force_wireload {auto_select | custom_wireload | none | inherit}
```

Default: auto_select

Read-write subdesign attribute. Forces RTL Compiler to use the specified wire-load model.

auto_select

Automatically selects wire-load models according to the wire-load selection table or default wire-load model in the technology library.

Wire-load models are not used if the `wireload_selection` attribute is set to none.

custom_wireload

Forces RTL Compiler to use the specified custom wire-load model. Specify the hierarchical path to the wire-load model to be used. You can obtain the path using the `find` command.

```
set_attribute force_wireload [find / -wireload "10x10"] \
==> [find / -design -subdesign "add"]
```

inherit

Uses wire-load model of the subdesign or design that instantiates this subdesign.

none

Prevents use of any wire-load models.

To revert to the default behavior, set the `force_wireload` attribute to an empty string:

```
set_attribute force_wireload "" [find / -design -subdesign "add"]
```

Related Information

[Specifying a Wire-load Model in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by these attributes: (design) [force_wireload](#) on page 547
[wireload_selection](#) on page 543

hard_region

hard_region {false | true}

Default: false

Read-write [subdesign](#) attribute. Specifies that the subdesign will be treated as a hard region in the floorplan and preserves pins and subports.

Note: Some place and route tools operate better if your design has no buffers between regions at the top level. To accommodate this, specify hard regions before mapping.

Related Information

Related attribute: (instance) [hard_region](#) on page 567

Clock Attributes

Contain information about clock objects in the specified design.

- To set a `clock` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -clock myclock]
```

- To get a `clock` attribute value, type

```
get_attribute attribute_name [find /des*/design -clock myclock]
```

clock_hold_uncertainty

`clock_hold_uncertainty` *Tcl_list*

Default: 0 0 {R F}

Read-write `clock` attribute. Specifies the uncertainty in the arrival times of capturing edges (in picoseconds) for the clock in early-mode (hold) timing analysis.

If you specify a single value, then both the rising and falling edge are set to that single value. If you specify a Tcl list containing two values, then the two values are interpreted as the rising and falling edge.

Note: RTL Compiler ignores this value in optimization and timing analysis, but can pass it to downstream tools.

Related Information

[Specifying Clock Skew in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:	(pin) <u>clock_hold_uncertainty</u> on page 578
	(port) <u>clock_hold_uncertainty</u> on page 619
	(clock) <u>clock_setup_uncertainty</u> on page 672
	(pin) <u>clock_setup_uncertainty</u> on page 588
	(port) <u>clock_setup_uncertainty</u> on page 629

clock_network_early_latency

`clock_network_early_latency Tcl_list`

Default: 0 0 0 0 {r f R F}

Read-write `clock` attribute. Specifies the delay (in picoseconds) between the ideal waveform edges and the sequential elements in the circuit for early-mode (hold) timing analysis.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and an actual sequential element.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: RTL Compiler ignores this value in optimization and timing analysis, but can pass it to downstream tools.

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:

(pin) [clock_network_early_latency](#) on page 580

(port) [clock_network_early_latency](#) on page 621

(clock) [clock_network_late_latency](#) on page 670

(pin) [clock_network_late_latency](#) on page 584

(port) [clock_network_late_latency](#) on page 625

(clock) [clock_source_early_latency](#) on page 673

- (pin) [clock_source_early_latency](#) on page 590
- (port) [clock_source_early_latency](#) on page 632
- (clock) [clock_source_late_latency](#) on page 675
- (pin) [clock_source_late_latency](#) on page 594
- (port) [clock_source_late_latency](#) on page 636

clock_network_late_latency

`clock_network_late_latency Tcl_list`

Default: 0 0 0 0 {r f R F}

Read-write [clock](#) attribute. Specifies the delay (in picoseconds) between the ideal waveform edges and the sequential elements in the circuit for late-mode (setup checking) timing analysis.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and an actual sequential element.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive latency values produce looser timing constraints. For maximum (capture) latency edges, positive latency values produce tighter timing constraints.

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Example

Both of the following examples set the latency to 150 picoseconds:

```
rc:/> set_attribute clock_network_late_latency 150 clk1
rc:/> set_attribute clock_network_late_latency {150 150} clk1
```

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

- Related attributes:
- (clock) [clock network early latency](#) on page 669
 - (pin) [clock network early latency](#) on page 580
 - (port) [clock network early latency](#) on page 621
 - (pin) [clock network late latency](#) on page 584
 - (port) [clock network late latency](#) on page 625
 - (clock) [clock source early latency](#) on page 673
 - (pin) [clock source early latency](#) on page 590
 - (port) [clock source early latency](#) on page 632
 - (clock) [clock source late latency](#) on page 675
 - (pin) [clock source late latency](#) on page 594
 - (port) [clock source late latency](#) on page 636

clock_setup_uncertainty

`clock_setup_uncertainty Tcl_list`

Default: 0 0

Read-write `clock` attribute. Specifies the uncertainty in the arrival times of capturing edges (in picoseconds) for the respective clock in late-mode (setup) timing analysis. The attribute value is a Tcl list of integers specifying the uncertainty in the rising and falling edges of the ideal clock waveform. Positive values indicate that the clock may arrive earlier in time, which causes timing constraints to be tighter.

If the attribute is set to a single value (instead of a list representing the rise and fall values), then both the rise and fall values are set to the specified single value.

Examples

- The following commands set the rise and fall setup uncertainty for `clk1` to 100 picoseconds:

```
rc:/> set_attribute clock_setup_uncertainty {100 100} clk1
rc:/> set_attribute clock_setup_uncertainty 100 clk1
```
- The following command sets the rise setup uncertainty to 100 picoseconds and the fall setup uncertainty to 50 picoseconds:

```
rc:/> set_attribute clock_setup_uncertainty {100 50} clk1
```

Related Information

[Specifying Clock Skew in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:

(clock) <u>clock_hold_uncertainty</u> on page 668
(pin) <u>clock_hold_uncertainty</u> on page 578
(port) <u>clock_hold_uncertainty</u> on page 619
(pin) <u>setup_uncertainty_by_clock</u> on page 607
(port) <u>clock_setup_uncertainty</u> on page 629

clock_source_early_latency

`clock_source_early_latency Tcl_list`

Default: { 0.0 0.0 0.0 0.0 }

Read-write `clock` attribute. Specifies the delay (in picoseconds) between the ideal waveform edges and the sequential elements in the circuit for early-mode (hold) timing analysis. You can specify a Tcl list of four delay values.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and an actual sequential element.

The attribute value is a Tcl list of four delay values that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Note: RTL Compiler ignores this value in optimization and timing analysis, but can pass it to downstream tools.

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes:

- (clock) [clock_network_early_latency](#) on page 669
- (pin) [clock_network_early_latency](#) on page 580
- (port) [clock_network_early_latency](#) on page 621
- (clock) [clock_network_late_latency](#) on page 670
- (pin) [clock_network_late_latency](#) on page 584

Attribute Reference for Encounter RTL Compiler

Constraint—Clock Attributes

- (port) [clock_network_late_latency](#) on page 625
- (pin) [clock_source_early_latency](#) on page 590
- (port) [clock_source_early_latency](#) on page 632
- (clock) [clock_source_late_latency](#) on page 675
- (pin) [clock_source_late_latency](#) on page 594
- (port) [clock_source_late_latency](#) on page 636

clock_source_late_latency

`clock_source_late_latency Tcl_list`

Default: { 0.0 0.0 0.0 0.0 }

Read-write `clock` attribute. Specifies the delay between the ideal waveform edges and the sequential elements in the circuit for late-mode (setup checking) timing analysis.

The total latency (or delay) for a clock edge is the sum of the “network” and “source” latencies. The source latency is the delay between the ideal clock waveform and the point in the circuit where the clock waveform is applied (such as the clock input port of the design). The network latency is the delay of the clock network between the point where the clock has been defined and an actual sequential element.

The attribute value is a Tcl list of four delay values in picoseconds that represent (in order):

- Minimum (capture) latency of the rise edge of the ideal waveform
- Minimum (capture) latency of the fall edge of the ideal waveform
- Maximum (launch) latency of the rise edge of the ideal waveform
- Maximum (launch) latency of the fall edge of the ideal waveform

RTL Compiler computes timing constraints for a path using the maximum clock latency at the launching clock edge and the minimum clock latency at the capturing clock edge. This produces the tightest possible timing constraints.

Positive values for latency indicate that the edge arrives later in time. For minimum (capture) latency edges, positive latency values produce looser timing constraints. For maximum (capture) latency edges, positive latency values produce tighter timing constraints.

If you specify a single value, all four edge latencies are set to that single value. If you specify a Tcl list containing two values, the two values are interpreted as rise and fall latency and the same value is applied for the minimum and maximum values.

Example

All of the following examples set the latency to 150 picoseconds:

```
rc:/> set_attribute clock_source_late_latency 150 clk1
rc:/> set_attribute clock_source_late_latency {150 150} clk1
rc:/> set_attribute clock_source_late_latency {150 150 150 150} clk1
```

Attribute Reference for Encounter RTL Compiler

Constraint—Clock Attributes

Related Information

[Specifying Clock Latency \(Insertion Delay\) in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

- Related attributes:
- (clock) [clock network early latency](#) on page 669
 - (pin) [clock network early latency](#) on page 580
 - (port) [clock network early latency](#) on page 621
 - (clock) [clock network late latency](#) on page 670
 - (pin) [clock network late latency](#) on page 584
 - (port) [clock network late latency](#) on page 625
 - (clock) [clock source early latency](#) on page 673
 - (pin) [clock source early latency](#) on page 590
 - (port) [clock source early latency](#) on page 632
 - (pin) [clock source late latency](#) on page 594
 - (port) [clock source late latency](#) on page 636

comment

`comment string`

Read-write [clock](#) attribute. Specifies the comment tagged to this clock.

Example

```
rc:/> dc::create_clock -name myclk -period 10 -comment "comment for create clock"
rc:/> get_attribute comment [find /designs/top -clock myclk]
comment for create clock
```

Related Information

Set by one of these commands: [dc::create_clock](#)

[dc::create_generated_clock](#)

Related attribute: (exception) [comment](#) on page 680

inverted_sources

`inverted_sources string`

Read-write `clock` attribute. Specifies a Tcl list of the ports and pins that are inverted sources of the clock waveform.

Related Information

[Combinational Feedback Loops in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attribute: [non_inverted_sources](#) on page 677

latch_max_borrow

`latch_max_borrow {no_value | integer}`

Default: no_value

Read-write `clock` attribute. Specifies the maximum amount of time that can be borrowed, in picoseconds, from the next clock cycle. The specified value must be an integer greater or equal to 0 (decimal values are rounded to the nearest integer) or no_value. This attribute is available on latch cells, clocks, or on a design. If it is set on a latch cell, it replaces the value set on the design. If the `latch_borrow` attribute has already been set, then the `latch_max_borrow` attribute is ignored.

Related Information

Affected by these attributes: (design) [latch_borrow](#) on page 551

(instance) [latch_borrow](#) on page 568

Related attributes: (design) [latch_max_borrow](#) on page 554

(instance) [latch_max_borrow](#) on page 571

(pin) [latch_max_borrow](#) on page 601

non_inverted_sources

`non_inverted_sources Tcl_list`

Read-write `clock` attribute. Specifies a Tcl list of the ports and pins that are sources of the clock waveform.

Related Information

[Combinational Feedback Loops in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attribute: [inverted_sources](#) on page 677

slew

`slew Tcl_list`

Default: 0.0 0.0 0.0 0.0

Read-write `clock` attribute. Specifies the minimum rise, minimum fall, maximum rise, and maximum fall slew values, respectively, in picoseconds. RTL Compiler ignores (does not use) the minimum values but they can be passed to downstream tools. The slew can affect both the delay through the sequential devices and the setup requirements within them.

If you specify different slew values, you must do so as a Tcl list (within braces).

Example

- The following example sets the minimum rise value to be 100, minimum fall value to be 110, maximum rise value to be 110 and maximum fall value to be 120 on `clock1`:

```
rc:/> set_attribute slew {100 110 110 120} \
    /designs/example_design/timing/clock_domains/domain_1/clock1
```

- The following example sets the minimum and maximum rise slew values to be 100 and the minimum and maximum fall slew values to be 110 on `clock1`:

```
rc:/> set_attribute slew {100 110} \
    /designs/example_design/timing/clock_domains/domain_1/clock1
```

- The following example specifies all rise and slew values to be 100 on `clock1`:

```
rc:/> set_attribute slew 100 \
    /designs/example_design/timing/clock_domains/domain_1/clock1
```

Related Information

[Specifying the Clock Transition in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Related attributes: (pin) [slew](#) on page 982
(port) [slew](#) on page 1019

Cost Group Attributes

Contain information about a group of timing paths in the specified design that have a single timing optimization objective.

- To set a `cost_group` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -cost_group group]
```

- To get a `cost_group` attribute value, type

```
get_attribute attribute_name [find /des*/design -cost_group group]
```

weight

`weight integer`

Default: 1

Read-write `cost_group` attribute. Specifies the weight value specified using the `-weight` option of the `define_cost_group` command. You can override this value using the `set_attribute` command.

Related Information

[Cost Group Examples in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Set by this command: [define_cost_group](#)

Exception Attributes

Contain information about the timing exceptions in the specified design. 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`.

- To set an exception attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design/timing -exception name]
```

- To get an exception attribute value, type

```
get_attribute attribute_name [find /des*/design/timing -exception name]
```

comment

comment string

Read-write exception attribute. Specifies the comment tagged to this exception.

Example

```
rc:/> dc::set_min_delay -from d 5 -comment "test comment for set min delay"
/designs/top/timing/exceptions/path_delays/del_4
rc:/> get_attribute comment [find /designs/top -exception del_4]
test comment for set min delay
```

Related Information

Set by one of these commands: [dc::group_path](#)

[dc::set_clock_groups](#)

[dc::set_false_path](#)

[dc::set_max_delay](#)

[dc::set_min_delay](#)

[dc::set_multicycle_path](#)

Related attribute: (clock) [comment](#) on page 676

max

```
max {true | false}
```

Default: true

Read-write exception attribute. Indicates if the exception was created for a MAX delay analysis. This attribute is set when you read in SDC constraints.

Note: RTL Compiler always performs a MAX delay analysis.

user_priority

```
user_priority integer
```

Default: 0

Read-write exception attribute. Specifies the user priority associated with a timing exception.

A timing path can meet the criteria of from-points, through-points, and to-points for a number of timing exceptions. However, only one exception can be applied to the path. Also, only one exception of type `path_group` can be applied to a path.

Note: This attribute is a read-only attribute for `path_adjust` exceptions. It does not make sense to associate a priority with `path_adjust` exceptions because multiple `path_adjust` exceptions can be applied to a single path.

Example

If a timing path satisfies a `path_disable` and a `multi_cycle` exception and also two different `path_group` exceptions, the timing engine selects one of the `path_disable` or `multi_cycle` exceptions and one of the two `path_group` exceptions as being active for that path. RTL Compiler chooses the exception with the higher user priority. In case of equal user priority, RTL Compiler chooses the exception that was last entered.

Related Information

[Setting Path Exception Priorities](#) in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Related attribute: [priority](#) on page 1066

External Delay Attributes

Contains information about the external delay constraints defined in the specified design. These attributes are read-write attributes.

- To set an `external_delay` attribute value, type

```
set_attribute attribute_name attribute_value \
[find /des*/design/timing -external_delay name]
```

- To get an `external_delay` attribute value, type

```
get_attribute attribute_name [find /des*/design/timing -external_delay name]
```

clock_network_latency_included

```
clock_network_latency_included {false | true}
```

Default: false

Read-write `external_delay` attribute. When set to `true`, the `external_delay` delay value (in ps) includes the clock latency values (in ps).

Normally, RTL Compiler constrains paths with both the `external_delay` value (in ps) and the clock network latency values (in ps). If this attribute is set to `true`, RTL Compiler ignores the clock network latency values if an `external_delay` on the path is already adjusting for the clock latency values.

Related Information

Related attributes:

(clock) [clock network early latency](#) on page 669
(pin) [clock network early latency](#) on page 580
(port) [clock network early latency](#) on page 621
(clock) [clock network late latency](#) on page 670
(pin) [clock network late latency](#) on page 584
(port) [clock network late latency](#) on page 625
(clock) [clock source early latency](#) on page 673
(pin) [clock source early latency](#) on page 590
(port) [clock source early latency](#) on page 632
(clock) [clock source late latency](#) on page 675

(pin) [clock_source_late_latency](#) on page 594
(port) [clock_source_late_latency](#) on page 636

clock_source_latency_included

`clock_source_latency_included {false | true}`

Default: false

Read-only [external_delay](#) attribute. When set to true, the `external_delay` delay value (in p.s.) includes the clock source latency values (in p.s.).

Normally, RTL Compiler constrains paths with both the `external_delay` value (in p.s.) and the clock source latency values (in p.s.). If this attribute is set to true, RTL Compiler ignores the clock source latency values if an `external_delay` on the path is already adjusting for the clock latency values.

Related Information

Related attributes:

(clock) [clock_network_early_latency](#) on page 669
(pin) [clock_network_early_latency](#) on page 580
(port) [clock_network_early_latency](#) on page 621
(clock) [clock_network_late_latency](#) on page 670
(pin) [clock_network_late_latency](#) on page 584
(port) [clock_network_late_latency](#) on page 625
(clock) [clock_source_early_latency](#) on page 673
(pin) [clock_source_early_latency](#) on page 590
(port) [clock_source_early_latency](#) on page 632
(clock) [clock_source_late_latency](#) on page 675
(pin) [clock_source_late_latency](#) on page 594
(port) [clock_source_late_latency](#) on page 636

Attribute Reference for Encounter RTL Compiler

Constraint—External Delay Attributes

Elaboration and Synthesis

Root Attributes

- [auto_partition](#) on page 696
- [auto_super_thread](#) on page 696
- [auto_ungroup](#) on page 697
- [bank_based_multibit_inferencing](#) on page 698
- [boundary_optimize_constant_hier_pins](#) on page 699
- [boundary_optimize_equal_opposite_hier_pins](#) on page 700
- [boundary_optimize_feedthrough_hier_pins](#) on page 701
- [boundary_optimize_invert_hier_pins](#) on page 702
- [boundary_optimize_invert_hier_pins_rename_nets](#) on page 703
- [boundary_optimize_invert_hier_pins_renaming_extension](#) on page 704
- [bus_naming_style](#) on page 704
- [comb_seq_merge_message_threshold](#) on page 705
- [constant_prop_through_iso_cell](#) on page 706
- [control_logic_optimization](#) on page 707
- [delete_flops_on_preserved_net](#) on page 707
- [delete_hier_insts_on_preserved_net](#) on page 708
- [delete_unloaded_insts](#) on page 708
- [delete_unloaded_seqs](#) on page 709
- [derive_bussed_pins](#) on page 709
- [dont_use_qbar_seq_pins](#) on page 710

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—List

- [dp_area_mode](#) on page 710
- [dp_csa](#) on page 711
- [dp_postmap_downsize](#) on page 711
- [dp_postmap_upsize](#) on page 712
- [dp_rewriting](#) on page 712
- [dp_sharing](#) on page 713
- [dp_speculation](#) on page 714
- [dp_ungroup_separator](#) on page 715
- [driver_for_unloaded_hier_pins](#) on page 716
- [exact_match_seq_async_ctrls](#) on page 716
- [exact_match_seq_sync_ctrls](#) on page 717
- [force_merge_combos_into_multibit_cells](#) on page 718
- [force_merge_seqs_into_multibit_cells](#) on page 719
- [gen_module_prefix](#) on page 720
- [hdl_append_generic_ports](#) on page 720
- [hdl_array_naming_style](#) on page 722
- [hdl_async_set_reset](#) on page 722
- [hdl_auto_async_set_reset](#) on page 723
- [hdl_auto_exec_sdc_scripts](#) on page 723
- [hdl_auto_sync_set_reset](#) on page 725
- [hdl_bidirectional_assign](#) on page 725
- [hdl_bus_wire_naming_style](#) on page 726
- [hdl_case_mux_threshold](#) on page 727
- [hdl_case_sensitive_instances](#) on page 728
- [hdl_create_label_for_unlabeled_generate](#) on page 729
- [hdl_decimal_parameter_name](#) on page 730
- [hdl_delete_transparent_latch](#) on page 732

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—List

- [hdl_enable_non_default_library_domain_binding](#) on page 732
- [hdl_enable_proc_name](#) on page 732
- [hdl_error_on_blackbox](#) on page 733
- [hdl_error_on_latch](#) on page 733
- [hdl_error_on_logic_abstract](#) on page 734
- [hdl_error_on_nededge](#) on page 735
- [hdl_ff_keep_explicit_feedback](#) on page 735
- [hdl_ff_keep_feedback](#) on page 736
- [hdl_flatten_complex_port](#) on page 737
- [hdl_generate_index_style](#) on page 739
- [hdl_generate_separator](#) on page 742
- [hdl_index_mux_threshold](#) on page 744
- [hdl_infer_unresolved_from_logic_abstract](#) on page 744
- [hdl_instance_array_naming_style](#) on page 745
- [hdl_interface_separator](#) on page 747
- [hdl_latch_keep_feedback](#) on page 748
- [hdl_max_loop_limit](#) on page 749
- [hdl_max_map_to_mux_control_width](#) on page 750
- [hdl_max_recursion_limit](#) on page 750
- [hdl_parameter_naming_style](#) on page 750
- [hdl_preserve_dangling_output_nets](#) on page 753
- [hdl_preserve_supply_nets](#) on page 753
- [hdl_preserve_sync_ctrl_logic](#) on page 754
- [hdl_preserve_unused_flop](#) on page 755
- [hdl_preserve_unused_latch](#) on page 757
- [hdl_preserve_unused_registers](#) on page 759
- [hdl_record_naming_style](#) on page 762

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—List

- [hdl reg naming style](#) on page 762
- [hdl rename cdn flop pins](#) on page 764
- [hdl sv module wrapper](#) on page 765
- [hdl sync set reset](#) on page 765
- [hdl unconnected input port value](#) on page 767
- [hdl undriven output port value](#) on page 771
- [hdl undriven signal value](#) on page 773
- [hdl use block prefix](#) on page 775
- [hdl use cw first](#) on page 776
- [hdl use default parameter values in design name](#) on page 776
- [hdl use default parameter values in name](#) on page 777
- [hdl use for generate prefix](#) on page 778
- [hdl use if generate prefix](#) on page 779
- [hdl use parameterized module by name](#) on page 781
- [hdl use port default value](#) on page 783
- [hdl use techelt first](#) on page 783
- [ignore preserve in tiecell insertion](#) on page 784
- [incr retime](#) on page 784
- [iopt allow tiecell with inversion](#) on page 785
- [iopt enable floating output check](#) on page 786
- [iopt enable parallelization](#) on page 786
- [iopt force constant removal](#) on page 787
- [iopt remap avoided cells](#) on page 787
- [iopt sequential duplication](#) on page 787
- [iopt sequential resynthesis](#) on page 788
- [iopt sequential resynthesis min effort](#) on page 788
- [iopt temp directory](#) on page 789

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—List

- [iopt ultra optimization](#) on page 789
- [lbr seq in out phase opto](#) on page 790
- [map drc first](#) on page 791
- [map latch allow async decomp](#) on page 792
- [map respect rtl clk phase](#) on page 792
- [map to master slave lssd](#) on page 793
- [map to multiple output gates](#) on page 794
- [merge combinational hier instances](#) on page 794
- [merge multibit power area based](#) on page 794
- [minimize uniquify](#) on page 796
- [multibit allow async phase map](#) on page 796
- [multibit allow unused bits](#) on page 798
- [multibit cells from different busses](#) on page 798
- [multibit debug](#) on page 799
- [multibit mapping effort level](#) on page 800
- [multibit predefined allow unused bits](#) on page 801
- [multibit prefix string](#) on page 801
- [multibit preserve inferred instances](#) on page 802
- [multibit preserved net check](#) on page 803
- [multibit seqs instance naming style](#) on page 803
- [multibit seqs members naming style](#) on page 805
- [multibit skip exception check](#) on page 807
- [multibit split string](#) on page 808
- [multibit unused input value](#) on page 809
- [optimize constant 0 flops](#) on page 810
- [optimize constant 1 flops](#) on page 810
- [optimize constant latches](#) on page 810

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—List

- [optimize_merge_flops](#) on page 811
- [optimize_merge_latches](#) on page 811
- [optimize_net_area](#) on page 812
- [optimize_seq_x_to](#) on page 812
- [propagate_constant_from_timing_model](#) on page 812
- [proto_feasible_target](#) on page 813
- [proto_feasible_target_adjust_slack_pct](#) on page 813
- [proto_feasible_target_threshold](#) on page 814
- [proto_feasible_target_threshold_clock_pct](#) on page 814
- [proto_hdl](#) on page 815
- [remove_assigns](#) on page 816
- [retime_async_reset](#) on page 817
- [retime_effort_level](#) on page 817
- [retime_move_mux_loop_with_reg](#) on page 818
- [retime_optimize_reset](#) on page 819
- [retime_reg_naming_suffix](#) on page 819
- [retime_verification_flow](#) on page 820
- [retiming_clocks](#) on page 821
- [stop_at_iopt_state](#) on page 822
- [tns_opto](#) on page 822
- [ultra_global_mapping](#) on page 823
- [uniquify_naming_style](#) on page 824
- [use_multibit_cells](#) on page 826
- [use_multibit_combo_cells](#) on page 827
- [use_multibit_seq_and_tristate_cells](#) on page 828
- [use_nextstate_type_only_to_assign_sync_ctrls](#) on page 829
- [use_scan_seqs_for_non_dft](#) on page 829

- [use_tiehilo_for_const](#) on page 830

Design Attributes

- [control_logic_optimization](#) on page 832
- [delete_unloaded_seqs](#) on page 833
- [dp_csa](#) on page 833
- [dp_rewriting](#) on page 834
- [dp_sharing](#) on page 835
- [dp_speculation](#) on page 836
- [dp_verify_ok](#) on page 836
- [hdl_cw_list](#) on page 837
- [preserve](#) on page 838
- [retime](#) on page 839
- [undesirable_libcells](#) on page 840

Instance Attributes

- [dont_infer_multibit](#) on page 841
- [dont_retime](#) on page 842
- [dont_split_multibit](#) on page 842
- [dont_use_qbar_pin](#) on page 843
- [hdl_proc_name](#) on page 843
- [infer_multibit](#) on page 844
- [inherited_preserve](#) on page 845
- [map_to_multibit_bank_label](#) on page 846
- [map_to_multibit_register](#) on page 847
- [map_to_mux](#) on page 848
- [map_to_register](#) on page 848

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—List

- [merge combinational hier instance](#) on page 849
- [multibit allow async phase map](#) on page 849
- [optimize constant 0 seq](#) on page 852
- [optimize constant 1 seq](#) on page 853
- [optimize merge seq](#) on page 854
- [preserve](#) on page 855
- [propagate constant from timing model](#) on page 856
- [trace retime](#) on page 857
- [undesirable libcells](#) on page 858
- [ungroup ok](#) on page 858
- [unresolved](#) on page 859

hdl arch Attributes

- [blackbox](#) on page 860
- [hdl error on blackbox](#) on page 860
- [hdl error on latch](#) on page 861
- [hdl error on logic abstract](#) on page 861
- [hdl error on negedge](#) on page 862
- [hdl ff keep explicit feedback](#) on page 862
- [hdl ff keep feedback](#) on page 863
- [hdl latch keep feedback](#) on page 863
- [hdl preserve unused flop](#) on page 864
- [hdl preserve unused latch](#) on page 865
- [hdl preserve unused registers](#) on page 865
- [ungroup](#) on page 866

hdl_block Attributes

- [group](#) on page 867

hdl_comp Attributes

- [ungroup](#) on page 869

hdl_impl Attributes

- [ungroup](#) on page 870

hdl_inst Attributes

- [ungroup](#) on page 871

hdl_proc Attributes

- [group](#) on page 872

hdl_subp Attributes

- [map_to_module](#) on page 875
- [map_to_operator](#) on page 875
- [return_port](#) on page 875

Net Attributes

- [preserve](#) on page 876

Pin Attributes

- [boundary_optimize_constant_hier_pins](#) on page 878
- [boundary_optimize_equal_opposite_hier_pins](#) on page 879
- [boundary_optimize_feedthrough_hier_pins](#) on page 881
- [boundary_optimize_hier_pin_invertible](#) on page 882
- [boundary_optimize_invert_hier_pins](#) on page 883

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—List

- [iopt avoid tiecell replacement](#) on page 884
- [lssd master clock](#) on page 885
- [preserve](#) on page 886
- [prune unused logic](#) on page 887

Pgpin Attributes

- [preserve](#) on page 888

Port Attributes

- [iopt avoid tiecell replacement](#) on page 890
- [lssd master clock](#) on page 890

Subdesign Attributes

- [boundary opto](#) on page 892
- [boundary optimize constant hier pins](#) on page 893
- [boundary optimize equal opposite hier pins](#) on page 894
- [boundary optimize feedthrough hier pins](#) on page 896
- [boundary optimize invert hier pins](#) on page 897
- [control logic optimization](#) on page 899
- [delete unloaded insts](#) on page 900
- [delete unloaded seqs](#) on page 901
- [dp csa](#) on page 901
- [dp rewriting](#) on page 902
- [dp sharing](#) on page 903
- [dp speculation](#) on page 904
- [dp verify ok](#) on page 904
- [hdl cw list](#) on page 905
- [minimize uniquify](#) on page 906

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—List

- [multibit allow async phase map](#) on page 907
- [preserve](#) on page 909
- [retime](#) on page 910
- [retime hard region](#) on page 910
- [user sub arch](#) on page 911

Support Attributes

- [boundary optimize hier pin invertible](#) on page 913
- [iopt avoid tiecell replacement](#) on page 914
- [lssd master clock](#) on page 914

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

auto_partition

```
auto_partition {true | false}
```

Default: true

Read-write `root` attribute. Activates automatic internal partitioning of large designs for efficient synthesis. This attribute must be set before synthesis.

Related Information

[Partitioning in Using Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

auto_super_thread

```
auto_super_thread {true | false}
```

Default: true

Read-write `root` attribute. Controls whether super-threading is automatically launched for the `synthesize` command when the tool is running on a multi-processor machine, you have an `RTL_Compiler_Ultra` license, **and** no super-thread servers were explicitly specified through the `super_thread_servers` attribute.

When the `auto_super_thread` attribute is set to `true`, the effect is the same as specifying:

```
set_attribute super_thread_servers {localhost localhost localhost localhost} /
```

Note: This attribute defaults to `false` when

- Running on a single-processor

- Not using the `RTL_Compiler_Ultra` license
- Invoking RTL Compiler in an LSF environment

Related Information

[Super-threading in Using Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

Affected by these attributes: [max_cpus_per_server](#) on page 107
 [super_thread_servers](#) on page 127

Related attributes: [max_super_thread_cache_size](#) on page 107
 [super_thread_cache](#) on page 123

auto_ungroup

`auto_ungroup {both | none}`

Default: both

Read-write [root](#) attribute. Activates automatic ungrouping to improve area and timing during synthesis. This attribute must be specified before synthesis.

none Ungrouping will not be performed.

both Ungrouping will be performed with an emphasis on both optimizing timing and area.

Note: Aggressive ungrouping of user hierarchies happens during

- RTL optimization—`synthesize -to_gen -effort high`
- Technology mapping—`synthesize -to_map -effort high`

Some ungrouping can occur with low or medium effort as well.

Related Information

[Automatic Ungrouping in Using Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

Affects this attribute: (subdesign) [ungroup_ok](#) on page 911

bank_based_multibit_inferencing

`bank_based_multibit_inferencing {false | true}`

Default: `false`

Read-write [root](#) attribute. Enables predefined multibit cell inferencing (MBCI) without limiting the multibit mapping to specific multibit library cells. The tool automatically finds a suitable multibit cell. Since this attribute enables *predefined* multibit cell inferencing, it implies a forced type of mapping and is therefore not QoR-driven.

Related Information

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

Affects this command: [synthesize](#)

Related attribute: (instance) [map_to_multibit_register](#) on page 847

boundary_optimize_constant_hier_pins

`boundary_optimize_constant_hier_pins {true | false}`

Default: true

Read-write `root` attribute. Controls constant propagation through the hierarchical boundary pins.

Note: If the `boundary_opto` attribute on a subdesign is set to `false`, the setting of this attribute is ignored for that subdesign. If the `boundary_optimize_constant_hier_pins` attribute on an instance pin or subdesign is set to `false` or `true` (does not have the `inherited` value), the setting of this root attribute is ignored for that pin or subdesign.

Related Information

[Setting Boundary Optimization in *Using Encounter RTL Compiler*](#)

Affects this command: [synthesize](#)

Affects these attributes:
(pin) [boundary_optimize_constant_hier_pins](#) on page 878
(subdesign) [boundary_optimize_constant_hier_pins](#) on page 893

Affected by this attribute: [boundary_opto](#) on page 892

Related attributes:
(pin) [boundary_optimize_equal_opposite_hier_pins](#) on page 879
(root) [boundary_optimize_equal_opposite_hier_pins](#) on page 700
(subdesign) [boundary_optimize_equal_opposite_hier_pins](#) on page 894

(pin) [boundary_optimize_feedthrough_hier_pins](#) on page 881

(root) [boundary_optimize_feedthrough_hier_pins](#) on page 701

(subdesign) [boundary_optimize_feedthrough_hier_pins](#) on page 896

(pin) [boundary_optimize_invert_hier_pins](#) on page 883

(root) [boundary_optimize_invert_hier_pins](#) on page 702

(subdesign) [boundary_optimize_invert_hier_pins](#) on page 897

boundary_optimize_equal_opposite_hier_pins

boundary_optimize_equal_opposite_hier_pins {true | false}

Default: true

Read-write root attribute. Controls collapsing of equal and opposite hierarchical boundary pins. Two hierarchical boundary pins are considered equal (opposite), if the tool determines that they always have the same (opposite or inverse) logic value.

Note: If the boundary_opto attribute on a subdesign is set to false, the setting of this attribute is ignored for that subdesign. If the boundary_optimize_equal_opposite_hier_pins attribute on an instance pin or subdesign is set to false or true (does not have the inherited value), the setting of this root attribute is ignored for that pin or subdesign.

Related Information

[Setting Boundary Optimization](#) in *Using Encounter RTL Compiler*

Affects this command: [synthesize](#)

Affects these attributes: [\(pin\) boundary_optimize_equal_opposite_hier_pins](#) on page 879

[\(subdesign\) boundary_optimize_equal_opposite_hier_pins](#) on page 894

Affected by this attribute: [boundary_opto](#) on page 892

Related attributes: [\(pin\) boundary_optimize_constant_hier_pins](#) on page 878

[\(root\) boundary_optimize_constant_hier_pins](#) on page 699

[\(subdesign\) boundary_optimize_constant_hier_pins](#) on page 893

[\(pin\) boundary_optimize_feedthrough_hier_pins](#) on page 881

[\(root\) boundary_optimize_feedthrough_hier_pins](#) on page 701

[\(subdesign\) boundary_optimize_feedthrough_hier_pins](#) on page 896

[\(pin\) boundary_optimize_invert_hier_pins](#) on page 883

[\(root\) boundary_optimize_invert_hier_pins](#) on page 702

[\(subdesign\) boundary_optimize_invert_hier_pins](#) on page 897

boundary_optimize_feedthrough_hier_pins

`boundary_optimize_feedthrough_hier_pins {true | false}`

Default: true

Read-write root attribute. Controls optimization of feedthrough pins. Hierarchical boundary pins are feedthrough pins, if output pins have always the same (or inverted) logic value as an input pin. Such feedthrough pins can be routed around the subdesign and no connections or logic is needed inside the subdesign for these pins. To disable this type of boundary optimization, set the attribute to `false`.

Note: If the `boundary_opto` attribute on a subdesign is set to `false`, the setting of this attribute is ignored for that subdesign. If the `boundary_optimize_feedthrough_hier_pins` attribute on an instance pin or subdesign is set to `false` or `true` (does not have the inherited value), the setting of this root attribute is ignored for that pin or subdesign.

Related Information

[Setting Boundary Optimization in Using Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

Affects these attributes: (pin) [boundary_optimize_feedthrough_hier_pins](#) on page 881

 (subdesign) [boundary_optimize_feedthrough_hier_pins](#) on page 896

Affected by this attribute: [boundary_opto](#) on page 892

Related attributes: (pin) [boundary_optimize_constant_hier_pins](#) on page 878

 (root) [boundary_optimize_constant_hier_pins](#) on page 699

 (subdesign) [boundary_optimize_constant_hier_pins](#) on page 893

 (pin) [boundary_optimize_equal_opposite_hier_pins](#) on page 879

 (root) [boundary_optimize_equal_opposite_hier_pins](#) on page 700

 (subdesign) [boundary_optimize_equal_opposite_hier_pins](#) on page 894

 (pin) [boundary_optimize_invert_hier_pins](#) on page 883

(root) [boundary_optimize_invert_hier_pins](#) on page 702

(subdesign) [boundary_optimize_invert_hier_pins](#) on
page 897

boundary_optimize_invert_hier_pins

boundary_optimize_invert_hier_pins {false | true}

Default: false

Read-write [root](#) attribute. Controls hierarchical boundary pin inversion. By default, hierarchical boundary pin inversion is disabled. If set to `true`, the boundary pin inversion is controlled by the [boundary_opto](#) subdesign attributes.

Note: If the [boundary_opto](#) attribute on a subdesign is set to `false`, the setting of this attribute is ignored for that subdesign. If the [boundary_optimize_invert_hier_pins](#) attribute on an instance pin or subdesign is set to `false` or `true` (does not have the inherited value), the setting of this root attribute is ignored for that pin or subdesign.

Note: If this attribute is set to `true`, the `write_do_lec` command adds the following LEC command to the dofile:

```
SET Naming Rule _BAR -Inverted_pin_extension
```

This LEC command specifies the string that is appended to pin names and net names by RTL Compiler when hierarchical boundary pins are inverted during synthesis. To modify this string use the [boundary_optimize_invert_hier_pins_renaming_extension](#) root attribute. In the command above, the `_BAR` value corresponds to the default value of this attribute.

Related Information

[Setting Boundary Optimization in Using Encounter RTL Compiler](#)

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

Affects these commands: [synthesize](#)

[write_do_lec](#)

Affects these attributes: (pin) [boundary_optimize_invert_hier_pins](#) on page 883

(subdesign) [boundary_optimize_invert_hier_pins](#) on
page 897

Affected by this attribute: [boundary_opto](#) on page 892

Related attributes:

- (pin) [boundary_optimize_constant_hier_pins](#) on page 878
- (root) [boundary_optimize_constant_hier_pins](#) on page 699
- (subdesign) [boundary_optimize_constant_hier_pins](#) on page 893
- (pin) [boundary_optimize_equal_opposite_hier_pins](#) on page 879
- (root) [boundary_optimize_equal_opposite_hier_pins](#) on page 700
- (subdesign) [boundary_optimize_equal_opposite_hier_pins](#) on page 894
- (pin) [boundary_optimize_feedthrough_hier_pins](#) on page 881
- (root) [boundary_optimize_feedthrough_hier_pins](#) on page 701
- (subdesign) [boundary_optimize_feedthrough_hier_pins](#) on page 896
- [boundary_optimize_invert_hier_pins_rename_nets](#) on page 703
- [boundary_optimize_invert_hier_pins_renaming_extension](#) on page 704

[boundary_optimize_invert_hier_pins_rename_nets](#)

`boundary_optimize_invert_hier_pins_rename_nets {true | false}`

Default: true

Read-write root attribute. Controls renaming of nets driven by the inverted hierarchical boundary pins. By default, the nets will be renamed.

Related Information

Affects this command: [synthesize](#)

Affected by this attribute: [boundary_optimize_invert_hier_pins](#) on page 702

Affects this attribute: [boundary_optimize_invert_hier_pins_renaming_extension](#) on page 704

boundary_optimize_invert_hier_pins_renaming_extension

`boundary_optimize_invert_hier_pins_renaming_extension string`

Default: _BAR

Read-write root attribute. Specifies the string to be appended to pin names and net names when hierarchical boundary pins are inverted during synthesis.

Note: If you specify an empty string, you implicitly disable renaming for pins and nets. However this is not recommended, because the formal verification tools need the naming extension to identify inverted boundary pins.

Note: If the `boundary_optimize_invert_hier_pins` root attribute is set to `true`, the `write_do_lec` command adds the following LEC command to the dofile:

```
SET Naming Rule _BAR -Inverted_pin_extension
```

This LEC command specifies the string that is appended to pin names and net names by RTL Compiler when hierarchical boundary pins are inverted during synthesis. To modify this string use the `boundary_optimize_invert_hier_pins_renaming_extension` root attribute. In the command above, the `_BAR` value corresponds to the default value of this attribute.

Related Information

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

Affects these commands: [synthesize](#)

[write do lec](#)

Affected by these attributes: [boundary_optimize_invert_hier_pins](#) on page 702

[boundary_optimize_invert_hier_pins_rename_nets](#) on page 703

bus_naming_style

`bus_naming_style string`

Default: %s[%d]

Read-write root attribute. Specifies the naming convention for bussed ports and nets in the design. The new naming convention will only be available within RTL Compiler. When the netlist is written out, the naming convention will revert to the default `%s[%d]` format.

Example

The following example specifies that the individual bits on bussed ports should take the form of %s_%d instead of the %s[%d] default:

```
rc:/> set_attribute bus_naming_style %s_%d
...
rc:/> synthesize -to_mapped
rc:/designs/kitkat/ports_in> ls
./      in1_0      in1_1      in1_2      in1_3
```

Related Information

Affects this command: [synthesize](#)

comb_seq_merge_message_threshold

`comb_seq_merge_message_threshold integer`

Default: 10

Read-write root attribute. Enables the printing of detailed messages when hierarchical instances with more cells than the specified threshold value are merged.

Example

```
rc:/> set_attr comb_seq_merge_message_threshold 2 /
Setting attribute of root '/': 'comb_seq_merge_message_threshold' = 2
Info   : Combinational hierarchical blocks with identical inputs have been merged.
[GLO-40]
      : Instances 'sub0' and 'sub1' in 'test' have been merged.
      : This optimization usually reduces design area. To prevent merging of
combinational hierarchical blocks, set the 'merge_combinational_hier_instances'
root attribute to 'false' or the 'merge_combinational_hier_instance' instance
attribute to 'false'.
```

Related Information

Affects this command: [synthesize](#)

Affect by this attribute: [merge_combinational_hier_instances](#) on page 794

constant_prop_through_iso_cell

```
constant_prop_through_iso_cell {true | false}
```

Default: true

Read-write root attribute. Enables the propagation of constants through isolation cells:

- If the constant value at the input and the isolation output type are the same, the constant is propagated and the isolation cell can be removed.
- If the constant value at the input and the isolation output type are different, constant propagation will **not** happen through the isolation cell, and the isolation cell remains.

If you set this attribute to `false`, constant propagation stops at the inputs of isolation cells.



This attribute has only an effect if the following flow is used:

```
set_attribute constant_prop_through_iso_cell true /  
...  
commit_power_intent  
synthesize -to_mapped -no_incremental  
...  
synthesize -incremental
```

Only when you set this attribute before committing the CPF file, will the tool propagate constants.

Related Information

Affects this command: [synthesize -incremental](#)

Affected by this command: [read_power_intent](#)

control_logic_optimization

```
control_logic_optimization {basic | advanced | none}
```

Default: basic

Read-write root attribute. Controls the optimization of control logic (described in the RTL through conditional constructs like case statements, if-then-else statements, conditional selects, and so on) during generic synthesis. You can specify any of the following values:

advanced	Applies advanced level optimization
basic	Applies basic optimization.
none	Turns off optimization.

Note: All control optimization transformations are verifiable. The advanced transformations might result in better QoR but can also increase the runtime. For best results, set this root attribute before you elaborate the design.

Related Information

Affects this command:	synthesize -to_generic
Related attributes:	(design) control_logic_optimization on page 832
	(subdesign) control_logic_optimization on page 899

delete_flops_on_preserved_net

```
delete_flops_on_preserved_net {true | false}
```

Default: true

Read-write root attribute. Controls the deletion of load flip-flops on a preserved net. By default, the flops can be deleted.

Set this attribute to `false` to keep load flip-flops even if they do not drive any primary output, can be replaced with a constant, or can be deleted in some other optimization.

Related Information

Affects this command:	synthesize
-----------------------	----------------------------

delete_hier_insts_on_preserved_net

```
delete_hier_insts_on_preserved_net {true | false}
```

Default: true

Read-write root attribute. Controls the deletion of empty hierarchical instances driven by a preserved net.

Set this attribute to `false` to keep the empty modules on a preserved net, even if they do not drive any primary output.

Related Information

Affects this command: [synthesize](#)

delete_unloaded_insts

```
delete_unloaded_insts {true | false}
```

Default: true

Read-write root attribute. Controls the deletion of unloaded hierarchical instances.

By default, RTL Compiler removes logic if none of the outputs are connected. Set this attribute to `false` if you want to preserve any pre-existing hierarchical instances. If you just want to maintain mapped and unmapped sequential instances, set the `delete_unloaded_seqs` attribute to `false`. Unmapped (Boolean) non-hierarchical instances cannot be rescued if they are unloaded.

Related Information

Affects this command: [synthesize](#)

Affects this attribute: (subdesign) [delete_unloaded_insts](#) on page 900

Related attributes: (root) [delete_unloaded_seqs](#) on page 709

 (subdesign) [delete_unloaded_seqs](#) on page 901

delete_unloaded_seqs

```
delete_unloaded_seqs {true | false}
```

Default: true

Read-write root attribute. Controls the deletion of unloaded sequential instances.

By default RTL Compiler removes flip-flops if none of the outputs are connected. To prevent this, set the `delete_unloaded_seqs` attribute to `false`.

Note: This attribute only affects the `synthesize` command, while the `hdl_preserve_unused_registers` attribute only affects the `elaborate` command.

Related Information

Affects this command: [synthesize](#)

Affects this attribute: (design) [delete_unloaded_seqs](#) on page 833

Related attributes: (root) [delete_unloaded_insts](#) on page 708

 (subdesign) [delete_unloaded_insts](#) on page 900

 (subdesign) [delete_unloaded_seqs](#) on page 901

[hdl_preserve_unused_registers](#) on page 759

derive_bussed_pins

```
derive_bussed_pins {true | false}
```

Default: true

Read-write root attribute. Merges all indexed pins into a bus.

Example

If the `derive_bussed_pins` is set to `true`, the following pins will be merged into a three bit bus named A:

```
A[0]  
A[1]  
A[2]
```

dont_use_qbar_seq_pins

`dont_use_qbar_seq_pins {false | true}`

Default: false

Read-write root attribute. Controls the use of the Qbar output pins of sequential libcells during optimization if other possibilities exist. By default, these pins can be used.

Related Information

Affects this command: [synthesize](#)

Related attribute: (instance) [dont_use_qbar_pin](#) on page 843

dp_area_mode

`dp_area_mode {false | true}`

Default: false

Read-write root attribute. When set to true, enables datapath optimizations that focus on improving area results. This attribute achieves area gain by performing the following:

- Full adder cell mapping
- Conservative Carry-save adder (CSA) transformations
- Sharing before Carry-save adder (CSA) transformations

Note: The sharing operation is only performed when the `synthesize -to_generic -effort high` command is used.

Related Information

[Improving Quality of Results \(QoR\) in Datapath Synthesis in Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

Affects this attribute: (design) [preserve](#) on page 838

dp_csa

dp_csa {basic | none}

Default: basic

Read-write root attribute. Controls the carry-save adder (CSA) transformations in datapath synthesis.

By default, CSA transformations are performed when synthesis is performed with the `-effort` option set to high.

basic	Applies basic transformation.
none	Turns off transformation.

Related Information

Controlling CSA Transformations in *Datapath Synthesis in Encounter RTL Compiler*

Affects this command: **synthesize -to_generic**

Related attributes: (design) dp csa on page 833

(subdesign) dp csa on page 901

dp_postmap_downsize

```
dp postmap downsize {false | true}
```

Default: false

Read-write root attribute. When set to true, performs architecture downsizing after mapping.

Using this attribute accurately downsizes a datapath component without degrading timing. However, potentially this can increase run-time and impact verifiability.

Note: Using this attribute is only effective during incremental optimization.

Example

```
rc:/> set attribute dp postmap downsize true /
```

Related Information

[Improving Quality of Results \(QoR\) in Datapath Synthesis in Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

dp_postmap_upsize

dp_postmap_upsize {false | true}

Default: false

Read-write root attribute. When set to true, allows upsizing of critical datapath components after mapping.

Using this attribute improves timing but potentially can increase run-time and impact verifiability.

Note: Using this attribute is only effective during incremental optimization.

Example

```
rc:/> set_attribute dp_postmap_upsize true /
```

Related Information

[Improving Quality of Results \(QoR\) in Datapath Synthesis in Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

dp_rewriting

dp_rewriting {basic | advanced | none}

Default: basic

Read-write root attribute. Controls how the datapath rewriting optimization is applied during `synthesize -to_generic -effort high` at a global level.

advanced	Applies advanced optimization
basic	Applies basic optimization.
none	Turns off optimization.

Note: All basic transformations are verifiable. The advanced transformations might result in better QoR but not all of them are verifiable.

Related Information

Affects this command: [synthesize -to_generic -effort high](#)

Related attributes: (design) [dp_rewriting](#) on page 834

 (subdesign) [dp_rewriting](#) on page 902

dp_sharing

`dp_sharing {advanced | basic | none}`

Default: advanced

Read-write [root](#) attribute. Controls how resource sharing in datapath synthesis is applied at a global level.

By default, sharing transformations are performed when synthesis is performed with the `-effort` option set to `high`.

advanced Applies advanced optimization.

basic Applies basic optimization.

none Turns off optimization.

Example

The following command turns off RTL sharing transformations:

```
rc:/> set_attribute dp_sharing none /
```

Related Information

[Controlling Sharing Transformations in Datapath Synthesis in Encounter RTL Compiler](#)

Affects this command: [synthesize -to_generic -effort high](#)

Related attributes: (design) [dp_sharing](#) on page 835

 (subdesign) [dp_sharing](#) on page 903

dp_speculation

dp_speculation {none | basic}

Default: none

Read-write root attribute. Controls RTL speculation (unsharing) transformations in datapath synthesis.

By default, speculation transformations are performed when synthesis is performed with the `-effort` option set to high.

basic	Applies basic transformation.
none	Turns off transformation.

Related Information

Controlling Speculation Transformations in Datapath Synthesis in Encounter RTL Compiler

Affects this command: [synthesize -to_generic](#)

Related attributes: (design) dp_speculation on page 836

(subdesign) dp_speculation on page 904

dp_ungroup_separator

dp_ungroup_separator *string*

Default: :

Read-write root attribute. Specifies the string used to separate the hierarchical names of arithmetic and subprogram instances that are ungrouped (flattened) during elaboration.

Example

Using the default value, you may hierarchical instance names that include the : character:

```
rc:/> find / -inst decode_60_5*
/designs/test_11/instances_hier/decode_60_5:lte_48_20_I1
/designs/test_11/instances_hier/decode_60_5:mux_exit_47_48_20
/designs/test_11/instances_hier/decode_60_5:mux_result_48_20
/designs/test_11/instances_hier/decode_60_5:mux_result_48_4
/designs/test_11/instances_hier/decode_60_5:mux_result_47_7
```

To use the underscore character as separator, change the dp_ungroup_separator attribute prior to elaborating the design

```
set_attribute dp_ungroup_separator _ /
elaborate
```

This will have the following effect on the instance names:

```
rc:/> find / -inst decode_60_5*
/designs/test_11/instances_hier/decode_60_5_lte_48_20_I1
/designs/test_11/instances_hier/decode_60_5_mux_exit_47_48_20
/designs/test_11/instances_hier/decode_60_5_mux_result_48_20
/designs/test_11/instances_hier/decode_60_5_mux_result_48_4
/designs/test_11/instances_hier/decode_60_5_mux_result_47_7
```

Related Information

Affects this command: [elaborate](#)

driver_for_unloaded_hier_pins

`driver_for_unloaded_hier_pins {z | 0}`

Default: z

Read-write root attribute. Controls how unloaded subdesign ports are handled during incremental optimization. You can specify to connect these ports to constant 0, or you can leave them unconnected (default) to minimize the number of assigns in the netlist.

Note: This attribute does not apply to clock-gating hierarchies.

Related Information

Affects this command: [synthesize -incremental](#)

exact_match_seq_async_ctrls

`exact_match_seq_async_ctrls {false | true}`

Default: false

Read-write root attribute. If set to true, RTL Compiler tries to avoid tying off the asynchronous control inputs of a flop to constants during technology mapping.

Note: You must set this attribute to true before you load your library.

Related Information

[Specifying Set and Reset Synthesis Pragmas in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [synthesize -to_mapped](#)

Related attribute [exact_match_seq_sync_ctrls](#) on page 717

exact_match_seq_sync_ctrls

`exact_match_seq_sync_ctrls {false | true}`

Default: false

Read-write root attribute. If set to true, RTL Compiler will try to force map the synchronous flops inferred from RTL to complex library flip-flops with corresponding synchronous inputs regardless of QoR. If the library does not have the appropriate synchronous control libcells, simple flip-flops and combinational logic will be used.

You must set this attribute to true before you load your library.

Note: This attribute may have a positive impact on clock gating coverage due to the explicit enable

Related Information

Specifying Set and Reset Synthesis Pragmas in HDL Modeling in Encounter RTL Compiler

Controlling Clock Gate Enable Modelling for Synchronous Reset in Low Power in Encounter RTL Compiler

Affects this command: [synthesize -to_mapped](#)

Related attribute [exact_match_seq_async_ctrls](#) on page 716

force_merge_combos_into_multibit_cells

```
force_merge_combos_into_multibit_cells {false | true}
```

Default: false

Read-write root attribute. Merges single-bit combinational instances into an appropriate multibit combinational instance independent of the impact on the QoR. When enabled, multibit cell inferencing will occur even if it degrades the delay, power, or area QOR of the design. This is useful for increasing multibit coverage but might negatively impact the QoR.

Examples

Assume two single bit instances g1 and 2 can be merged into a multibit instance. The library has three suitable multibit cells: 2g1x1, 2g1x2, and 2g1x3.

- Consider only `force_merge_combos_into_multibit_cells` enabled:

```
set_attribute merge_multibit_power_area_based false /  
set_attribute force_merge_combos_into_multibit_cells true /
```

In this case, the tool computes the total cost with timing for each of the possible multibit libcell candidates. The best candidate for multibit cell merging will be chosen irrespective of whether it improves or degrades the delay QoR of the design.

- Consider both `force_merge_combos_into_multibit_cells` and `merge_multibit_power_area_based` are enabled:

```
set_attribute merge_multibit_power_area_based true /  
set_attribute force_merge_combos_into_multibit_cells true /
```

In this case, the tool computes the power and area costs for each of the possible multibit libcell candidates. The best candidate for multibit cell merging will be chosen irrespective of whether it improves the delay QoR of the design.

Related Information

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

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [use_multibit_cells](#) on page 826

Related attributes: [dont_infer_multibit](#) on page 841

[force_merge_seqs_into_multibit_cells](#) on page 719

[infer_multibit](#) on page 844

[map_to_multibit_register](#) on page 847

[merge_multibit_power_area_based](#) on page 794

force_merge_seqs_into_multibit_cells

```
force_merge_seqs_into_multibit_cells {false | true}
```

Default: false

Read-write [root](#) attribute. Merges single-bit sequential instances into an appropriate multibit sequential instance independent of the impact on the QoR. When enabled, multibit cell inferencing will occur even if it degrades the delay, power, or area QOR of the design. This is useful for increasing multibit coverage but might negatively impact the QoR.

Examples

Assume two single bit instances `reg1` and `reg2` can be merged into a multibit instance. The library has three suitable multibit cells: `dual1`, `dual2`, and `dual3`.

- Consider only `force_merge_seqs_into_multibit_cells` enabled:

```
set_attribute merge_multibit_power_area_based false /  
set_attribute force_merge_seqs_into_multibit_cells true /
```

In this case, the tool computes the total cost with timing for each of the possible multibit libcell candidates. The best candidate for multibit cell merging will be chosen irrespective of whether it improves or degrades the delay QoR of the design.

- Consider both `force_merge_seqs_into_multibit_cells` and `merge_multibit_power_area_based` are enabled:

```
set_attribute merge_multibit_power_area_based true /  
set_attribute force_merge_seqs_into_multibit_cells true /
```

In this case, the tool computes the power and area costs for each of the possible multibit libcell candidates. The best candidate for multibit cell merging will be chosen irrespective of whether it improves the delay QoR of the design.

Related Information

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

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [use_multibit_cells](#) on page 826

Related attributes: [dont_infer_multibit](#) on page 841

[force_merge_combos_into_multibit_cells](#) on page 718

[infer_multibit](#) on page 844

[map_to_multibit_register](#) on page 847

[merge_multibit_power_area_based](#) on page 794

gen_module_prefix

gen_module_prefix *string*

Read-write root attribute. Specifies the prefix to be used for internally generated module names (arithmetic, logic, register-file modules, and so on).

```
set_attribute gen_module_prefix RC_DP_ /
```

Note: This attribute is supported only in the RTL flow.

Related Information

Affects this command: [read_hdl](#)

hdl_append_generic_ports

hdl_append_generic_ports {true | false}

Default: true

Read-write root attribute. When set to false, prevents that generic port information is appended to the module name. By default, RTL Compiler changes the name of the module by appending the interface name and modport name to the original module name.

Example

Consider the following RTL:

```
interface nand_intf (
    input wire [1:0] a,
    input wire [1:0] b,
    output wire [1:0] y );
modport FOO (
    input a, b,
    output y);
endinterface

// Top module
module top;
nand_intf intf_inst ();
test1 t1(
    intf_inst.FOO
);
endmodule

// Sub Module
module test1(nand_intf.FOO intf);
assign intf.y = intf.a & intf.b;
endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

By default, the generated netlist after elaboration looks like:

```
module test1_intf_nand_intf_FOO(intf_a, intf_b, intf_y);
    input [1:0] intf_a, intf_b;
    output [1:0] intf_y;
    wire [1:0] intf_a, intf_b;
    wire [1:0] intf_y;
    and g1 (intf_y[0], intf_a[0], intf_b[0]);
    and g2 (intf_y[1], intf_a[1], intf_b[1]);
endmodule

module top;
    wire [1:0] intf_inst_a;
    wire [1:0] intf_inst_b;
    wire [1:0] intf_inst_y;
    test1_intf_nand_intf_FOO t1(intf_inst_a, intf_inst_b, intf_inst_y);
endmodule
```

When you set the `hdl_append_generic_ports` to `false` before reading in the RTL file, the generated netlist after elaboration looks like:

```
module test1(intf_a, intf_b, intf_y);
    input [1:0] intf_a, intf_b;
    output [1:0] intf_y;wire [1:0] intf_a, intf_b;
    wire [1:0] intf_y;
    and g1 (intf_y[0], intf_a[0], intf_b[0]);
    and g2 (intf_y[1], intf_a[1], intf_b[1]);
endmodule

module top;
    wire [1:0] intf_inst_a;
    wire [1:0] intf_inst_b;
    wire [1:0] intf_inst_y;
    test1 t1(intf_inst_a, intf_inst_b, intf_inst_y);
endmodule
```

Related Information

Affects this command: [elaborate](#)

hdl_array_naming_style

`hdl_array_naming_style string`

Default: `%s\[%d\]`

Read-write root attribute. Specifies the format used to name individual bits of array variables in the RTL. `%s` is the variable name and `%d` is the individual bit. Set this attribute before using the `elaborate` command.

Note: This attribute is supported only in the RTL flow.

Related Information

Naming Individual Bits of Array and Record Ports and Registers in *Using Encounter RTL Compiler*

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_async_set_reset

`hdl_async_set_reset Tcl_list`

Read-write root attribute. Specifies that RTL Compiler implement the listed signals using asynchronous set and reset pins on a latch if that logic controls an asynchronous assignment.

Note: This attribute is only supported in the RTL flow.

Note: This attribute does not affect the `synthesize` command, which may implement the set and reset logic using latch data pins.

Example

```
rc:/ set_attr hdl_async_set_reset {r1 r2} /
```

This command has the same effect as using the `async_set_reset` pragma in the RTL. For example:

```
// cadence async_set_reset "r1 r2".
```

For more information, see the RTL example and the corresponding schematic for `hdl_async_set_reset` in the HDL-Related Attributes section of *HDL Modeling in Encounter RTL Compiler*.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_auto_async_set_reset

`hdl_auto_async_set_reset {false | true}`

Default: `false`

Read-write [root](#) attribute. When set to `true`, specifies that RTL Compiler implement logic using asynchronous set and reset pins on a latch if that logic controls an asynchronous assignment of a constant 0 or a constant 1.

Note: This attribute is only supported in the RTL flow.

Note: This attribute does not affect the `synthesize` or `techmap` commands, which may implement the set and reset logic using latch data pins.

Example

The following command implements the `reset` signal in the RTL using a latch asynchronous reset pin. See the example RTL and the corresponding schematic in [HDL-Related Attributes of HDL Modeling in Encounter RTL Compiler](#).

`rc:/ set_attribute hdl_auto_async_set_reset true`

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_auto_exec_sdc_scripts

`hdl_auto_exec_sdc_scripts {true | false}`

Default: `true`

Read-write [root](#) attribute. When set to `true`, SDC scripts found in the RTL input will be automatically run during elaboration. If this attribute is set to `false`, then SDC scripts will not be automatically started and you will need to use the [exec_embedded_script](#) command to apply SDC scripts onto the design when elaborating the netlist.

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

The embedded script can have any SDC command that is supported by the `read_sdc` command.

Note: This attribute is supported only in the RTL flow.

Example

An embedded script in the RTL code is a sequence of comments between the `script_begin` pragma and the `script_end` pragma.

```
module my_xor (y, a, b);
    parameter w = 8;
    input [w-1:0] a, b;
    output [w-1:0] y;
    assign y = a ^ b;
    // synopsys dc_script_begin
    // create_clock -name vCLK -period 1 -waveform {0 0.5}
    // set_input_delay 0.25 -clock [get_clocks {vCLK}] a*
    // set_input_delay 0.25 -clock [get_clocks {vCLK}] b*
    // set_output_delay 0.25 -clock [get_clocks {vCLK}] y*
    // synopsys dc_script_end
endmodule

module test (y, a, b);
    parameter w = 8;
    input [w-1:0] a, b;
    output [w-1:0] y;
    my_xor u1 (y, a, b);
endmodule
```

The pragma keyword of the `script_begin` pragma is controlled by the `script_begin` attribute.

The pragma keyword of the `script_end` pragma is controlled by the `script_end` attribute.

Related Information

Affects these commands:	elaborate exec_embedded_script read_hdl
Affected by these attributes:	script_begin on page 278 script_end on page 278

hdl_auto_sync_set_reset

```
hdl_auto_sync_set_reset {true | false}
```

Default: true

Read-write root attribute. When set to true, specifies that RTL Compiler implement logic using synchronous set and reset pins on a flip-flop if that logic controls a synchronous assignment of a constant 0 or a constant 1.

This attribute does not affect the `synthesize` command, which may implement the set and reset logic using flip-flop data pins.

Note: This attribute is supported only in the RTL flow.

Example

The following command implements the `reset` signal shown in the RTL using a flip-flop synchronous reset pin. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

```
rc:/ set_attribute hdl_auto_sync_set_reset true
```

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_bidirectional_assign

```
hdl_bidirectional_assign {true | false}
```

Default: true

Read-write root attribute. When set to true, specifies that RTL Compiler must interpret Verilog continuous assign statements, where the right-hand-side is undriven, as being bidirectional. In this case, RTL Compiler matches the behavior of the Encounter® Conformal® Logical Equivalence Checking tool. To prevent bidirectional assignments, set this attribute to false. In the latter case, RTL Compiler matches the behavior of the Verilog Language Reference and the Incisive Simulator.

Note: The attribute does not affect VHDL designs.

Related Information

Affects this command: [elaborate](#)

hdl_bus_wire_naming_style

hdl_bus_wire_naming_style *string*

Default: %s[%d]

Read-write root attribute. Specifies the format used to name individual bits of bus wires. %s is the variable name and %d is the individual bit. Set this attribute before using the elaborate command.

Example

Consider the following RTL code.

```
module test1(clk,d,q);
    input clk;
    input [0:3] d;
    output [0:3] q;
    reg [0:3] q, tmp;

    always @ (posedge clk) begin
        tmp = d;
    end
    always @ (posedge clk) begin
        q = tmp;
    end
endmodule
```

Assume the following script:

```
set_attr library tutorial.lbr /
set_attr hdl_bus_wire_naming_style %s_%d /
read_hdl test.v
elaborate
```

After elaboration, the netlist will look like:

```
module test1(clk, d, q);
    input clk;
    input [0:3] d;
    output [0:3] q;
    wire clk;
    wire [0:3] d;
    wire [0:3] q;
    wire tmp_0, tmp_1, tmp_2, tmp_3;
    CDN_flop \tmp_reg[3] (.clk (clk), .d (d[3]), .sena (1'b1), .aclr
        (1'b0), .apre (1'b0), .srl (1'b0), .srd (1'b0), .q (tmp_3));
    ...
    CDN_flop \tmp_reg[0] (.clk (clk), .d (d[0]), .sena (1'b1), .aclr
        (1'b0), .apre (1'b0), .srl (1'b0), .srd (1'b0), .q (tmp_0));
    CDN_flop \q_reg[3] (.clk (clk), .d (tmp_3), .sena (1'b1), .aclr
        (1'b0), .apre (1'b0), .srl (1'b0), .srd (1'b0), .q (q[3]));
    ...
    CDN_flop \q_reg[0] (.clk (clk), .d (tmp_0), .sena (1'b1), .aclr
        (1'b0), .apre (1'b0), .srl (1'b0), .srd (1'b0), .q (q[0]));
endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

```
`ifdef RC_CDN_GENERIC_GATE
`else
module CDN_flop(clk, d, sena, aclr, apre, srl, srd, q);
...
endmodule
```

Note: The attribute setting only affected vector wire tmp. Ports clk, d, q are not affected by this setting. Also sequential elements names tmp_reg[*] and q_reg[*] are not affected by this setting.

Related Information

Affects this command: [elaborate](#)

hdl_case_mux_threshold

hdl_case_mux_threshold integer

Default: 2

Read-write root attribute. Determines whether a binary multiplexer (encoded select inputs) or a one-hot multiplexer (decoded select inputs) is generated to implement signals assigned within case, if-then-else, and choice statements. If the number of distinct values assigned to the signal is greater than or equal to the specified value, a binary multiplexer is implemented, otherwise a one-hot multiplexer is implemented.

Example

Consider the following RTL code.

```
module bmux(a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8,a_9,a_10,out_1,sel);
  input a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8,a_9,a_10;
  input [3:0] sel;
  output      out_1;
  reg        out_1;
  always @((sel or a_1 or a_2 or a_3 or a_4 or a_5 or a_6 or a_7 or a_8 or a_9 or a_10))
  begin
    case(sel)
      4'd6: out_1 = a_1;
      4'd11: out_1 = a_2;
      4'd13: out_1 = a_3;
      4'd9: out_1 = a_4;
      4'd7: out_1 = a_5;
      4'd3: out_1 = a_6;
      4'd4: out_1 = a_7;
      4'd2: out_1 = a_8;
      4'd14: out_1 = a_9;
      4'd15: out_1 = a_10;
      default: out_1 = 1'b0;
    endcase
  end
endmodule // bmux
```

Including the default branch, there are 11 case items in this case statement, which are addressed using a 4 bit `sel` signal. The tool needs to implement a 16-to-1 binary multiplexer to select among these 11 case items.

- Assume the following script:

```
set_attr library tutorial.lbr /
set_attr hdl_case_mux_threshold 17 /
read_hdl test.v
elaborate
```

The number of distinct values supported by the binary mux needed (16) is less than value of the `hdl_case_mux_threshold` attribute (17). Therefore, the tool will use a *one hot mux* to implement this case statement.

- Assume the following script:

```
set_attr hdl_case_mux_threshold 15 /
set_attr library tutorial.lbr /
read_hdl test.v
elaborate
```

The number of distinct values supported by the binary mux needed (16) is more than value of the `hdl_case_mux_threshold` attribute (15). Therefore, the tool will use a binary multiplexer to implement this case statement.

Related Information

Affects this command: [elaborate](#)

hdl_case_sensitive_instances

`hdl_case_sensitive_instances {none | false | true}`

Default: `none`

Read-write root attribute. Controls how Verilog instances must be linked to modules. Following the language rules, instances in Verilog files will by default be linked case sensitively to modules in Verilog/ VHDL files, while instances in VHDL files will be linked case-insensitively to modules in Verilog/VHDL files. This attribute can have the following values:

<code>false</code>	Allows case-insensitive instance name matching for Verilog instances to modules in Verilog /VHDL files.
<code>none</code>	Indicates that the parent language rules govern.
<code>true</code>	Allows only case-sensitive instance name matching for Verilog instances to modules in Verilog /VHDL files during linking.

Related Information

Using Case Sensitivity in Verilog/VHDL Mixed-Language Designs in Using Encounter RTL Compiler

Affects this command: [elaborate](#)

hdl_create_label_for_unlabeled_generate

hdl_create_label_for_unlabeled_generate {true | false}

Default: true

Read-write root attribute. Controls whether to create a label for an unnamed block. The created label will be of the form genblkx.

Examples

For the following examples consider the following RTL Verilog code:

```
module top ();
    parameter SIZE = 2;
    genvar i, j, k, m;
    generate
        for (i=0; i<SIZE+1; i=i+1) begin // first scope
            M1 N1();
            for (j=0; j<SIZE; j=j+1) begin // second scope
                M2 N2();
            end
        end
    endgenerate
endmodule // top
```

- The following script uses the default setting of the hdl_create_label_for_unlabeled_generate attribute.

```
set_attr library tutorial.lbr
set_attribute hdl_create_label_for_unlabeled_generate true /
read_hdl test.v -v2001
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module top;
    M1 \genblk1[0].N1 ();
    M2 \genblk1[0].genblk1[0].N2 ();
    M2 \genblk1[0].genblk1[1].N2 ();
    M1 \genblk1[1].N1 ();
    M2 \genblk1[1].genblk1[0].N2 ();
    M2 \genblk1[1].genblk1[1].N2 ();
    M1 \genblk1[2].N1 ();
    M2 \genblk1[2].genblk1[0].N2 ();
    M2 \genblk1[2].genblk1[1].N2 ();
endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- The following script uses the `false` setting of the `hdl_create_label_for_unlabeled_generate` attribute.

```
set_attr library tutorial.lbr
set_attribute hdl_create_label_for_unlabeled_generate false /
read_hdl test.v -v2001
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module top;
  M1 N1();
  M2 N2();
  M2 N2_0();
  M1 N1_0();
  M2 N2_1();
  M2 N2_2();
  M1 N1_1();
  M2 N2_3();
  M2 N2_4();
endmodule
```

Related Information

Affects this command: [elaborate](#)

hdl_decimal_parameter_name

```
hdl_decimal_parameter_name {false | true}
```

Default: `false`

Read-write `root` attribute. Controls the format for parameters that are appended to the names of instantiated parameterized modules in the netlist after elaboration. When set to `true`, the tool uses a decimal format for parameters with values less than or equal to the size of a 32-bit integer, prepending the value with width and sign information (if any). Otherwise, the tool uses binary or hex format as applicable.

Example

Consider the following RTL:

```
// RTL file test.v
module TOP(q, d);
    output q;
    input d;
    SUB #(32'b1) u1(q, d);
endmodule
module SUB(q, d);
    parameter p = 123;
    output q;
    input d;
    assign q = d > p;
endmodule
```

- Using the default (`false`) setting of the `hdl_decimal_parameter_name` attribute, the following netlist is printed after elaboration:

```
module TOP(q, d);
    input d;
    output q;
    wire d;
    wire q;
    SUB_p32h00000001 u1(q, d);
endmodule
.....
.....
```

- Setting the `hdl_decimal_parameter_name` attribute to `true` before elaboration, results in the following netlist is printed:

```
module TOP(q, d);
    input d;
    output q;
    wire d;
    wire q;
    SUB_p32d1 u1(q, d);
endmodule
.....
.....
```

Related Information

Affects this command: [elaborate](#)

Related attribute: [hdl_parameter_naming_style](#) on page 750

hdl_delete_transparent_latch

```
hdl_delete_transparent_latch {true | false}
```

Default: true

Read-write root attribute. Controls whether transparent latches are preserved or deleted during elaboration. When set to `true`, deletes latches that are always enabled.

Note: This attribute is supported only in the RTL flow.

Related Information

Affects this command: [elaborate](#)

hdl_enable_non_default_library_domain_binding

```
hdl_enable_non_default_library_domain_binding {true | false}
```

Default: true

Read-write root attribute. Enables the linking of cells to library cells in non default library domains during elaboration. Set this attribute to `false` to only link to library cells in the default library domain during elaboration.

Note: In the CPF flow, you should use the default setting to save runtime later in the flow.

Related Information

Affects this command: [elaborate](#)

hdl_enable_proc_name

```
hdl_enable_proc_name {false | true}
```

Default: false

Read-write root attribute. When set to `true`, updates the value of the `hdl_proc_name` instance attribute for sequential elements during elaboration.

Related Information

Affects these commands: [elaborate](#)

Affects this attribute (hdl_arch) [hdl_proc_name](#) on page 843

hdl_error_on_blackbox

hdl_error_on_blackbox {false | true}

Default: false

Read-write [root](#) attribute. When set to true, issues an error message if there is an unresolved reference (black-box) during elaboration.

Note: This attribute is supported only in the RTL flow.

Related Information

[Global and User Control of Elaboration in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [elaborate](#)

Related attribute: (hdl_arch) [hdl_error_on_blackbox](#) on page 860

hdl_error_on_latch

hdl_error_on_latch {false | true}

Default: false

Read-write [root](#) attribute. When set to true, issues an error message if a latch is inferred for a design.

Note: This attribute is supported only in the RTL flow.

Related Information

[Global and User Control of Elaboration in HDL Modeling in Encounter RTL Compiler](#)

Affects these commands: [elaborate](#)

Related attribute: (hdl_arch) [hdl_error_on_latch](#) on page 861

hdl_error_on_logic_abstract

`hdl_error_on_logic_abstract {false | true}`

Default: `false`

Read-write root attribute. When set to `true`, issues an error message if a logic abstract is inferred for a design.

Related Information

[Global and User Control of Elaboration in HDL Modeling in Encounter RTL Compiler](#)

Affects these commands: [elaborate](#)

Related attribute: (hdl_arch) [hdl_error_on_logic_abstract](#) on page 861

hdl_error_on_negedge

```
hdl_error_on_negedge {false | true}
```

Default: false

Read-write root attribute. When set to `true`, issues an error message if a design infers a flip-flop that is triggered by a falling clock edge. In case where the `root` and `hdl_arch` `hdl_error_on_negedge` attributes are set to different values, the last specification takes precedence. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

Note: This attribute is supported only in the RTL flow.

Related Information

[Global and User Control of Elaboration in HDL Modeling in Encounter RTL Compiler](#)

Affects these commands: [elaborate](#)

Related attribute: (hdl_arch) [hdl_error_on_negedge](#) on page 862

hdl_ff_keep_explicit_feedback

```
hdl_ff_keep_explicit_feedback {true | false}
```

Default: true

Read-write root attribute. Controls how flip-flop stable states are implemented for feedback assignments that are explicitly specified in the RTL.

Note: This attribute is ignored, and all flip-flop feedback is removed when the `lp_insert_clock_gating` attribute is set to `true`.

This attribute is supported only in the RTL flow. This attribute will overwrite and will be the initial value for `hdl_arch` object type as well. That is,
`hdl_ff_keep_explicit_feedback` on a `root` object will overwrite and be the initial value for `hdl_ff_keep_explicit_feedback` on a `hdl_arch` object type.

In case the `root` and `hdl_arch` `hdl_ff_keep_explicit_feedback` attributes are set to different values, the last one takes precedence. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

Examples

- The following command implements flip-flop stable states for feedback assignments. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

```
rc:/> set_attribute hdl_ff_keep_explicit_feedback true /
```

- The following command implements a synchronous enable signal from the Q output to the D input for the RTL. See the example RTL and the corresponding schematic in the HDL-Related Attributes section of the *HDL Modeling in Encounter RTL Compiler* manual.

```
rc:/> set_attribute hdl_ff_keep_feedback false /  
rc:/> set_attribute hdl_ff_keep_explicit_feedback false /
```

Related Information

[Global and User Control of Elaboration in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [elaborate](#)

Related attribute (hdl_arch) [hdl_ff_keep_explicit_feedback](#) on page 862

hdl_ff_keep_feedback

`hdl_ff_keep_feedback {false | true}`

Default: `false`

Read-write `root` attribute. Controls how flip-flop stable states are implemented. When set to `true`, implements a feedback path from the Q output to the D input. When set to `false`, uses a synchronous flip-flop enable signal to implement the stable states.

In case the root and hdl_arch `hdl_ff_keep_feedback` attributes are set to different values, the last one takes precedence. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

Note: This attribute is ignored, and all flip-flop feedback is removed when the `lp_insert_clock_gating` attribute is set to `true`.

This attribute is supported only in the RTL flow. It will overwrite and will be the initial value for `hdl_arch` object type as well. That is, `hdl_ff_keep_feedback` on a `root` object will overwrite and be the initial value for `hdl_ff_keep_feedback` on a `hdl_arch` object.

Examples

- The following command implements a feedback path from the `Q` output to the `D` input for the RTL. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

```
rc:/> set attribute hdl ff keep feedback true /
```

- The following command implements a synchronous enable signal from the Q output to the D input for the RTL. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

```
rc:/> set attribute hdl ff keep feedback false /
```

Related Information

Global and User Control of Elaboration in *HDL Modeling in Encounter RTL Compiler*

Affects these commands: elaborate

read hdl

Related attribute (hdl arch) hdl ff keep feedback on page 863

hdl flatten complex port

hdl flatten complex port {false | true}

Default: false

Read-write root attribute. Controls how complex ports in the RTL are represented in the generated netlist.

A port is considered complex, when it is not declared as a simple vector of bits, nor as a single bit scalar signal. For example, a port declared using a structure `typedef` is a complex port.

When you set this attribute to `true`, elaboration creates a single lumped (packed) vector port for any complex port. The bits in the single vector port (created with `true` setting) are set in the order in which fields/dimensions are ordered in the definition of the complex port. By default, elaboration generates (and names) multiple bit-vector ports, which correspond to various disjoint fields/pieces of the complex port.

Limitation: This attribute has no impact on any complex port that is defined using interface modport. The tool would continue to generate (and name) multiple ports for such a complex port.

Examples

- Consider the following RTL file with Verilog struct port:

```

typedef struct packed {
    bit a;
    bit b;
} type_1;
typedef struct packed {
    type_1 t1;
    bit c;
} type_2;
module test( input type_2 i_var, output type_2 o_var);
    assign o_var = i_var;
endmodule

```

Using the default setting of the `hdl_flatten_complex_port` attribute, the netlist generated after elaboration would look as follows:

```

module test(\i_var[c], \i_var[t1][b] , \i_var[t1][a] , \o_var[c] ,
\o_var[t1][b] , \o_var[t1][a] );
    input \i_var[c] , \i_var[t1][b] , \i_var[t1][a] ;
    output \o_var[c] , \o_var[t1][b] , \o_var[t1][a] ;
    wire \i_var[c] , \i_var[t1][b] , \i_var[t1][a] ;
    wire \o_var[c] , \o_var[t1][b] , \o_var[t1][a] ;
    assign \o_var[t1][a] = \i_var[t1][a] ;
    assign \o_var[t1][b] = \i_var[t1][b] ;
    assign \o_var[c] = \i_var[c] ;
endmodule

```

By setting the `hdl_flatten_complex_port` attribute to true, the netlist generated after elaboration would look as follows:

```

module test(i_var, o_var);
    input [2:0] i_var;
    output [2:0] o_var;
    wire [2:0] i_var;
    wire [2:0] o_var;
    assign o_var[0] = i_var[0];
    assign o_var[1] = i_var[1];
    assign o_var[2] = i_var[2];
endmodule

```

- Consider the following RTL file with the ports defined using VHDL declared type:

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
package package_Array_2d is
    subtype bus2 is unsigned (1 downto 0);
    type new_bus is array (0 to 1) of bus2;
end package_Array_2d;
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use WORK.package_Array_2d.all;

entity sub is
    port ( var_in : in new_bus;
           var_out : out new_bus);

```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

```
end sub;
architecture rtl of sub is
begin
    var_out <= var_in;
end rtl;
```

Using the default setting of the `hdl_flatten_complex_port` attribute, the netlist generated after elaboration would look as follows:

```
module sub(\var_in[1] , \var_in[0] , \var_out[1] , \var_out[0] );
    input [1:0] \var_in[1] , \var_in[0] ;
    output [1:0] \var_out[1] , \var_out[0] ;
    wire [1:0] \var_in[1] , \var_in[0] ;
    ...
endmodule
```

By setting the `hdl_flatten_complex_port` attribute to true, the netlist generated after elaboration would look as follows:

```
module sub(var_in, var_out);
    input [3:0] var_in;
    output [3:0] var_out;
    wire [3:0] var_in;
    wire [3:0] var_out;
    assign var_out[0] = var_in[0];
    ...
endmodule
```

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_generate_index_style

`hdl_generate_index_style string`

Default: %s[%d]

Read-write `root` attribute. Specifies the format to be used to compose the instance name of instances instantiated inside a `for`-generate statement.

The format string contains the `for`-generate block label (%s) and optionally the individual `for`-generate bit index (%d). The default value of %s[%d] complies with the naming rules defined by Section 12.1.3.1 in Verilog-2001 LRM, as well as by Section 12.4.1 in Verilog-2005 LRM.

In VHDL, a generate statement must have a label. In Verilog, the block of a generate statement can be unnamed. If a generate block is unlabeled, it is given a name based on the rules defined in the Verilog-2005 LRM (in Section 12.4.3 of IEEE Std 1364-2005).

You must specify this attribute before you elaborate the design.

This attribute affects instances in a `for`-generate statement. It does not affect instances in a `if`-generate or `case`-generate statement.

Examples

- For the following examples consider the following RTL Verilog code:

```
module top ();
    parameter SIZE = 2;
    genvar i, j, k, m;
    generate
        for (i=0; i<SIZE+1; i=i+1) begin:B1      // scope B1[i]
            M1 N1();                         // instantiates B1[i].N1[i]
            for (j=0; j<SIZE; j=j+1) begin:B2    // scope B1[i].B2[j]
                M2 N2();                     // instantiates B1[i].B2[j].N2
            end
        end
    endgenerate
endmodule // top
```

The generate block in this module has two block labels: B1 and B2.

- The following script uses the default setting of the `hdl_generate_index_style` attribute.

```
set_attr library tutorial.lbr
set_attribute hdl_generate_index_style %s[%d]
read_hdl test.v -v2001
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module top;
    M1 \B1[0].N1 ();
    M2 \B1[0].B2[0].N2 ();
    M2 \B1[0].B2[1].N2 ();
    M1 \B1[1].N1 ();
    M2 \B1[1].B2[0].N2 ();
    M2 \B1[1].B2[1].N2 ()
    M1 \B1[2].N1 ();
    M2 \B1[2].B2[0].N2 ();
    M2 \B1[2].B2[1].N2 ();
endmodule
```

- The following script uses a different value for the `hdl_generate_index_style` attribute.

```
set_attr library tutorial.lbr
set_attribute hdl_generate_index_style %s_%d
read_hdl test.v -v2001
elaborate
write_hdl > foo.v
```

After elaboration, the netlist will look like:

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

```
module top;
    M1 \B1_0.N1 ();
    M2 \B1_0.B2_0.N2 ();
    M2 \B1_0.B2_1.N2 ();
    M1 \B1_1.N1 ();
    M2 \B1_1.B2_0.N2 ();
    M2 \B1_1.B2_1.N2 ();
    M1 \B1_2.N1 ();
    M2 \B1_2.B2_0.N2 ();
    M2 \B1_2.B2_1.N2 ();
endmodule
```

- The following example uses VHDL code.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity top is
end top;
architecture rpl of top is
constant SIZE: integer := 2;

component M1 is
end component;

component M2 is
end component;

begin
    B1 : for i in 0 to SIZE generate
        N1: M1;
        B2 : for j in 0 to SIZE-1 generate
            N2: M2;
        end generate;
    end generate;
end rpl;
```

The generate block in this module has two block labels: B1 and B2.

The following script uses the %s_%d value for the hdl_generate_index_style attribute.:.

```
set_attr library tutorial.lbr
set_attribute hdl_generate_index_style %s_%d
read_hdl test.v -vhdl
elaborate
write_hdl > foo.v
```

After elaboration, the netlist will look like:

```
module M1;
endmodule

module M2;
endmodule

module top;
    M1 \B1_0.N1 ();
    M2 \B1_0.B2_0.N2 ();
    M2 \B1_0.B2_1.N2 ();
    M1 \B1_1.N1 ();
```

```
M2 \B1_1.B2_0.N2 ();
M2 \B1_1.B2_1.N2 ();
M1 \B1_2.N1 ();
M2 \B1_2.B2_0.N2 ();
M2 \B1_2.B2_1.N2 ();
endmodule
```

Related Information

Affects this command: [elaborate](#)

Related attribute: [hdl_generate_separator](#) on page 742
[hdl_use_block_prefix](#) on page 775

hdl_generate_separator

`hdl_generate_separator string`

Default: .

Read-write root attribute. Specifies the separator string that appears between block labels in the instance name of an instance instantiated inside a generate statement. A generated instance name contains multiple block labels if it is inside of nested generate statements.

In Verilog, each layer of the nested generate can be a for-generate, an if-generate, or a case-generate. In VHDL, each layer can be either a for-generate or an if-generate.

You must specify this attribute before you elaborate the design.

Examples

For the following examples consider the following RTL code:

```
module top ();
    parameter SIZE = 2;
    genvar i, j, k, m;
    generate
        for (i=0; i<SIZE+1; i=i+1) begin:B1      // scope B1[i]
            M1 N1();                      // instantiates B1[i].N1[i]
            for (j=0; j<SIZE; j=j+1) begin:B2      // scope B1[i].B2[j]
                M2 N2();                  // instantiates B1[i].B2[j].N2
            end
        end
    endgenerate
endmodule // top
```

The generate block in this module has two block labels: B1 and B2.

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- Using the following script:

```
set_attr library tutorial.lbr
set_attribute hdl_generate_separator . /
read_hdl test.v -v2001
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module top;
  M1 \B1[0].N1 ();
  M2 \B1[0].B2[0].N2 ();
  M2 \B1[0].B2[1].N2 ();
  M1 \B1[1].N1 ();
  M2 \B1[1].B2[0].N2 ();
  M2 \B1[1].B2[1].N2 ()
  M1 \B1[2].N1 ();
  M2 \B1[2].B2[0].N2 ();
  M2 \B1[2].B2[1].N2 ();
endmodule
```

- Using the following script:

```
set_attr library tutorial.lbr
set_attribute hdl_generate_separator / /
read_hdl test.v -v2001
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module top;
  M1 \B1[0]/N1 ();
  M2 \B1[0]/B2[0]/N2 ();
  M2 \B1[0]/B2[1]/N2 ();
  M1 \B1[1]/N1 ();
  M2 \B1[1]/B2[0]/N2 ();
  M2 \B1[1]/B2[1]/N2 ()
  M1 \B1[2]/N1 ();
  M2 \B1[2]/B2[0]/N2 ();
  M2 \B1[2]/B2[1]/N2 ();
endmodule
```

Related Information

Affects this command: [elaborate](#)

Related attributes: [hdl_generate_index_style](#) on page 739

[hdl_use_block_prefix](#) on page 775

[hdl_use_if_generate_prefix](#) on page 779

hdl_index_mux_threshold

`hdl_index_mux_threshold integer`

Default: 8

Read-write root attribute. Specifies the minimum number of data inputs the tool requires to implement binary multiplexers instead of AND/OR logic for variable index array references, such as $y = x[i]$.

Example

Consider the following RTL code:

```
module top (bit_out, dword, sel);
    output big_out;
    input [3:0] dword;
    input [1:0] sel;
    assign big_vect = dword[sel];
endmodule
```

Using the following script:

```
set_attr library tutorial.lbr
set_attribute hdl_index_mux_threshold 4 /
read_hdl test.v
elab
```

The number of data inputs in the array indexing (4) is not less than the value of the `hdl_index_mux_threshold` attribute (4). Therefore, a 4-to-1 binary mux is used to implement the selection of one of the four bits of signal `dword`.

Related Information

Affects these commands: [elaborate](#)

hdl_infer_unresolved_from_logic_abstract

`hdl_infer_unresolved_from_logic_abstract {true | false}`

Default: true

Read-write root attribute. Controls how a logic abstract that is inferred from an empty module, an entity without an architecture, or an entity whose architecture is empty, is interpreted.

If set to `true`, the logic abstract becomes an unresolved reference in the design. RTL Compiler tries to find a library cell with a name corresponding to that of the logic abstract. If

it finds such a cell, the logic abstract is represented as an instance of that library cell. Otherwise, the logic abstract becomes an unresolved reference.



If set to false, the logic abstract remains as a subdesign in the design hierarchy. In this case, an empty subdesign is created which is not considered unresolved. This is an unrealistic situation. The tool would treat the output ports of the module as undriven and would assign them values based on the setting of the hdl_undriven_output_port_value attribute. This can potentially create false non-eq points during verification with Lec. Therefore, this setting should be avoided.

Note: This attribute is supported in the RTL and structural flows.

Related Information

[Modeling Logic Abstracts in HDL Modeling in Encounter RTL Compiler](#).

Affects these commands: [elaborate](#)

[read_hdl](#)

[read_netlist](#)

Affected by this attribute: [hdl_use_techelt_first](#) on page 783

hdl_instance_array_naming_style

`hdl_instance_array_naming_style string`

Default: %s[%d]

Read-write [root](#) attribute. Specifies the format used to name individual instance names of an array in the RTL. %s is the array name and %d is the individual bit. Set this attribute before using the `elaborate` command.

Note: This attribute is supported only in the RTL flow.

Examples

For the following examples consider the following RTL Verilog code:

```
module test1(a,b,y);
    input[2:0] a,b;
    output [2:0] y;
    nand2 G [2:0] (.A(a), .B(b), .Y(y));
endmodule
```

- The following script uses the default setting of the `hdl_instance_array_naming_style` attribute.

```
set_attr library tutorial.lbr
set_attribute hdl_instance_array_naming_style %s\[%d\] /
read_hdl test1.v
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module test1(a, b, y);
    input [2:0] a, b;
    output [2:0] y;
    wire [2:0] a, b;
    wire [2:0] y;
    nand2 \G[2] (.A (a[2]), .B (b[2]), .Y (y[2]));
    nand2 \G[1] (.A (a[1]), .B (b[1]), .Y (y[1]));
    nand2 \G[0] (.A (a[0]), .B (b[0]), .Y (y[0]));
endmodule
```

- The following script changes the setting of the `hdl_instance_array_naming_style` attribute.

```
set_attr library tutorial.lbr
set_attribute hdl_instance_array_naming_style %sx%dx /
read_hdl test1.v
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module test1(a, b, y);
    input [2:0] a, b;
    output [2:0] y;
    wire [2:0] a, b;
    wire [2:0] y;
    nand2 Gx2x (.A (a[2]), .B (b[2]), .Y (y[2]));
    nand2 Gx1x (.A (a[1]), .B (b[1]), .Y (y[1]));
    nand2 Gx0x (.A (a[0]), .B (b[0]), .Y (y[0]));
endmodule
```

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_interface_separator

```
hdl_interface_separator string
```

Default: _

Read-write root attribute. Specifies the string used to separate the interface elements.

Examples

For the following examples consider the following System Verilog code:

```
interface nand_intf (
    input wire [1:0] a,
    input wire [1:0] b,
    output wire [1:0] y
);
endinterface
module test1(nand_intf intf);
assign intf.y = intf.a & intf.b;
endmodule
```

- The following script uses the default setting of the hdl_interface_separator attribute.

```
set_attr library tutorial.lbr
read_hdl -sv nand_intf.v
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module test1(intf_a, intf_b, intf_y);
    input [1:0] intf_a, intf_b;
    output [1:0] intf_y;
    wire [1:0] intf_a, intf_b;
    wire [1:0] intf_y;
    and g1 (intf_y[0], intf_a[0], intf_b[0]);
    and g2 (intf_y[1], intf_a[1], intf_b[1]);
endmodule
```

- The following script changes the setting of the hdl_interface_separator attribute.

```
set_attr library tutorial.lbr
set_attribute hdl_interface_separator . /
read_hdl -sv nand_intf.v
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module test1(\intf.a , \intf.b , \intf.y );
    input [1:0] \intf.a , \intf.b ;
    output [1:0] \intf.y ;
    wire [1:0] \intf.a , \intf.b ;
    wire [1:0] \intf.y ;
    and g1 (\intf.y [0], \intf.a [0], \intf.b [0]);
    and g2 (\intf.y [1], \intf.a [1], \intf.b [1]);
endmodule
```

Related Information

Affects these commands: [elaborate](#)
 [write_hdl](#)

hdl_latch_keep_feedback

`hdl_latch_keep_feedback {false | true}`

Default: `false`

Read-write [root](#) attribute. Controls how explicitly-specified latch stable states (for example, `q <= q`) are implemented. When set to `true`, implements a feedback path from the Q output to the D input, resulting in a combinational loop. When set to `false`, implements a latch with an enable signal.

In case the root and `hdl_arch` `hdl_latch_keep_feedback` attributes are set to different values, the last one takes precedence. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

Note: This attribute is supported only in the RTL flow.

This attribute will overwrite or will be the initial value for `hdl_arch` object type as well. That is, `hdl_latch_keep_feedback` on a `root` object will overwrite or be the initial value for `hdl_latch_keep_feedback` on a `hdl_arch` object type.

Examples

- For the following command, RTL Compiler implements a feedback path from the Q output to the D input for the RTL, resulting in a combinational loop. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

```
rc:/> set_attribute hdl_latch_keep_feedback true /
```

- For the following command, RTL Compiler implements a latch with an enable signal specified in the RTL. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

```
rc:/> set_attribute hdl_latch_keep_feedback false /
```

Related Information

[Global and User Control of Elaboration in HDL Modeling in Encounter RTL Compiler](#)

Affects these commands: [elaborate](#)
 [read_hdl](#)

Related attribute: (hdl_arch) [hdl_latch_keep_feedback](#) on page 863

hdl_max_loop_limit

`hdl_max_loop_limit integer`

Default: 1024

Read-write [root](#) attribute. Determines the maximum number of iterations for unfolding a loop construct of any type. RTL Compiler stops and produces an error message when it needs to unroll a loop that has more iterations than the specified threshold.

This is a safety measure to avoid infinite unrolling, as well as to avoid loops with an unexpectedly large number of iterations, which can cause errors in the RTL code. Usually the overflow detection mechanism catches overflow conditions that cause infinite unrolling.

For example, unrolling the following loop causes an infinite loop and is therefore disallowed by the overflow detection mechanism:

```
reg [3:0] i;  
for (i=0; i<16; i=i+1)  
...
```

Being 4-bit wide, the loop index goes back to 0 after reaching 15.

The overflow detection mechanism also catches infinite unrolling that is *not* caused by overflow conditions. For example:

```
parameter incr = 0;  
reg [4:0] i;  
for (i=0; i<16; i=i+incr)  
...
```

Note: This attribute is supported only in the RTL flow.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_max_map_to_mux_control_width

`hdl_max_map_to_mux_control_width integer`

Default: 10

Read-write root attribute. Specifies the maximum size of multiplexers that can be generated by RTL Compiler for HDL case statements that are marked with the `map_to_mux` pragma. If the width of the `case` condition, which is implemented as the multiplexer control input, exceeds the value set by this attribute, RTL Compiler will ignore the pragma and generate more efficient logic using AND and OR gates.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_max_recursion_limit

`hdl_max_recursion_limit integer`

Default: 1024

Read-write root attribute. Sets the maximum number of elaborations for recursive instantiations to prevent possible infinite recursions.

Note: This attribute is supported only in the RTL flow.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_parameter_naming_style

`hdl_parameter_naming_style string`

Default: `_%s%d`

Read-write root attribute. Specifies the format of the suffix added to the original module name for each parameter overwrite.

Each unqualified parameterized module is named by concatenating the original module name and a number of suffix strings, one string per parameter.

For the top-level module, the attribute applies to the parameters whose values are specified with the `-parameters` option of the `elaborate` command.

For an instantiated module, the attribute is effective if both of the following conditions apply:

- The module name of the child module is not parameterized.
- The parameter value is given by the instantiation statement.

This attribute is useful when you have a design where Verilog parameters or generic ports in the VHDL are causing long module names that exceed another tool's internal limit.

Note: This attribute is supported only in the RTL flow.

Examples

- If module `adder` has two parameters, `w` and `A`, and the module is instantiated with the following parameter overwrite `adder #(10, 2)`, the subdesign name corresponding to `adder` with `W=10` and `A=2` is:
 - `adder_10_2` if `parameter_naming_style` was set to `_%d`
 - `adder_W_10_A_2` if `parameter_naming_style` was set to `_%s_%d`
 - `adder_W10_A2` if `parameter_naming_style` was set to `_%s%d`
- Consider the following RTL code:

```
module top (z, a, b, c);
    parameter na = 12, nb = 7, nc = 9;
    input [3:0] a, b, c;
    output [3:0] z;
    btm #(na, nb) u1 (z, a, b, c); // <== Note 2
endmodule
module btm (y, d, e, f);
    parameter ma = 12, mb = 7, mc = 9;
    input [3:0] d, e, f;
    wire [3:0] p, q, r;
    output [3:0] y;
    assign p = d & ma;
    assign q = e & mb;
    assign r = f & mc;
    assign y = p ^ q ^ r;
endmodule
```

Using the following script:

```
set_attr library tutorial.lbr
read_hdl test.v
elaborate -parameters {{nb 11} {nc 9}} } # <== Note 1
write_hdl
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

The netlist after elaboration will look like:

```
module top_nb11_nc9 (z, a, b, c); // <== Note 1
    input [3:0] a, b, c;
    output [3:0] z;
    btm_ma12_mb11 u1 (z, a, b, c); // <== Note 2
endmodule
module btm_ma12_mb11 (y, d, e, f); // <== Note 2
    input [3:0] d, e, f;
    output [3:0] y;
    wire n_19
    assign y[2] = d[2];
    assign y[1] = e[1];
    xor g04 (n_19, d[3], e[3]);
    xor g21 (y[3], n_19, f[3]);
    xor g18 (y[0], e[0], f[0]);
endmodule
```

Note 1: parameterized name `top_nb11_nc9` has `nb` and `nc`, but not `na` because `na` was not specified with the `-parameters` option.

Note 2: parameterized name `btm_ma12_mb11` has `ma` and `mb`, but not `mc` because no value was specified for `mc` in the instantiation statement.

Related Information

[Naming Parameterized Modules](#) in *Using Encounter RTL Compiler* manual for detailed examples.

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_preserve_async_sr_priority_logic

`hdl_preserve_async_sr_priority_logic {false | true}`

Default: `false`

Read-write `root` attribute. When set to `true`, RTL Compiler is prevented from generating redundant logic when, during mapping, a flip-flop cell is selected that has the same priority for the asynchronous set and reset operations as specified in the input HDL.

Related Information

Affects this command: [elaborate](#)

hdl_preserve_dangling_output_nets

```
hdl_preserve_dangling_output_nets {true | false}
```

Default: true

Read-write root attribute. When set to true, RTL Compiler preserves the names of dangling output nets in designs that are read using the `read_netlist` command or the `read_hdl -netlist` command.

Note: This attribute is supported only in the structural flow.

Related Information

Affects these commands: [read_netlist](#)

[read_hdl -netlist](#)

hdl_preserve_supply_nets

```
hdl_preserve_supply_nets {true | false}
```

Default: true

Read-write root attribute. When set to true, RTL Compiler preserves the supply nets in the design that is read in.

Examples

The following module has two supply nets in the design.

```
module test_64(Y, Z, A);
    input A;
    output Y, Z;

    supply0 s0;
    supply1 s1;

    assign Z = s1;
    and u1 (Y, s0, A);

endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- Using the default value for the `hdl_preserve_supply_nets` attribute, produces the following post-elaboration netlist:

```
module test_64(Y, Z, A);
    input A;
    output Y, Z;
    wire A;
    wire Y, Z;
    supply0 s0;
    supply1 s1;
    assign Z = s1;
    and u1 (Y, s0, A);
endmodule
```

- Setting the value for the `hdl_preserve_supply_nets` attribute to `false`, produces the following post-elaboration netlist:

```
module test_64(Y, Z, A);
    input A;
    output Y, Z;
    wire A;
    wire Y, Z;
    assign Z = 1'b1;
    and u1 (Y, 1'b0, A);
endmodule
```

Related Information

Affects these commands: [elaborate](#)
 [read_netlist](#)
 [write_hdl](#)

hdl_preserve_sync_ctrl_logic

`hdl_preserve_sync_ctrl_logic {false | true}`

Default: `false`

Read-write `root` attribute. When set to `true`, RTL Compiler will try to keep the synchronous control logic close to the flip-flops and prevent merging it with surrounding logic in the fanin cone.

You must set this attribute before reading in the design so RTL Compiler can automatically preserve the synchronous control pins of the flip-flops.

Related Information

Affects this command: [elaborate](#)

hdl_preserve_sync_set_reset

```
hdl_preserve_sync_set_reset {false | true}
```

Default: false

Read-write root attribute. When set to true, RTL Compiler will try to keep the synchronous set (reset) control logic close to the flip-flops and prevent merging it with surrounding logic in the fanin cone.

You must set this attribute before reading in the design so RTL Compiler can automatically preserve the synchronous control pins of the flip-flops.

Related Information

Affects this command: [elaborate](#)

hdl_preserve_unused_flop

```
hdl_preserve_unused_flop {false | true}
```

Default: false

Read-write root attribute. When set to true, RTL Compiler does not remove flip-flops that do not, directly or indirectly, affect any outputs. This can be used, for example, to keep flops that are only used to observe internal nets through scan chains in test mode.

This attribute only affects flops that are inferred by elaboration. It does not affect flops that are explicitly instantiated in the HDL code.

Note: This attribute is supported only in the RTL flow.

Examples

In the following example, the `hdl_preserve_unused_flop` attribute affects the `xl_3_reg` and `yf_3_reg` instances that are inferred during elaboration. It does not affect the `u3` and `u4` instances that are explicitly instantiated in the HDL code.

```
module test (y, d, clk);
    input clk, d;
    output y;
    wire xl_2, yf_2;
    reg xl_3, yf_3;
    latch_in_lib u3 (.D(d), .G(clk), .Q(xl_2));
    flop_in_lib u4 (.CP(clk), .D(d), .Q(yf_2));
    always @(clk or d)
        begin
            if (clk)
                xl_3 <= d;
        end
    always @ (posedge clk)
        begin
            yf_3 <= d;
        end
    assign y = ~d;
endmodule
```

- Using the default setting (`false`) of the `hdl_preserve_unused_flop` attribute:

```
rc:/> set_attribute hdl_preserve_unused_flop false
rc:/> set_attribute library my_lib.lib
rc:/> read_hdl test.v
rc:/> elaborate
rc:/> write_hdl
```

Creates the following post-elaboration netlist:

```
module test (y, d, clk);
    input d, clk;
    output y;
    wire d, clk;
    wire y;
    wire xl_2, yf_2;
    latch_in_lib u3(.D (d), .G (clk), .Q (xl_2));
    flop_in_lib u4(.CP (clk), .D (d), .Q (yf_2));
    not g1 (y, d);
endmodule
```

- Setting the `hdl_preserve_unused_flop` attribute to `true`, creates the following post-elaboration netlist:

```
module test (y, d, clk);
    input d, clk;
    output y;
    wire d, clk;
    wire y;
    wire UNCONNECTED, xl_2, yf_2;
    latch_in_lib u3(.D (d), .G (clk), .Q (xl_2));
    flop_in_lib u4(.CP (clk), .D (d), .Q (yf_2));
    not g2 (y, d);
    CDN_flop yf_3_reg(.clk (clk), .d (d), .sena (1'b1), .aclr (1'b0),
        .apre (1'b0), .srl (1'b0), .srd (1'b0), .q (UNCONNECTED));
endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

```
`ifdef RC_CDN_GENERIC_GATE
`else
module CDN_flop(clk, d, sena, aclr, apre, srl, srd, q);
    input clk, d, sena, aclr, apre, srl, srd;
    output q;
    wire clk, d, sena, aclr, apre, srl, srd;
    wire q;
    reg qi;
    assign #1 q = qi;
    always
        @(posedge clk or posedge apre or posedge aclr)
            if (aclr)
                qi <= 0;
            else if (apre)
                qi <= 1;
            else if (srl)
                qi <= srd;
            else begin
                if (sena)
                    qi <= d;
            end
    initial
        qi <= 1'b0;
endmodule
`endif
```

Note: Using this attribute is ignored in the structural flow.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

Related attribute: (hdl_arch) [hdl_preserve_unused_registers](#) on page 865

hdl_preserve_unused_latch

hdl_preserve_unused_latch {false | true}

Default: false

Read-write root attribute. When set to true, RTL Compiler does not remove latches that do not, directly or indirectly, affect any outputs. This can be used, for example, to keep registers that are only used to observe internal nets through scan chains in test mode.

This attribute only affects latches that are inferred by elaboration. It does not affect latches that are explicitly instantiated in the HDL code.

Note: This attribute is supported only in the RTL flow.

Examples

In the following example, the `hdl_preserve_unused_latch` attribute affects the `xl_3_reg` and `yf_3_reg` instances that are inferred during elaboration. It does not affect the `u3` and `u4` instances that are explicitly instantiated in the HDL code.

```
module test (y, d, clk);
    input clk, d;
    output y;
    wire xl_2, yf_2;
    reg xl_3, yf_3;
    latch_in_lib u3 (.D(d), .G(clk), .Q(xl_2));
    flop_in_lib u4 (.CP(clk), .D(d), .Q(yf_2));
    always @(clk or d)
        begin
            if (clk)
                xl_3 <= d;
        end
    always @ (posedge clk)
        begin
            yf_3 <= d;
        end
    assign y = ~d;
endmodule
```

- Using the default setting (`false`) of the `hdl_preserve_unused_latch` attribute:

```
rc:/> set_attribute hdl_preserve_unused_latch false
rc:/> set_attribute library my_lib.lib
rc:/> read_hdl test.v
rc:/> elaborate
rc:/> write_hdl
```

Creates the following post-elaboration netlist:

```
module test (y, d, clk);
    input d, clk;
    output y;
    wire d, clk;
    wire y;
    wire xl_2, yf_2;
    latch_in_lib u3(.D (d), .G (clk), .Q (xl_2));
    flop_in_lib u4(.CP (clk), .D (d), .Q (yf_2));
    not g1 (y, d);
endmodule
```

- Setting the `hdl_preserve_unused_latch` attribute to `true`, creates the following post-elaboration netlist:

```
module test (y, d, clk);
    input d, clk;
    output y;
    wire d, clk;
    wire y;
    wire UNCONNECTED, xl_2, yf_2;
    latch_in_lib u3(.D (d), .G (clk), .Q (xl_2));
    flop_in_lib u4(.CP (clk), .D (d), .Q (yf_2));
    not g2 (y, d);
    CDN_latch xl_3_reg(.d (d), .ena (clk), .aclr (1'b0), .apre (1'b0),
        .q (UNCONNECTED));
endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

```
`ifdef RC_CDN_GENERIC_GATE
`else
module CDN_latch(ena, d, aclr, apre, q);
    input ena, d, aclr, apre;
    output q;
    wire ena, d, aclr, apre;
    wire q;
    reg  qi;
    assign #1 q = qi;
    always
        @(d or ena or apre or aclr)
            if (aclr)
                qi <= 0;
            else if (apre)
                qi <= 1;
            else begin
                if (ena)
                    qi <= d;
                end
        initial
            qi <= 1'b0;
endmodule
`endif
```

Note: Using this attribute is ignored in the structural flow.

Related Information

Affects these commands: [elaborate](#)

[read_hdl](#)

Related attribute: (hdl_arch) [hdl_preserve_unused_registers](#) on page 865

hdl_preserve_unused_registers

hdl_preserve_unused_registers {false | true}

Default: false

Read-write [root](#) attribute. When set to true, RTL Compiler does not remove registers (latches and flip-flops) that do not, directly or indirectly, affect any outputs. This can be used, for example, to keep registers that are only used to observe internal nets through scan chains in test mode.

This attribute only affects registers that are inferred by elaboration. It does not affect registers that are explicitly instantiated in the HDL code.

Note: This attribute is supported only in the RTL flow.

Examples

In the following example, the `hdl_preserve_unused_registers` attribute affects the `xl_3_reg` and `yf_3_reg` instances that are inferred during elaboration. It does not affect the `u3` and `u4` instances that are explicitly instantiated in the HDL code.

```
module test (y, d, clk);
    input clk, d;
    output y;
    wire xl_2, yf_2;
    reg xl_3, yf_3;
    latch_in_lib u3 (.D(d), .G(clk), .Q(xl_2));
    flop_in_lib u4 (.CP(clk), .D(d), .Q(yf_2));
    always @(clk or d)
        begin if (clk)
            xl_3 <= d;
        end
    always @ (posedge clk)
        begin
            yf_3 <= d;
        end
    assign y = ~d;
endmodule
```

- Using the default setting (`false`) of the `hdl_preserve_unused_registers` attribute:

```
rc:/> set_attribute hdl_preserve_unused_registers false
rc:/> set_attribute library my_lib.lib
rc:/> read_hdl test.v
rc:/> elaborate
rc:/> write_hdl
```

Creates the following post-elaboration netlist:

```
module test (y, d, clk);
    input d, clk;
    output y;
    wire d, clk;
    wire y;
    wire xl_2, yf_2;
    latch_in_lib u3(.D (d), .G (clk), .Q (xl_2));
    flop_in_lib u4(.CP (clk), .D (d), .Q (yf_2));
    not g1 (y, d);
endmodule
```

- Setting the `hdl_preserve_unused_registers` attribute to `true`, creates the following post-elaboration netlist:

```
module test (y, d, clk);
    input d, clk;
    output y;
    module test (y, d, clk);
        input d, clk;
        output y;
        wire d, clk;
        wire y;
        wire UNCONNECTED, UNCONNECTED0, xl_2, yf_2;
        latch_in_lib u3(.D (d), .G (clk), .Q (xl_2));
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

```
flop_in_lib u4(.CP (clk), .D (d), .Q (yf_2));
not g3 (y, d);
CDN_latch xl_3_reg(.d (d), .ena (clk), .aclr (1'b0), .apre (1'b0),
.q (UNCONNECTED));
CDN_flop yf_3_reg(.clk (clk), .d (d), .sena (1'b1), .aclr (1'b0),
.apre (1'b0), .srl (1'b0), .srd (1'b0), .q (UNCONNECTED0));
endmodule
`ifdef RC_CDN_GENERIC_GATE
`else
module CDN_latch(ena, d, aclr, apre, q);
    input ena, d, aclr, apre;
    output q;
    wire ena, d, aclr, apre;
    wire q;
    reg qi;
    assign #1 q = qi;
    always
        @(d or ena or apre or aclr)
            if (aclr)
                qi <= 0;
            else if (apre)
                qi <= 1;
            else begin
                if (ena)
                    qi <= d;
            end
    initial
        qi <= 1'b0;
endmodule
`endif
`ifdef RC_CDN_GENERIC_GATE
`else
module CDN_flop(clk, d, sena, aclr, apre, srl, srd, q);
    input clk, d, sena, aclr, apre, srl, srd;
    output q;
    wire clk, d, sena, aclr, apre, srl, srd;
    wire q;
    reg qi;
    assign #1 q = qi;
    always
        @(posedge clk or posedge apre or posedge aclr)
            if (aclr)
                qi <= 0;
            else if (apre)
                qi <= 1;
            else if (srl)
                qi <= srd;
            else begin
                if (sena)
                    qi <= d;
            end
    initial
        qi <= 1'b0;
endmodule
`endif
```

Note: Using this attribute is ignored in the structural flow.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

Related attribute: (hdl_arch) [hdl_preserve_unused_registers](#) on page 865

hdl_record_naming_style

`hdl_record_naming_style string`

Default: %s\[%s\]

Read-write [root](#) attribute. Specifies the format used to name individual bits of record variables in the RTL. The first %s is the variable name and the second %s is the field name. Set this attribute before using the [elaborate](#) command.

Note: This attribute is supported only in the RTL flow.

Related Information

[Naming Individual Bits of Array and Record Ports and Registers](#) in *Using Encounter RTL Compiler* for examples on how to use this attribute.

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_reg_naming_style

`hdl_reg_naming_style string`

Default: %s_reg%

Read-write [root](#) attribute. Specifies the format used to name flip-flop and latch instances inferred from scalar and vector variables in the RTL. The first %s is the variable name. If the variable is a vector, the second %s is the individual bit of the vector as specified by the [hdl_array_naming_style](#) attribute.

Note: This attribute is supported only in the RTL flow.

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

Related Information

Naming Individual Bits of Array and Record Ports and Registers in *Using Encounter RTL Compiler* for examples on how to use this attribute.

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_rename_cdn_flop_pins

```
hdl_rename_cdn_flop_pins list_of_pin_mappings
```

Read-write root attribute. Renames the specified pins of inferred CDN_flop components . Specify a list of pin mappings, where each mapping renames the given pin.

After elaboration and prior to mapping, RTL Compiler uses CDN_flop components to represent the flip-flops. The default CDN_flop pin names are:

Pin Name	Pin Function
aclr	asynchronous clear
apre	asynchronous preset
clk	clock
d	data input
q	output
sena	synchronous data enable
srd	synchronous set/reset data
srl	synchronous set/reset load

When different CDN_flop pin names are required (for example, an SDC constraints file may use different pin names), use this attribute to rename the CDN_flop pins.

Example

```
set_attribute hdl_rename_cdn_flop_pins {{q Q} {d D} {clk CK}}
```

This setting will allow RTL Compiler to interpret the following constraints correctly:

```
set_max_delay 3.4 -from [get_pins {datab_reg_reg[6]/CK}] \
-to [get_pins {multa_reg_reg[14]/D}]
set_min_delay 3.4 -from [get_pins {datab_reg_reg[6]/CK}] \
-to [get_pins {multa_reg_reg[14]/D}]
```

Related Information

Affects these commands: [elaborate](#)
[read_hdl](#)

hdl_sv_module_wrapper

```
hdl_sv_module_wrapper {false | true}
```

Default: false

Read-write root attribute. Controls whether the `write_sv_wrapper` command must generate a System Verilog module wrapper that connects the original I/O interfaces to the port names in the Verilog netlist.

Set this attribute to `true` before you elaborate the design to have the `write_sv_wrapper` command generate the module wrapper with the definitions of the complex types used.

Note: This attribute applies only to RTL using System Verilog interfaces. When interfaces are used, the I/O ports written by the `write_hdl` command in the Verilog netlist will by default not match the test bench.

Related Information

Affects this command: [elaborate](#)
[write_sv_wrapper](#)

hdl_sync_set_reset

```
hdl_sync_set_reset Tcl_list
```

Default: null

Read-write root attribute. Specifies that RTL Compiler implement the listed signals using synchronous set and reset pins on a flip-flop if that logic controls a synchronous assignment.

Note: This attribute is supported only in the RTL flow.

Note: This attribute does not affect the `synthesize` command, which may implement the set and reset logic using flip-flop data pins.

Example

```
rc:/ set_attr hdl_sync_set_reset {r1 r2} /
```

This command has the same effect as using the `sync_set_reset` pragma in the RTL. For example:

```
// cadence sync_set_reset "r1 r2"
```

For more information, see the RTL example and the corresponding schematic for `hdl_sync_set_reset` in the [HDL-Related Attributes](#) section of *HDL Modeling in Encounter RTL Compiler*.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_track_module_elab_memory_and_runtime

`hdl_track_module_elab_memory_and_runtime {false | true}`

Default: `false`

Read-write [root](#) attribute. Controls whether during elaboration the tool tracks the memory used (in MegaBytes) and run time taken (cpu-time in seconds) for elaboration of each module.

If you set this attribute to `true`, the run time and memory usage numbers will be printed (to standard output) for each module during elaboration. The numbers printed for each module indicate the time and memory needed to elaborate that module but the numbers do not include the time and memory taken to elaborate other modules instantiated by the module.

Set this attribute to `true` for designs that have a long elaboration time to identify the modules that are impacting the performance.

Note: This attribute is supported only in the RTL flow. The attribute will only print out the numbers for modules, which are read in and elaborated with the `read_hdl` command (the high level RTL flow) and will have no effect for modules which are read in with the `read_netlist` command (the structural elaboration flow).

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_unconnected_input_port_value

```
hdl_unconnected_input_port_value {none | 0 | 1 | x }
```

Default: none

Read-write root attribute. Connects each unconnected input port of a module or cell instantiation to the specified value. During an instantiation, using this attribute controls the value driving an input port of the child module that is undriven at the parent module.

In the RTL, either

- an (i) input port of the child module is missing in the instantiation statement
- an (ii) input port of the child module exists in the instantiation statement but it not given a signal, such as `u1 (.a(), ...);`

If you set the attribute to `none`, an unconnected input port of a child module remains unconnected. If you specify `x`, an unconnected port is driven by an `x` dont-care value. The specified value is case-insensitive. You can specify `x`, `X`, `none`, `None`, or `NONE`.



If the instantiation statement contains a signal with insufficient bit width, it looks like the child module has a partially unconnected input port. At the parent module, this becomes an undriven internal signal. In that case, you can control how this signal is synthesized through the `hdl_undriven_signal_value` attribute, as shown in the second example of that attribute.

Note: This attribute is supported only in the RTL flow.

Examples

- In the following example, the `c` input port of the `and3` sub-module is unconnected in all three instantiation statements.

```
module and3 (y, a, b, c);
    input [7:0] a, b, c;
    output [7:0] y;
    assign y = a & b & c;
endmodule

module test (x, y, z, a, b);
    input [7:0] a, b;
    output [7:0] x, y, z;
    and3 u1 (.y(x), .a(a), .b(b), .c());
    and3 u2 (.y(y), .a(a), .b(b));
    and3 u3 (z, a, b);
endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- ❑ If you specify 0, the post-elaboration netlist will look like the following example.

```
module and3(y, a, b, c);
  ...
endmodule
module test(x, y, z, a, b);
  input [7:0] a, b;
  output [7:0] x, y, z;
  wire [7:0] a, b;
  wire [7:0] x, y, z;
  and3 u1(.y (x), .a (a), .b (b), .c (8'b00000000));
  and3 u2(.y (y), .a (a), .b (b), .c (8'b00000000));
  and3 u3(z, a, b, 8'b00000000);
endmodule
```

- ❑ If you specify 1, the post-elaboration netlist will look like the following example:

```
module and3(y, a, b, c);
  ...
endmodule
module test(x, y, z, a, b);
  input [7:0] a, b;
  output [7:0] x, y, z;
  wire [7:0] a, b;
  wire [7:0] x, y, z;
  and3 u1(.y (x), .a (a), .b (b), .c (8'b11111111));
  and3 u2(.y (y), .a (a), .b (b), .c (8'b11111111));
  and3 u3(z, a, b, 8'b11111111);
endmodule
```

- ❑ If you specify x, the post-elaboration netlist will look like the following example:

```
module and3(y, a, b, c);
  ...
endmodule
module test(x, y, z, a, b);
  input [7:0] a, b;
  output [7:0] x, y, z;
  wire [7:0] a, b;
  wire [7:0] x, y, z;
  and3 u1(.y (x), .a (a), .b (b), .c (8'bxxxxxxxx));
  and3 u2(.y (y), .a (a), .b (b), .c (8'bxxxxxxxx));
  and3 u3(z, a, b, 8'bxxxxxxxx);
endmodule
```

- ❑ If you specify none, the post-elaboration netlist will look like the following example:

```
module and3(y, a, b, c);
  ...
endmodule
module test(x, y, z, a, b);
  input [7:0] a, b;
  output [7:0] x, y, z;
  wire [7:0] a, b;
  wire [7:0] x, y, z;
  wire n_18, n_19, n_20, n_21, n_22, n_23, n_24, n_25;
  wire n_26, n_27, n_28, n_29, n_30, n_31, n_32, n_33;
  wire n_34, n_35, n_36, n_37, n_38, n_39, n_40, n_41;
  and3 u1(.y (x), .a (a), .b (b), .c ({n_25, n_24, n_23, n_22, n_21,
    n_20, n_19, n_18}));
  and3 u2(.y (y), .a (a), .b (b), .c ({n_33, n_32, n_31, n_30, n_29,
    n_28, n_27, n_26}));
  and3 u3(z, a, b, {n_41, n_40, n_39, n_38, n_37, n_36, n_35, n_34});
endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- Consider the following hierarchical RTL description.

```
module lower (x, y, z, a, b, c);
    input [2:0] a, b, c;
    output [2:0] x, y, z;
    wire [2:0] d;
    assign x = a | b;
    assign y = a | c;
    assign z = a | d;
endmodule

module top (x, y, z, a, b);
    input [3:0] a;
    input [1:0] b;
    output [3:0] x, y, z;
    lower u1 (.x(x), .y(y), .z(z), .a(a), .b(b));
endmodule
```

This example has the following undriven signals and ports:

- The following output ports are undriven: `top/x[3]`, `top/y[3]`, `top/z[3]`
In the top-level module, the bit width of signals `x`, `y`, `z` is larger than the bit width of their corresponding ports in the lower-level module. The `hdl_undriven_output_port_value` attribute controls how these ports are synthesized.
- The following input port is completely unconnected: `lower/c`
In the top-level module, the instantiation statement does not provide anything to drive the `c` input port of the lower-level module. The `hdl_unconnected_input_port_value` attribute controls how this port is synthesized.
- The following input port is partially unconnected: `lower/b[2]`
The bit width of signal `b` in the top-level module is smaller than the bit width of its corresponding port `b` in the lower-level module. The `hdl_undriven_signal_value` attribute controls how this port is synthesized.
- The following internal signal is undriven: `lower/d`
In the lower-level module, all bits of signal `d` are undriven. The `hdl_undriven_signal_value` attribute controls how this signal is synthesized.

Assume the following script:

```
set_attr library tutorial.lbr /
set_attr hdl_undriven_output_port_value 1 /
set_attr hdl_unconnected_input_port_value 0 /
set_attr hdl_undriven_signal_value 1 /
read_hdl example.v
elaborate
write_hdl
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

The post-elaboration netlist will be similar to:

```
module lower (x, y, z, a, b, c);
    input [2:0] a, b, c;
    output [2:0] x, y, z;
    or g1 (x[0], a[0], b[0]);
    or g2 (x[1], a[1], b[1]);
    or g3 (x[2], a[2], b[2]);
    or g4 (y[0], a[0], c[0]);
    or g5 (y[1], a[1], c[1]);
    or g6 (y[2], a[2], c[2]);
    assign z[0] = 1'b1;
    assign z[1] = 1'b1;
    assign z[2] = 1'b1;
endmodule
module top (x, y, z, a, b);
    input [3:0] a;
    input [1:0] b;
    output [3:0] x, y, z;
    lower u1 (.x(x[2:0]), .y(y[2:0]), .z(z[2:0]),
              .a(a[2:0]), .b({1'b1, b}), .c(3'b000));
    assign x[3] = 1'b1;
    assign y[3] = 1'b1;
    assign z[3] = 1'b1;
endmodule
```

All bits of signal *a* of the lower-level module are undriven and are assigned a constant 1, according to the setting of the `hdl_undriven_signal_value` attribute. Consequently, all the OR gates driving signal *z* in the lower-level module produce a constant 1.

Input port *b*[2] of the lower-level module is *partially* undriven and is assigned a constant 1, according to the setting of the `hdl_undriven_signal_value` attribute.

Input port *c* of the lower-level module is *entirely* unconnected and all bits of this port are assigned a constant 0, according to the setting of the `hdl_unconnected_input_port_value` attribute.

Output ports *x*[3], *y*[3], and *z*[3] of the top-level module are undriven and are assigned a constant 1, according to setting of the `hdl_undriven_output_port_value` attribute.

Assume the following script:

```
set_attr library tutorial.lbr /
set_attr hdl_undriven_output_port_value none /; # default
set_attr hdl_unconnected_input_port_value none /; # default
set_attr hdl_undriven_signal_value none /; # default
read_hdl example.v
elaborate
write_hdl
```

The post-elaboration netlist will be similar to:

```
module lower (x, y, z, a, b, c);
    input [2:0] a, b, c;
    output [2:0] x, y, z;
    wire \d[0], \d[1], \d[2];
    or g1 (x[0], a[0], b[0]);
    or g2 (x[1], a[1], b[1]);
    or g3 (x[2], a[2], b[2]);
    or g4 (y[0], a[0], c[0]);
    or g5 (y[1], a[1], c[1]);
    or g6 (y[2], a[2], c[2]);
    or g7 (z[0], a[0], \d[0]);
    or g8 (z[1], a[1], \d[1]);
    or g9 (z[2], a[2], \d[2])
endmodule

module top (x, y, z, a, b);
    input [3:0] a;
    input [1:0] b;
    output [3:0] x, y, z;
    wire n_7, n_8, n_9, n_10;
    lower u1 (.x(x[2:0]), .y(y[2:0]), .z(z[2:0]),
              .a(a[2:0]), .b({n_7, b}), .c({n_10, n_9, n_8}));
endmodule
```

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_undriven_output_port_value

hdl_undriven_output_port_value {none | 0 | 1 | x}

Default: none

Read-write [root](#) attribute. Connects each undriven output port, including undriven bits of a bus, to the specified value. In a module, using this attribute controls the value driving an undriven output port.

If you specify none, the undriven output port remains undriven. If you specify x, the undriven output port is driven by a x dont-care value. The specified value is case-insensitive. You can specify x, X, none, None, or NONE.

Note: This attribute does not affect bidirectional ports and is only supported in the RTL flow.

Examples

- In the following example, the x output port is undriven.

```
module test (x, y, a, b);
    input [7:0] a, b;
    output [7:0] x, y;
    assign y = a & b;
endmodule
```

If you specify none, then the post-elaboration netlist will look like the following example:

```
module test (x, y, a, b);
    input [3:0] a, b;
    output [3:0] x, y;
    and g1 (y[0], a[0], b[0]);
    and g2 (y[1], a[1], b[1]);
    and g3 (y[2], a[2], b[2]);
    and g4 (y[3], a[3], b[3]);
endmodule
```

If you specify 0, the post-elaboration netlist will look like the following example:

```
module test (x, y, a, b);
    input [3:0] a, b;
    output [3:0] x, y;
    assign x[0] = 1'b0;
    assign x[1] = 1'b0;
    assign x[2] = 1'b0;
    assign x[3] = 1'b0;
    and g1 (y[0],a[0], b[0]);
    and g2 (y[1],a[1], b[1]);
    and g3 (y[2],a[2], b[2]);
    and g4 (y[3],a[3], b[3]);
endmodule
```

If you specify 1, the post-elaboration netlist will look like the following example:

```
module test (x, y, a, b);
    input [3:0] a, b;
    output [3:0] x, y;
    assign x[0] = 1'b1;
    assign x[1] = 1'b1;
    assign x[2] = 1'b1;
    assign x[3] = 1'b1;
    and g1 (y[0],a[0], b[0]);
    and g2 (y[1],a[1], b[1]);
    and g3 (y[2],a[2], b[2]);
    and g4 (y[3],a[3], b[3]);
endmodule
```

If you specify X, the post-elaboration netlist will look like the following example:

```
module test (x, y, a, b);
    input [3:0] a, b;
    output [3:0] x, y;
    assign x[0] = 1'bx;
    assign x[1] = 1'bx;
    assign x[2] = 1'bx;
    assign x[3] = 1'bx;
    and g1 (y[0],a[0], b[0]);
    and g2 (y[1],a[1], b[1]);
    and g3 (y[2],a[2], b[2]);
    and g4 (y[3],a[3], b[3]);
endmodule
```

- For a hierarchical example, refer to [Examples of the hdl_unconnected_input_port_value attribute](#).

Related Information

Affects these commands:

[elaborate](#)
[read_hdl](#)

hdl_undriven_signal_value

`hdl_undriven_signal_value {none | 0 | 1 | x}`

Default: none

Read-write [root](#) attribute. Connects each undriven signal, including undriven bits of a bus, to the specified value. In a module, using this attribute controls the value driving an undriven internal signal (wire or register).

If you set the attribute value to `none`, then an undriven signal remains undriven. If you specify the `x` value, then an undriven signal is driven by an `x` dont-care value. The specified value is case-insensitive. You can specify `x`, `X`, `none`, `None`, or `NONE`.

Note: This attribute is supported only in the RTL flow.

Examples

- In the following example, the `wc` signal is undriven:

```
module test (y, a, b);
    input [7:0] a, b;
    output [7:0] y;
    wire [7:0] wa, wb, wc;
    assign wa = a;
    assign wb = b;
    assign y = wa & wb & wc;
endmodule
```

- If you set the attribute to 0, the post-elaboration netlist will look like:

```
module test (y, a, b);
    input [3:0] a, b;
    output [3:0] y;
    assign y[0] = 1'b0;
    assign y[1] = 1'b0;
    assign y[2] = 1'b0;
    assign y[3] = 1'b0;
endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- If you set the attribute to 1, the post-elaboration netlist will look like:

```
module test (y, a, b);
    input [3:0] a, b;
    output [3:0] y;
    and g1 (y[0], a[0], b[0]);
    and g6 (y[1], a[1], b[1]);
    and g3 (y[2], a[2], b[2]);
    and g4 (y[3], a[3], b[3]);
endmodule
```

- If you set the attribute to none, the post-elaboration netlist will look like:

```
module test (y, a, b);
    input [3:0] a, b;
    output [3:0] y;
    wire n_2, n_3, n_4, n_5, \wc[0] ,\wc[1] , \wc[2] , \wc[3] ;
    and g1 (n_2, a[0], b[0]);
    and g7(y[0], n_2, \wc[0]);
    and g6 (n_3, a[1], b[1]);
    and g2(y[1], n_3, \wc[1]);
    and g3 (n_4, a[2],b[2]);
    and g8(y[2], n_4, \wc[2]);
    and g4 (n_5, a[3], b[3]);
    and g9(y[3], n_5, \wc[3]);
endmodule
```

- In the following example one bit of input port a is undriven.

```
module lower (a, b);
    input [1:0] a;
    output [1:0] b;
    assign b = a ;
endmodule
module top (inp1, out);
    input inp1;
    output [1:0] out;
    lower u0(.a(inp1), .b(out));
endmodule
```

- If you set the attribute to none, the post-elaboration netlist will look similar to:

```
module lower(a, b);
    input [1:0] a;
    output [1:0] b;
    assign b[0] = a[0];
    assign b[1] = a[1];
endmodule
module top(inp1, out);
    input inp1;
    output [1:0] out;
    wire n_3;
    lower u0(.a ({n_3, inp1}), .b (out));
endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- If you set the attribute to 1, the post-elaboration netlist will look similar to:

```
module lower(a, b);
    input [1:0] a;
    output [1:0] b;
    assign b[0] = a[0];
    assign b[1] = a[1];
endmodule
module top(inp1, out);
    input inp1;
    output [1:0] out;
    lower u0(.a ({1'b1, inp1}), .b (out));
endmodule
```

- If you set the attribute to x, the post-elaboration netlist will look similar to:

```
module lower(a, b);
    input [1:0] a;
    output [1:0] b;
    assign b[0] = a[0];
    assign b[1] = a[1];
endmodule
module top(inp1, out);
    input inp1;
    output [1:0] out;
    wire _X_;
    lower u0(.a ({_X_, inp1}), .b (out));
    CDN_dc logicX_inst(.cf (1'b0), .dcf (1'b1), .z (_X_));
endmodule
```

- For a hierarchical example, refer to [Examples](#) of the `hdl_unconnected_input_port_value` attribute.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_use_block_prefix

`hdl_use_block_prefix {true | false}`

Default: true

Read-write `root` attribute. Controls whether block statement labels should be used as a prefix for instance names specified in the input RTL description.

Related Information

Affects this command: [elaborate](#)

Related attributes: [hdl_generate_index_style](#) on page 739

[hdl_generate_separator](#) on page 742

[hdl_use_if_generate_prefix](#) on page 779

hdl_use_cw_first

`hdl_use_cw_first {false | true}`

Default: false

Read-write `root` attribute. This attribute is applicable in those cases in which the RTL code has a user defined module that has the identical name of a ChipWare component. If this attribute is set to true, RTL Compiler will use ChipWare components to instantiate the functionality in the design as opposed to the user defined modules (Verilog) or entities (VHDL).

hdl_use_default_parameter_values_in_design_name

`hdl_use_default_parameter_values_in_design_name {false | true}`

Default: false

Read-write `root` attribute. Setting this attribute to `true` uses all the parameter values in the parameterized module name for a top-level design or a design specified with the `elaborate` command.

Note: This attribute is supported only in the RTL flow.

Examples

The following example renames a top-level module or design specified with the `elaborate` command to use all parameter values in a parameterized module name.

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

The following is a Verilog example with a parameterized top-module named `top` and a parameterized sub-module named `comp`:

```
module top(a, b, c);
    parameter w = 4;
    input [2:0] a, b ;
    output [2:0] c;
    comp #2 U1(C, a, b);
endmodule // module top

module comp (q,d1,d2);
    parameter p = 5;
    parameter w = 3;
    output [2:0] q;
    input [2:0] d1, d2;
    foo u1(q, d1,d2);
endmodule // module comp
```

Setting the following attributes:

```
rc:/> set_attribute hdl_use_default_parameter_values_in_name false // default
rc:/> set_attribute hdl_use_default_parameter_values_in_design_name true
```

Then using the `elaborate` command results in renaming the top-level design `top` to `top_w4`.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_use_default_parameter_values_in_name

`hdl_use_default_parameter_values_in_name {false | true}`

Default: `false`

Read-write `root` attribute. Setting this attribute to `false` shortens the name of the parameterized module by using only the parameter values specified at instantiation, while setting this attribute to `true` uses all the available parameters in the module name.

Note: This attribute is supported only in the RTL flow.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_use_for_generate_prefix

```
hdl_use_for_generate_prefix {true | false}
```

Default: true

Read-write root attribute: Controls whether the block label of a `for`-generate statement will be used with the `for`-generate index values as prefix of the name of an instance instantiated within the generate construct.

- If this attribute is set to `false`, the `for`-generate index values are appended to the name of instances instantiated within the generate construct.
- If this attribute is set to `true`, instance naming follows the naming style defined in the Verilog-2005 LRM (in Section 12.4.2 and 12.4.3 of IEEE Std 1364-2005).

You must specify this attribute before you elaborate the design.

Examples

For the following examples consider the following VHDL code:

```
L1: FOR i IN 2 TO 4 generate
  u1 : sub port map(q => q(i), d1 => d1(i), d2 => d2(i));
end generate;
```

- When you set the `hdl_use_for_generate_prefix` attribute to `true`, RC generates the following:

```
sub \L1[2].u1 (.q (q[2]), .d1 (d1[2]), .d2 (d2[2]));
sub \L1[3].u1 (.q (q[3]), .d1 (d1[3]), .d2 (d2[3]));
sub \L1[4].u1 (.q (q[4]), .d1 (d1[4]), .d2 (d2[4]));
```

- When you set the `hdl_use_for_generate_prefix` attribute to `false`, RC generates the following:

```
sub \u1.[2] (.q (q[2]), .d1 (d1[2]), .d2 (d2[2]));
sub \u1.[3] (.q (q[3]), .d1 (d1[3]), .d2 (d2[3]));
sub \u1.[4] (.q (q[4]), .d1 (d1[4]), .d2 (d2[4]));
```

Related Information

Affects this command: [elaborate](#)

Related attributes: [hdl_generate_index_style](#) on page 739
[hdl_generate_separator](#) on page 742

hdl_use_if_generate_prefix

```
hdl_use_if_generate_prefix {true | false}
```

Default: true

Read-write root attribute: Controls whether the block label of an `if`-generate statement will be used as prefix of the instance name of an instance instantiated within the generate construct.

- If this attribute is set to `false`, this part of instance naming is backward-compatible with prior releases.
- If this attribute is set to `true`, this part of instance naming follows the naming style defined in the Verilog-2005 LRM (in Section 12.4.2 of IEEE Std 1364-2005).

In VHDL, a generate statement must have a label. In Verilog, the block of a generate statement can be unnamed. If without a label, it is given a name based on the rules defined in the Verilog-2005 LRM (in Section 12.4.3 of IEEE Std 1364-2005).

This attribute does not affect a Verilog `case`-generate statement, with which the block label is always used when composing instance names.

You must specify this attribute before you elaborate the design.

Examples

For the following examples consider the following RTL code:

```
library ieee;
use ieee.std_logic_1164.all;
entity sub is
    port(q : out std_logic; d1, d2 : in std_logic);
end;

architecture a of sub is
begin
    q <= d1 xor d2;
end;

library ieee;
use ieee.std_logic_1164.all;
entity test_02 is
    port(q : out std_logic; d1, d2 : in std_logic);
end;

architecture a of test_02 is
begin
    f1: if 1 = 1 generate
        u1 : entity work.sub
            port map(q => q, d1 => d1, d2 => d2);
    end generate;
end;
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- Using the following script:

```
set_attr library tutorial.lbr
set_attribute hdl_use_if_generate_prefix false /
read_hdl test.vhdl -vhdl
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module sub(q, d1, d2);
    input d1, d2;
    output q;
    wire d1, d2;
    wire q;
    xor g1 (q, d1, d2);
endmodule

module test_02(q, d1, d2);
    input d1, d2;
    output q;
    wire d1, d2;
    wire q;
    sub ul(.q (q), .d1 (d1), .d2 (d2));
endmodule
```

- Using the following script:

```
set_attr library tutorial.lbr
set_attribute hdl_use_if_generate_prefix true /
read_hdl test.vhdl -vhdl
elaborate
write_hdl
```

After elaboration, the netlist will look like:

```
module sub(q, d1, d2);
    input d1, d2;
    output q;
    wire d1, d2;
    wire q;
    xor g1 (q, d1, d2);
endmodule

module test_02(q, d1, d2);
    input d1, d2;
    output q;
    wire d1, d2;
    wire q;
    sub \f1.ul (.q (q), .d1 (d1), .d2 (d2));
endmodule
```

Related Information

Affects this command: [elaborate](#)

Related attributes: [hdl_generate_separator](#) on page 742

[hdl_use_block_prefix](#) on page 775

hdl_use_parameterized_module_by_name

```
hdl_use_parameterized_module_by_name {false | true}
```

Default: false

Read-write root attribute. Controls whether module linking uses the parameterized module name. During elaboration, RTL Compiler tries to identify a child module for every instantiation statement in the RTL code.

By default, RTL Compiler does not look for a parameterized module name to resolve the instantiation, but instead looks for a module with the exact same name as given in the instantiation statement.

If you set this attribute to `true`, RTL Compiler looks for the parameterized module names during module linking.

- If the instantiation statement provides the parameter name(s), the child module name is parameterized using the `_%s%d` style as illustrated in the first example.
- If the instantiation statement does not provide a parameter name, the child module name is parameterized using the `_%d` style as illustrated in the second example.
- Child modules with a parameter in their body are not considered during linking as illustrated in the third example.

At the module linking step during elaboration, the `hdl_parameter_naming_style` attribute setting does not affect how the instantiated module name is parameterized.

Note: This attribute is supported only in the RTL flow.

Examples

- The following RTL code of the top-level module has an instantiation statement that contains the parameter name `p` and references the parameter value 16.

```
module top (y, a, b);
    parameter w = 16;
    input [w-1:0] a, b;
    output [w-1:0] y;
    btm #(p(w)) u1 (y, a, b);
endmodule
```

If the attribute is set to `false`, RTL Compiler links this instance to a Verilog module (or a VHDL entity) whose name is `btm`. If set to `true`, RTL Compiler links the instance to a module or entity whose name is `btm_p16`. The parameterized module name includes both the parameter name `p` and the parameter value 16.

The linked child module might be similar to:

```
module btm_p16 (z, c, d);
    input [15:0] c, d;
    output [15:0] z;
    ...
endmodule
```

- The following RTL code of the top-level module has an instantiation statement that references a parameter value (16) but has no parameter name.

```
module top (y, a, b);
    parameter w = 16;
    input [w-1:0] a, b;
    output [w-1:0] y;
    btm #(w) u1 (y, a, b);
endmodule
```

If the attribute is set to `false`, RTL Compiler links this instance of instantiation to a Verilog module (or a VHDL entity) whose name is `btm`. If set to `true`, RTL Compiler links the instance to a module or entity whose name is `btm_16`. This parameterized module name does not include the parameter name, because the tool cannot derive that from the instantiation statement. The linked child module might be similar to:

```
module btm_16 (z, c, d);
    input [15:0] c, d;
    output [15:0] z;
    ...
endmodule
```

- The following RTL code of the top-level module has an instantiation statement that references a parameter value (16) and has no parameter name. Even when the attribute is set to `true`, RTL Compiler will not link instance `u0` to module `btm_16` because module `btm_16` has a parameter statement.

```
module top (...);
    btm #(16) u0 (...);
    ...
endmodule

module btm_16 (...);
    parameter p = 32;
    ...
endmodule
```

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_use_port_default_value

`hdl_use_port_default_value {true | false}`

Default: true

Read-write root attribute. When set to `true`, RTL Compiler honors default initial values of input ports in a VHDL component declaration or entity declaration.

Note: This attribute is supported only in the RTL flow.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

hdl_use_techelt_first

`hdl_use_techelt_first {false | true}`

Default: false

Read-write root attribute. When set to `true`, RTL Compiler binds instances of design `M` to component `M` from the technology library. If there is no `M` component in the technology library, RTL Compiler binds instances of `M` to a user-defined module `M`, provided this module exists. If there is no `M` user-defined module, instances of `M` are assumed to be unresolved references (blackboxes).

Note: This attribute is supported in the RTL flow and the structural flow.

Related Information

[Modeling Logic Abstracts in HDL Modeling in Encounter RTL Compiler](#)

Affects these commands: [elaborate](#)
 [read_hdl](#)
 [read_netlist](#)

ignore_preserve_in_tiecell_insertion

```
ignore_preserve_in_tiecell_insertion {false | true}
```

Default: false

Read-write root attribute. Ignores all preserve settings when inserting tie-cells during synthesis. During synthesis, the insertion of tie cells is controlled by the use_tiehilo_for_const root attribute.

Note: This attribute has no effect when the insertion of tie cells is requested using the insert_tiehilo_cells command.

Related Information

Affected by this attribute: [use_tiehilo_for_const](#) on page 830

Related attribute: [iopt_allow_tiecell_with_inversion](#) on page 785

incr_retime

```
incr_retime {false | true}
```

Default: false

Read-write root attribute. Controls incremental retiming during incremental optimization. Incremental retiming tries to improve the most critical path in the design by retiming the startpoint or endpoint registers. When this path is improved, the next most critical path is improved till no further improvement is possible on the critical paths.

Use the dont_retime instance attribute to control the scope of the incremental retiming.

The verification flow for incremental retiming is similar to the regular retiming verification flow. After incremental optimization, use the write_hdl -lec command to generate a LEC-friendly netlist that has retiming state points for any registers that are incrementally retimed. Next, use the write_do_lec command to create the required dofile to perform formal verification.

Related Information

Recommended Flow in *Interfacing between Encounter RTL Compiler and Encounter Conformal*.

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [dont_retime on page 842](#)

Related attributes (design) [retime](#) on page 839

 (subdesign) [retime](#) on page 910

[retime verification flow](#) on page 820

iopt_allow_tiecell_with_inversion

`iopt_allow_tiecell_with_inversion {false | true}`

Default: false

Read-write [root](#) attribute. Controls the use of a tie cell with an inverter if either the tie high or tie low cell is not found. By default, a tie cell with inverter will not be used.

Related Information

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [use_tiehilo_for_const on page 830](#)

Related attribute: [ignore_preserve_in_tiecell_insertion](#) on page 784

iop_t_enable_floating_output_check

```
iop_t_enable_floating_output_check {false | true}
```

Default: false

Read-write root attribute. Controls the use of multiple output gates with floating outputs during incremental optimization. By default, multiple output gates with floating outputs can be used.

Setting the attribute to true has the following effects:

- During sizing, a gate can only be replaced with a higher or lower driver with the same number of outputs.
- During gate composition, a gate will not be replaced with a multiple-output gate with floating outputs, that is, it can be replaced with a single output gate.

Related Information

Affects this command: [synthesize -incremental](#)

iop_t_enable_parallelization

```
iop_t_enable_parallelization {true | false}
```

Default: true

Read-write root attribute. Enables parallel incremental optimization when super-thread servers are set.

Related Information

Affects this command: [synthesize -incremental](#)

iopt_force_constant_removal

`iopt_force_constant_removal {false | true}`

Default: false

Read-write root attribute. Forces the optimization of gates with inputs tied to constants independent of the QoR.

Related Information

Affects this command: [synthesize -incremental](#)

iopt_remap_avoided_cells

`iopt_remap_avoided_cells {false | true}`

Default: false

Read-write root attribute. Controls whether to replace (remap) unpreserved avoided cells in the mapped netlist. For those cells that cannot be remapped, the tool will issue messages

Related Information

Affects this command: [synthesize -incremental](#)

iopt_sequential_duplication

`iopt_sequential_duplication {false | true}`

Default: false

Read-write root attribute. Controls duplication of sequential (flops) elements during incremental optimization. It duplicates the flops on the critical timing path to improve the timing of this path.

Related Information

Affects this command: [synthesize -incremental](#)

iopt_sequential_resynthesis

`iopt_sequential_resynthesis {true | false}`

Default: true

Read-write root attribute. Controls sequential resynthesis during incremental optimization.

Related Information

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [iopt_sequential_resynthesis_min_effort](#) on page 788

iopt_sequential_resynthesis_min_effort

`iопt_sequential_resynthesis_min_effort {high | medium | low}`

Default: high

Read-write root attribute. Specifies the minimum effort required for incremental optimization to perform sequential resynthesis.

Related Information

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [iopt_sequential_resynthesis](#) on page 788

iopt_temp_directory

`iopt_temp_directory string`

Default: .

Read-write root attribute. Specifies the directory where parallel incremental optimization can write temporary files. By default, the files are written to the current directory from where RTL Compiler was invoked. The directory must be writable.

Related Information

Affects this command: [synthesize -incremental](#)

iopt_ultra_optimization

`iopt_ultra_optimization {false | true}`

Default: false

Read-write root attribute. Controls how rigorously incremental optimization should work to achieve the best QoR (timing and area).

Due to the iterative nature of the algorithms, this attribute will have an impact on the run-time.

Related Information

Affects this command: [synthesize -incremental](#)

Ibr_seq_in_out_phase_opto

```
Ibr_seq_in_out_phase_opto {false | true}
```

Default: false

Read-write root attribute. Enables the tool to perform the following transformations during synthesis if they result in a QoR improvement.

- If the original netlist has no inverters at the `d` input and the `q` output, the tool can add inverters at this input and this output during mapping.
A simple flip-flop can be converted to a flop with inversion at the `d` input and the `q` output
- If the original netlist has an inverter at the `d` input and the `q` output, the tool can remove both inverters from this input and this output during mapping.
A flop with inversion at the `d` input and the `q` output can be converted to a simple flop by removing the inversions.
- If the original netlist has an inverter at the `d` input, the tool can move the inverter from the `d` input to the `q` output of the sequential cell.

This can be referred to as propagating an inverter through the sequential cell, or also bubble pushing.

Note: Verification of these transforms requires compute-expensive phase mapping in Conformal. The affected key points will be reported as inverted equivalents.

- Inverting the `d` input and `q` output also inverts the sense of the preset (set) and clear (reset) signals (both synchronous and asynchronous). A clear sequential libcell with inverted `d` input and `q` output becomes a preset libcell. And a preset libcell becomes a clear one. The preset/clear swapping can only happen if we invert the `d` and `q` phases.

This is useful when the RTL functionality requires a preset function but the technology library has only a reset flop (or vice-versa).

Set this attribute before reading the library.

By default, these operations are not performed because they can impact verification of the design.

Example

```
module invert_d_q (q, q1, d, clk, rstn, psth);
    input clk, rstn, psth, d;  output q, q1;  reg q, q1;
    always @ (posedge clk)
        begin
            if (!rstn) q = 1'b0;
            else q = d;
            if (!psth) q1 = 1'b1;
            else q1 = d;
        end
    endmodule
```

- By default this RTL description is synthesized into a reset flop and a preset flop.
- If the attribute is set to `true` and if the library has only reset flops, the preset flop will be mapped to a reset flop.
- If the attribute is set to `true` and if the library has only preset flops, the reset flop will be mapped to a preset flop.

Related Information

[Translating Phase Mapping in Interfacing between Encounter RTL Compiler and Encounter Conformal](#)

Affects these commands: [synthesize -to_mapped](#)
 [write do lec](#)

map_drc_first

`map_drc_first {false | true}`

Default: `false`

Read-write [root](#) attribute. Specifies whether to use the design rule constraints as a preferred costing factor for selecting cells from the library during mapping. These constraints are considered with the other costs, such as area, timing and power, but by default they are not given the highest weight.

Related Information

Affects this command: [synthesize -to_mapped](#)
Related attribute: [drc first](#) on page 529

map_latch_allow_async_decomp

```
map_latch_allow_async_decomp {false | true}
```

Default: false

Read-write root attribute. If set to true, RTL Compiler will implement asynchronous control logic outside the latch using combinational gates if the library does not have latch libcells with the required asynchronous functionality. For example, a reset latch could be implemented using a simple latch libcell with combinational logic driving the `d` and `ena` inputs.

Related Information

Affects this command: [synthesize -to_mapped](#)

map_respect_rtl_clk_phase

```
map_respect_rtl_clk_phase {false | true}
```

Default: false

Read-write root attribute. If set to true, RTL Compiler will use flip-flop libcells with the same clock edge, as specified in the RTL. If the library has no libcells with the required clock phase, RTL Compiler will use any available libcells.

By default, RTL Compiler will use flip-flop libcells with a certain clock phase that avoid adding an inverter for the clock.

Example

Consider the following module in RTL.

```
module mix_edge ( clk, d0, q0, q1, q0_inv, q1_inv, d1, d2,d3 );
input clk, d0, d1, d2, d3;
output q0, q1, q0_inv, q1_inv;
reg q0, q1, q0_inv, q1_inv;
wire clkn;

assign clkn = ~clk;
always @(negedge clk)
  q0 = d0;
always @(posedge clk)
  q1 = d1;
always @(negedge clkn)
  q0_inv = d2;
always @(posedge clkn)
  q1_inv = d3;
endmodule
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

If you use the default setting of false for `map_respect_rtl_clk_phase`, RTL Compiler will use the following mappings:

q0_reg will be mapped to FFRNQ1, a negative-edge triggered flip-flop
q1_reg will be mapped to FFPQ1, a positive-edge triggered flip-flop
q0_inv_reg will be mapped to FFPQ1, a positive-edge triggered flip-flop
q1_inv_reg will be mapped to FFRNQ1, a negative-edge triggered flip-flop

If you set `map_respect_rtl_clk_phase` to `true`, RTL Compiler uses the following mappings:

`q0_reg` will be mapped to `FFRNQ1`, a negative-edge triggered flip-flop
`q1_reg` will be mapped to `FFPQ1`, a positive-edge triggered flip-flop
`q0_inv_reg` will be mapped to `FFRNQ1`, a negative-edge triggered flip-flop + inverter
`q1_inv_reg` will be mapped to `FFPQ1`, a positive-edge triggered flip-flop + inverter

Related Information

Affects this command: **synthesize -to mapped**

map_to_master_slave_lssd

`map_to_master_slave_lssd {false | true}`

Default: false

Read-write `root` attribute. Specifies to recognize master-slave flip-flops as valid sequential cells and not as timing models during library parsing and mapping. You must set this attribute before reading the libraries. The slave clock, identified by the Liberty `clocked_on_also` attribute in the ff group of the library cell, will be used as the main (triggering) clock pin of the cell.

Related Information

Scan Cell Requirements in the *Library Guide* for *Encounter RTL Compiler*

Using Special Master-Slave LSSD Flip-Flops in Design for Test in Encounter RTL Compiler

Affects this command: **synthesize**

Affects these attributes: (port) [lssd_master_clock](#) on page 890
(support) [lssd_master_clock](#) on page 914

map_to_multiple_output_gates

```
map_to_multiple_output_gates {true | false}
```

Default: true

Read-write root attribute. When the value is `true` (default), RTL Compiler can map to more complex multi-output cells beyond half and full adders during incremental optimization.

Related Information

Affects this command: [synthesize -incremental](#)

merge_combinational_hier_instances

```
merge_combinational_hier_instances {true | false}
```

Default: true

Read-write root attribute. Allow merging of combinational hierarchical instances.

Related Information

Affects these commands: [synthesize -to_generic](#)
 [synthesize -to_mapped](#)

Affects this attribute: [merge_combinational_hier_instance](#) on page 849

merge_multibit_power_area_based

```
merge_multibit_power_area_based {true | false}
```

Default: true

Read-write root attribute. Merges single bit instances into multibit instances and chooses the multibit library cell based on power and area considerations, while ignoring the timing impact. This is useful for increasing multibit coverage and improving the run time, but might negatively impact the QoR, especially in timing-critical designs.

Examples

Assume two single bit instances `reg1` and `reg2` can be merged into a multibit instance. The library has three suitable multibit cells: `dual1`, `dual2`, and `dual3`.

- Consider only `merge_multibit_power_area_based` enabled:

```
set_attribute merge_multibit_power_area_based true /  
set_attribute force_merge_combos_into_multibit_cells false /  
set_attribute force_merge_seqs_into_multibit_cells false /
```

In this case, the tool computes the power and area cost savings for each of the possible candidates. Only when the power and area can be improved will the merging be applied and the best candidate will be chosen.

- Consider `merge_multibit_power_area_based`, `force_merge_combos_into_multibit_cells`, and `force_merge_seqs_into_multibit_cells` enabled:

```
set_attribute merge_multibit_power_area_based true /  
set_attribute force_merge_combos_into_multibit_cells true /  
set_attribute force_merge_seqs_into_multibit_cells true /
```

In this case, the tool computes the power and area costs for each of the possible candidates. The best candidate for multibit cell inferencing will be chosen irrespective whether it improves the QoR.

- Consider `merge_multibit_power_area_based` disabled, and `force_merge_combos_into_multibit_cells`, and `force_merge_seqs_into_multibit_cells` enabled:

```
set_attribute merge_multibit_power_area_based false /  
set_attribute force_merge_combos_into_multibit_cells true /  
set_attribute force_merge_seqs_into_multibit_cells true /
```

In this case, the tool considers the timing, power, and area impact for each of the possible candidates. The best candidate for multibit cell inferencing will be chosen irrespective whether it improves the QoR.

Related Information

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

Affects this command: [synthesize -incremental](#)

Related attributes: [dont infer multibit](#) on page 841

[force merge combos into multibit cells](#) on page 718

[force merge seqs into multibit cells](#) on page 719

[infer multibit](#) on page 844

minimize_uniquify

```
minimize_uniquify {false | true}
```

Default: false

Read-write root attribute. Controls uniquification of all subdesigns in the design.

By default, the tool uniquifies multiple instantiations with different context (timing, constants, and so on) to deliver the best QoR.

Set this attribute to true to limit the scenarios of uniquification of multiply-instantiated subdesigns and potentially improve runtime with a trade-off against QoR. This will not necessarily prevent uniquification from algorithmic considerations of the design topology.

Related Information

Affects this command: [synthesize -incr](#)

Affects this attribute: (subdesign) [minimize_uniquify](#) on page 906

multibit_allow_async_phase_map

```
multibit_allow_async_phase_map {true | false}
```

Default: true

Read-write root attribute. Controls whether asynchronous pins can be interchanged during multibit cell inferencing. By default, asynchronous pin phase mapping will be enabled.

Examples

Consider the following netlist after mapping and before incremental optimization:

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    DFSF cnt_0_reg[0] (.SDN (resetn), .CP (clk), .D(in[0]), .Q (out[0]));
    DFCF cnt_0_reg[1] (.CDN (resetn), .CP (clk), .D(in[1]), .Q (out[1]));
    DFCF cnt_0_reg[2] (.CDN (resetn), .CP (clk), .D(in[2]), .Q (out[2]));
    DFSF cnt_0_reg[3] (.SDN (resetn), .CP (clk), .D(in[3]), .Q (out[3]));
endmodule
```

The netlist contains two clear sequential libcells and two preset sequential libcells.

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- When the `multibit_allow_async_phase_map` attribute is set to `true` (default), you allow the tool to map clear sequential cells to a preset sequential cells while inverting the D input and Q output if this allows multibit inferencing. For the previous netlist, that means that cells `DFSF` and `DFCF` can be combined in a multibit register `DUALDFSF`. The D inputs and the Q outputs of the `DFCF` cell are inverted.

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    wire resetn, clk;
    wire [3:0] in;
    wire [3:0] out;
    wire n_31, n_32, n_43, n_44;
    DUALDFSF \CDN_MBIT_cnt_0_reg[0_1] (.SDN (resetn), .CP (clk), .D1(in[0]),
        .D2 (n_31), .Q1 (out[0]), .Q2 (n_32));
    INV g5(.I (in[1]), .ZN (n_31));
    INV g6(.I (n_32), .ZN (out[1]));
    DUALDFSF \CDN_MBIT_cnt_0_reg[2_3] (.SDN (resetn), .CP (clk), .D1(n_43),
        .D2 (in[3]), .Q1 (n_44), .Q2 (out[3]));
    INV g23(.I (in[2]), .ZN (n_43));
    INV g24(.I (n_44), .ZN (out[2]));
endmodule
```

- When the `multibit_allow_async_phase_map` attribute is set to `false`, the clear sequential cells cannot be mapped to preset ones, and consequently the `DFCF` cells cannot be replaced with a multibit register in this case, while the two `DFSF` cells are mapped to a multibit register.

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    wire resetn, clk;
    wire [3:0] in;
    wire [3:0] out;
    DFCF cnt_0_reg[1] (.CDN (resetn), .CP (clk), .D (in[1]),
        .Q (out[1]));
    DFCF cnt_0_reg[2] (.CDN (resetn), .CP (clk), .D (in[2]),
        .Q (out[2]));
    DUALDFSF \CDN_MBIT_cnt_0_reg[0_3] (.SDN (resetn), .CP (clk),
        .D1(in[0]), .D2 (in[3]), .Q1 (out[0]), .Q2 (out[3]));
endmodule
```

Related Information

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

Affects this command: [synthesize -incremental](#)

Related attributes: [cell_bitwidth](#) on page 191

[multibit_prefix_string](#) on page 801

[use_multibit_cells](#) on page 826

multibit_allow_unused_bits

```
multibit_allow_unused_bits {true | false}
```

Default: true

Read-write root attribute. Controls whether a mapped multibit instance can have one or more unused bits. By default, RTL Compiler will try to merge the single-bit registers to multibit instances without unused bits. If during merging, RTL Compiler cannot find a suitable multibit libcell with a width that matches the left-over bits of a bus, it will choose the closest matching width which leaves the minimum unused bits. If the attribute is set to false, any merging which results in unused (undriven/unloaded) bits will be prevented and thus reduce the multibit merging coverage.

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize -incremental](#)

Related attributes: [cell_bitwidth](#) on page 191

[use_multibit_cells](#) on page 826

multibit_cells_from_different_busses

```
multibit_cells_from_different_busses {false | true}
```

Default: false

Read-write root attribute. If this attribute remains false, only the sequential cells with the same basename will be merged. If this attribute is set to true, RTL Compiler can merge single-bit cells from different banks (also called busses) into a single, multibit cell. For example, four single-bit flops:

`p_reg[4], p_reg[5], q_reg[8] and q_reg[9]`

can be merged into one four-bit flop

`p_reg[4_5]_q_reg[8_9]`

The single-bit cells affected by this attribute include: flip-flops, latches, and tristate buffers.

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [multibit_seqs_members_naming_style](#) on page 805

Related attributes: [cell_bitwidth](#) on page 191

[use_multibit_cells](#) on page 826

multibit_debug

`multibit_debug {false | true}`

Default: false

Read-write [root](#) attribute. Controls whether to write debug information related to multibit mapping to the logfile.

Example

Consider the following netlist after mapping and before incremental synthesis

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [1:0] in;
    output [1:0] out;

    DFCF cnt_0_reg[0] (.CDN (resetn), .CP (clk), .D(in[0]), .Q (out[0]));
    DFCF cnt_0_reg[1] (.CDN (resetn), .CP (clk), .D(in[1]), .Q (out[1]));
endmodule

module top(resetn, clk, in, out);
    input resetn, clk;
    input [1:0] in;
    output [1:0] out;
    test t(resetn, clk, in, out);
endmodule
```

When `multibit_debug` is set to true, the following debug information is dumped in logfile during multibit mapping.

- If multibit instance is created:

```
Trying to make multibit bank  CDN_MBIT_cnt_0_reg[0]_MB_cnt_0_reg[1] out of
cnt_0_reg[0]          (DFCF)
cnt_0_reg[1]          (DFCF)
CDN_MBIT_cnt_0_reg[0]_MB_cnt_0_reg[1] multibit bank of type DUALDFSF
accepted in sandbox.
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- If multibit instance is not created:

```
Trying to make multibit bank  CDN_MBIT_cnt_0_reg[0]_MB_cnt_0_reg[1] out of
cnt_0_reg[0]          (DFCF)
cnt_0_reg[1]          (DFCF)
CDN_MBIT_cnt_0_reg[0]_MB_cnt_0_reg[1] multibit bank rejected (worse QoR).
```

Related Information

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

Affects this command: [synthesize -incremental](#)

Related attributes: [cell_bitwidth](#) on page 191

[use_multibit_cells](#) on page 826

multibit_mapping_effort_level

```
multibit_mapping_effort_level {high | auto| low}
```

Default: high

Read-write [root](#) attribute. Controls the effort level for physical-aware multibit mapping; that is, controls the tradeoff between multibit coverage and QoR. This attribute can have the following values:

auto	Balances multibit coverage and QoR to get the most optimal results.
high	Increases multibit coverage but might impact QoR.
low	Gives the best QoR.

Related Information

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

Affects this command: [synthesize -to_placed -incremental](#)

Related attributes: [use_multibit_cells](#) on page 826

multibit_predefined_allow_unused_bits

```
multibit_predefined_allow_unused_bits {false | true}
```

Default: false

Read-write root attribute. Controls whether instances of predefined multibit cells can have unused bits. By default, RTL Compiler does not allow unused bits in predefined multibit instances.

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize -incremental](#)

Related attributes: [cell_bitwidth](#) on page 191

[use_multibit_cells](#) on page 826

multibit_prefix_string

```
multibit_prefix_string string
```

Default: CDN_MBIT_

Read-write root attribute. Specifies the prefix to be used to name multibit instances. The recommended value for the verification flow is CDN_MBIT_.

Example

Consider the following netlist after mapping and before incremental optimization:

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    DFSF cnt_0_reg[0] (.SDN(resetn), .CP(clk), .D(in[0]), .Q(out[0]));
    DFCF cnt_0_reg[1] (.CDN(resetn), .CP(clk), .D(in[1]), .Q(out[1]));
    DFCF cnt_0_reg[2] (.CDN(resetn), .CP(clk), .D(in[2]), .Q(out[2]));
    DFSF cnt_0_reg[3] (.SDN(resetn), .CP(clk), .D(in[3]), .Q(out[3]));
endmodule
```

The netlist contains two clear sequential libcells and two preset sequential libcells.

Assume the `multibit_allow_async_phase_map` attribute is also set to `true`. For the previous netlist, that means that cells DFSF and DFCF can be combined in a multibit register

DUALDFSF. The D inputs and the Q outputs of the DFCF cell are inverted. Assume the multibit_prefix_string attribute is set to MBIT_. The resulting netlist will look like:

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    wire resetn, clk;
    wire [3:0] in;
    wire [3:0] out;
    wire n_31, n_32, n_43, n_44;
    DUALDFSF \MBIT_cnt_0_reg[0_1] (.SDN (resetn), .CP (clk), .D1(in[0]),
        .D2 (n_31), .Q1 (out[0]), .Q2 (n_32));
    INV g5(.I (in[1]), .ZN (n_31));
    INV g6(.I (n_32), .ZN (out[1]));
    DUALDFSF \MBIT_cnt_0_reg[2_3] (.SDN (resetn), .CP (clk), .D1(n_43),
        .D2 (in[3]), .Q1 (n_44), .Q2 (out[3]));
    INV g23(.I (in[2]), .ZN (n_43));
    INV g24(.I (n_44), .ZN (out[2]));
endmodule
```

Related Information

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

Affects this command: [synthesize -incremental](#)

Related attributes: [cell_bitwidth](#) on page 191

[use_multibit_cells](#) on page 826

[multibit_seqs_members_naming_style](#) on page 805

multibit_preserve_inferred_instances

```
multibit_preserve_inferred_instances {false | true}
```

Default: false

Read-write [root](#) attribute. Specifies whether to preserve multibit instances that are inferred during incremental optimization. If set to true, the [preserve](#) attribute for all multibit instances is set to [size_delete_ok](#). This preserve setting prevents that any optimization step down the flow breaks the multibit cells. For example, multibit tristate and combinational cells can get optimized in many places.

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize](#) -incremental

multibit_preserved_net_check

`multibit_preserved_net_check {false | true}`

Default: false

Read-write root attribute. Controls whether instances can be merged into multibit instances if the connected nets are marked preserved. Set this attribute to `true` to prevent multibit merging when any of the nets connected to the instances are marked preserved.

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize](#) -incremental

multibit_seqs_instance_naming_style

`multibit_seqs_instance_naming_style {concat | auto | short}`

Default: concat

Read-write root attribute. Controls the naming of inferred multibit (flops, latches and tristates) instances. This attribute can have the following values:

`auto` Creates the name of the multibit instance based on busses of merged instances.

- If the instances belong to the same bus, the name of the multibit instance starts with the bus name followed by the range of the merged bit-indices.
- If the instances belong to different busses, the name of the multibit instance contains each bus name followed by the range of corresponding bit-indices.

`concat` Creates the name of the multibit instance by concatenating the names of merged instances.

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

short	<p>Creates the name of the multibit instance by using the bus name for only the first index of a multibit cell if all instances belong to the same bus.</p> <p>If the multibit instance contains single bits from different busses, the <code>short</code> naming style has no effect and the <code>concat</code> style is followed.</p>
-------	--

Examples

- Consider the following instances:

```
SCFFQXC1 I3256 (.CLK(CLK), .DATA(D1), .Q(Q1));
SCFFQXC1 I3267 (.CLK(CLK), .DATA(D2), .Q(Q2));
SCFFQXC1 I3312 (.CLK(CLK), .DATA(D3), .Q(Q3));
SCFFQXC1 I3439 (.CLK(CLK), .DATA(D4), .Q(Q4));
SCFFQXC1 I3572 (.CLK(CLK), .DATA(D5), .Q(Q5));
SCFFQXC1 I4188 (.CLK(CLK), .DATA(D6), .Q(Q6));
SCFFQXC1 I4357 (.CLK(CLK), .DATA(D7), .Q(Q7));
SCFFQXC1 I4567 (.CLK(CLK), .DATA(D8), .Q(Q8));
```

- Using the `auto` setting:

```
set_attr multibit_seqs_instance_naming_style auto /
```

The multibit instances are named as follows:

```
SCCSGFF4QXC1 I3256_3439(.CLK (CLK), .DATA0 (D1), .DATA1 (D2),
                           .DATA2 (D3), .DATA3 (D4), .Q0 (Q1), .Q1 (Q2), .Q2 (Q3), .Q3 (Q4));
SCCSGFF4QXC1 I3572_4567(.CLK (CLK), .DATA0 (D5), .DATA1 (D6),
                           .DATA2 (D7), .DATA3 (D8), .Q0 (Q5), .Q1 (Q6), .Q2 (Q7), .Q3 (Q8));
```

- Using the `concat` setting:

```
set_attr multibit_seqs_instance_naming_style concat /
```

The multibit instances are named as follows:

```
SCCSGFF4QXC1 I3256_I3267_I3312_I3439(.CLK (CLK), .DATA0 (D1), .DATA1 (D2),
                                           .DATA2 (D3), .DATA3 (D4), .Q0 (Q1), .Q1 (Q2), .Q2 (Q3), .Q3 (Q4));
SCCSGFF4QXC1 I3572_I4188_I4357_I4567(.CLK (CLK), .DATA0 (D5), .DATA1 (D6),
                                           .DATA2 (D7), .DATA3 (D8), .Q0 (Q5), .Q1 (Q6), .Q2 (Q7), .Q3 (Q8));
```

- Consider the following instances:

```
a_reg[2]
a_reg[1]
b_reg[2]
b_reg[1]
a_reg[0]
b_reg[0]
```

- Using the `short` setting:

```
set_attr multibit_seqs_instance_naming_style short /
```

The multibit instances are named as follows:

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

```
CDN_MBIT_a_reg[0]_MB_a_reg[1]_MB_a_reg[2]_MB_b_reg[0]
CDN_MBIT_b_reg[1]_MB_[2]
```

The short naming style was not used for the first multibit instance because it contains single bits from two different busses.

The short naming style was used for the second multibit instance because both single bits belong to the same bus. The bus name is only shown for the first bit.

Related Information

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

Affects this command: [synthesize -incremental](#)

Related attribute: [multibit_seqs_name_concat_string](#) on page 807

multibit_seqs_members_naming_style

```
multibit_seqs_members_naming_style {list_of_strings}
```

Default: "%s%d" "%s%d_reg" "%s_%d_" "%s_%d_reg" "%s[%d]" "%s[%d]_reg"

Read-write [root](#) attribute. Controls which single-bit sequential instances can be merged into a multibit sequential instance during synthesis. The attribute value can contain a list of naming styles of the single-bit instances for which the merging can be done.

Examples

- To combine single bit cells a1 and a2, the following naming style must be part of the attribute value:

```
set_attribute multibit_seqs_members_naming_style "%s%d" /
```
- To combine single bit cells a1_reg and a2_reg, the following naming style must be part of the attribute value:

```
set_attribute multibit_seqs_members_naming_style "%s%d_reg" /
```
- To combine single bit cells a[1] and a[2], the following naming style must be part of the attribute value:

```
set_attribute multibit_seqs_members_naming_style {%s[%d]} /
```
- To add a new naming style to the current default, you can use the following commands:

```
set fmt [concat [get_attr multibit_seqs_members_naming_style /] %s%d_new]
set_attr multibit_seqs_members_naming_style $fmt /
```

This allows to combine single bit cells a1_new and a2_new into a multibit cell.

- The following example changes the default.

```
rc:/> get_attr multibit_seqs_members_naming_style /
%$%d %$%d_reg {$s[%d]} {$s[%d]_reg}
rc:/> set_attr multibit_seqs_members_naming_style {"%$%d" "%$%d_reg" "%$[%d]"}
      Setting attribute of root '/': 'multibit_seqs_members_naming_style' = %$%d
      %$%d_reg {$s[%d]}
```

- To combine single bit cells, a[1]_reg and a[2]_reg, the following naming style must be part of the attribute value:

```
set_attribute multibit_seqs_members_naming_style {$s[%d]_reg} /
```

Related Information

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

Affects this command: [synthesize -incremental](#)

Affects this attribute: [multibit cells from different busses](#) on page 798

multibit_short_prefix_string

`multibit_short_prefix_string string`

Default: `_CDN_CPX_`

Read-write `root` attribute. Specifies the string to separate the base name from the indexes of the multibit instances in those cases where the merged instances were created from the same bus, and the multibit instance name was created using the short naming style. The default value is recommended for the verification flow.

Example

Consider the following instances:

```
a_reg[0]
a_reg[1]
a_reg[2]
```

If the following attribute settings are used

```
set_attribute multibit_seqs_instance_naming_style short /
set_attribute multibit_short_prefix_string _CDN_CPX_ /
```

The multibit instance will be named as follows:

```
CDN_MBIT_a_reg_CDN_CPX_[0]_MB_[1]_MB_[2]
```

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize -incremental](#)

Affects this attribute: [multibit_seqs_instance_naming_style](#) on page 803

multibit_seqs_name_concat_string

`multibit_seqs_name_concat_string string`

Default: `_MB_`

Read-write [root](#) attribute. Specifies the separator string to be used in the name of the multibit instance created by concatenating the names of merged instances. The recommended values for the verification flow are `_MB_` and `_mb_`.

Example

```
set_attribute multibit_seqs_name_concat_string _mb_ /
```

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize -incremental](#)

Affects this attribute: [multibit_seqs_instance_naming_style](#) on page 803

multibit_skip_exception_check

`multibit_skip_exception_check {true | false}`

Default: `true`

Read-write [root](#) attribute. Controls whether timing exception checks should be skipped and sequential instances be merged even if they having timing exceptions. By default, the exception checks are skipped and merging happens for flops with identical exceptions.

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize -incremental](#)

multibit_split_string

`multibit_split_string string`

Default: `_split_`

Read-write root attribute. Specifies the string to be used to name single bit instances when the netlist that was read in already contains multibit cells.

When the tool merges single bit instances into a multibit instances, it keeps track of the original names during the current session. If the tool splits the multibit cell back into single bits in the same session, it can reuse the original names. If the merging and splitting occur in different sessions, the original names are not stored and the tool uses this attribute value to create the single bit names.

Example

Assume the following netlist was read in during the current session:

```
module PIPO_REGISTER_REGDIMENSION9(CLK, RSN, Inpbus, Outbus);
    input CLK, RSN;
    input [9:0] Inpbus;
    output [9:0] Outbus;

HS65V_GSH_4SDFPQX18 \OutbusintCK_reg[0_3] (.CP (CLK), .\D[0]
    (Inpbus[0]), .\D[1] (Inpbus[1]), .\D[2] (Inpbus[2]), .\D[3]
    (Inpbus[3]), .TI (Outbus[0]), .TE (1'b0), .\Q[0] (Outbus[0]),
    .\Q[1] (Outbus[1]), .\Q[2] (Outbus[2]), .\Q[3] (Outbus[3]));
endmodule
```

While performing incremental optimization the multibit cell is split again in single bit instances. The tool uses the value of this attribute to create the single bit names. Assume the attribute was set to `_user_`, the netlist written out would look like:

```
module PIPO_REGISTER_REGDIMENSION9(CLK, RSN, Inpbus, Outbus);
    input CLK, RSN;
    input [9:0] Inpbus;
    output [9:0] Outbus;
    wire CLK, RSN;
    wire [9:0] Inpbus;
    wire [9:0] Outbus;
    wire UNCONNECTED, UNCONNECTED0, UNCONNECTED1, UNCONNECTED2;
SDFFX4 \OutbusintCK_reg[0_3]_user_0 (.CK (CLK), .D (Inpbus[0]), .SI
    (Outbus[0]), .SE (1'b0), .Q (Outbus[0]), .QN (UNCONNECTED));
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

```
SDFFX4 \OutbusintCK_reg[0_3]_user_1 (.CK (CLK), .D (Inpbus[1]), .SI  
  (Outbus[1]), .SE (1'b0), .Q (Outbus[1]), .QN (UNCONNECTED0));  
SDFFX4 \OutbusintCK_reg[0_3]_user_2 (.CK (CLK), .D (Inpbus[2]), .SI  
  (Outbus[2]), .SE (1'b0), .Q (Outbus[2]), .QN (UNCONNECTED1));  
SDFFX4 \OutbusintCK_reg[0_3]_user_3 (.CK (CLK), .D (Inpbus[3]), .SI  
  (Outbus[3]), .SE (1'b0), .Q (Outbus[3]), .QN (UNCONNECTED2));  
endmodule
```

Related Information

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

Affects this command: [synthesize -incremental](#)

Related attributes: [cell_bitwidth](#) on page 191

[use_multibit_cells](#) on page 826

[multibit_seqs_members_naming_style](#) on page 805

multibit_unused_input_value

```
multibit_unused_input_value {0 | 1 | none}
```

Default: 0

Read-write [root](#) attribute. Specifies the value to which all unconnected input pins in the multibit cells must be connected. By default, the unused input pins of multibit cells are connected to constant 0. If you set the attribute to `none`, the input pins will be left floating.

Related Information

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

Affects this command: [synthesize -incremental](#)

Related attribute: [use_multibit_cells](#) on page 826

optimize_constant_0_flops

```
optimize_constant_0_flops {true | false}
```

Default: true

Read-write root attribute. Allows constant 0 propagation through flip-flops.

Related Information

Affects these commands: [elaborate](#)
 [synthesize](#)

Affects this attribute: [optimize_constant_0_seq](#) on page 852

optimize_constant_1_flops

```
optimize_constant_1_flops {true | false}
```

Default: true

Read-write root attribute. Allows constant 1 propagation through flip-flops.

Related Information

Affects these commands: [elaborate](#)
 [synthesize](#)

Affects this attribute: [optimize_constant_1_seq](#) on page 853

optimize_constant_latches

```
optimize_constant_latches {true | false}
```

Default: true

Read-write root attribute. When set to `true`, a latch whose output never changes is replaced by the corresponding constant value.

Related Information

Affects these commands: [elaborate](#)

synthesize

Affects this attribute: [optimize_constant_0_seq](#) on page 852
 [optimize_constant_1_seq](#) on page 853

optimize_merge_flops

`optimize_merge_flops {true | false}`

Default: true

Read-write root attribute. Controls merging of equivalent flops. Set this attribute to `false` to prevent merging.

Related Information

[When to Write out Intermediate Netlist](#) in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects these commands: [synthesize](#)
 [write_hdl -lec](#)

Affects this attribute: [optimize_merge_seq](#) on page 854

Related attribute: [optimize_merge_latches](#) on page 811

optimize_merge_latches

`optimize_merge_latches {true | false}`

Default: true

Read-write root attribute. Controls merging of equivalent latches. Set this attribute to `false` to prevent merging.

Related Information

[When to Write out Intermediate Netlist](#) in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects these commands: [synthesize](#)
 [write_hdl -lec](#)

Affects this attribute: [optimize_merge_seq](#) on page 854

Related attribute: [optimize_merge_flops](#) on page 811

optimize_net_area

`optimize_net_area {true | false}`

Default: true

Read-write root attribute. Controls optimization of the net area during global mapping and incremental optimization. By default, optimization tries to minimize the total area (the sum of the cell and net area). If this attribute is set to `false`, only the cell area will be minimized, which reduces the cell area at the cost of higher net area.

Related Information

Affects this command: [synthesize](#)

optimize_seq_x_to

`optimize_seq_x_to {0 | 1}`

Default: 0

Read-write root attribute. Allows the propagation of the specified constant value (0 or 1) through flops/latches when they are in `dont_care` (x) state.

propagate_constant_from_timing_model

`propagate_constant_from_timing_model {true | false}`

Default: true

Read-write root attribute. Controls constant propagation from timing model instances.

If your design contains instances of timing models whose output function is defined as constant 0 or 1, you must indicate whether RTL Compiler can propagate the constant. If you set the attribute to `false`, constants will not be propagated.

Related Information

Affects this command: [synthesize](#)

Related attribute: (instance) [propagate_constant_from_timing_model](#) on page 856

proto_feasible_target

`proto_feasible_target {false | true}`

Default: false

Read-write [root](#) attribute. Enables support for incomplete SDC constraints. When enabled, the mapper ignores huge negative slack endpoints in the target computation and optimization.



This feature is currently available as a [limited access feature](#).

Related Information

[Prototype Synthesis with Incomplete SDC Constraints in Encounter RTL Compiler Synthesis Flows](#)

Affects this command: [synthesize](#)

proto_feasible_target_adjust_slack_pct

`proto_feasible_target_adjust_slack_pct integer`

Default: 100

Read-write [root](#) attribute. Specifies the percentage of slack value to be used as positive adjust value in `path_adjust` exceptions.



This feature is currently available as a [limited access feature](#).

Related Information

[Prototype Synthesis with Incomplete SDC Constraints in Encounter RTL Compiler Synthesis Flows](#)

Affects this command: [synthesize](#)

proto_feasible_target_threshold

`proto_feasible_target_threshold float`

Default: no_value

Read-write [root](#) attribute. Specifies the minimum threshold delay for which `path_adjust` should be applied.



This feature is currently available as a [limited access feature](#).

Related Information

[Prototype Synthesis with Incomplete SDC Constraints in Encounter RTL Compiler Synthesis Flows](#)

Affects this command: [synthesize](#)

proto_feasible_target_threshold_clock_pct

`proto_feasible_target_threshold_clock_pct integer`

Default: 75

Read-write [root](#) attribute. Specifies the slack to clock period percentage for which `path_adjust` should be applied.



This feature is currently available as a [limited access feature](#).

Related Information

Prototype Synthesis with Incomplete SDC Constraints in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize](#)

proto_hdl

proto_hdl {false | true}

Default: false

Read-write [root](#) attribute. Enables support for incomplete HDL.



This feature is currently available as a [limited access feature](#).

Related Information

Prototype Synthesis with Incomplete HDL in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [elaborate](#)

remove_assigns

```
remove_assigns {false | true}
```

Default: false

Read-write root attribute. Determines whether assign statements should be replaced with buffers or inverters in the netlist.

When this attribute is set to true, the generated netlist will not contain any assign statements. Depending on the availability of cells in the library, and the settings of the `set_remove_assign_options` command, the assign statements might be replaced with buffers or inverters.

Examples

For the following examples, assume the following constant assignments in the RTL code:

```
assign out1 = 1'b0;  
assign out2 = 1'b0;
```

- If the `remove_assigns` attribute is set to false, the netlist will remain unchanged.
- If the `remove_assigns` attribute is set to true, the netlist will look like:

```
BUFX1 rm_assigns_buf_q3(.A (1'b0), .Y (out1));  
BUFX1 rm_assigns_buf_q3(.A (1'b0), .Y (out2));
```

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 these commands: [synthesize -incremental](#)

Affected by this command: [set_remove_assign_options](#)

Affects this attribute: [use_tiehi_lo_for_const on page 830](#)

retime_async_reset

```
retime_async_reset {true | false}
```

Default: true

Read-write root attribute. Specifies whether registers with asynchronous set or reset signals should be retimed. Registers with both asynchronous set and reset signals will not be retimed regardless of this attribute's value.

Example

The following example specifies that all asynchronous registers with set or reset signals should be considered for retiming:

```
rc:/> set_attribute retime_async_reset true /  
...  
rc:/> retime -min_delay
```

Related Information

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

Affects these commands: [retime](#)
 [synthesize -to_map](#)

Related attributes: [dont_retime](#) on page 842
 [\(design\) retime](#) on page 910
 [\(subdesign\) retime](#) on page 910
 [retime_optimize_reset](#) on page 819

retime_effort_level

```
retime_effort_level {medium | high}
```

Default: medium

Read-write root attribute. Controls the optimizations used during retiming.

Using the default effort ensures that the design can be verified with the Encounter® Conformal® Equivalence Checking tool using the default flow. Using high effort causes the tool to use optimization techniques which could trade off better QoR against ease of formal verification.

Related Information

Retiming the Design in Using Encounter RTL Compiler

Affects these commands:	retime synthesize -to_map write do lec
Related attributes:	dont_retime on page 842 (design) retime on page 910 (subdesign) retime on page 910

retime_move_mux_loop_with_reg

`retime_move_mux_loop_with_reg {true | false}`

Default: true

Read-write [root](#) attribute. Directs retiming to maintain the mux-feedback loop for a flop with the flop itself during the retiming moves. This implies that retiming cannot separate the mux loop and move logic through the feedback loop.

Related Information

Retiming the Design in Using Encounter RTL Compiler

Affects these commands:	retime synthesize -to_map
Related attributes:	dont_retime on page 842 (design) retime on page 910 (subdesign) retime on page 910

retime_optimize_reset

```
retime_optimize_reset {false | true}
```

Default: false

Read-write root attribute. Specifies whether the asynchronous reset signals of registers should be optimized during retiming optimization. If set to `true`, any asynchronous reset signal that can be eliminated while preserving the functionality of the logic will be removed. That is, the reset will be dropped if the computation results in a `dont_care` value.

Example

The following example specifies that all reset signals of registers should be optimized during retiming:

```
set_attribute retime_optimize_reset true /  
..  
retime -min_area
```

Related Information

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

Affects these commands: [retime](#)

[synthesize -to_map](#)

Related attributes: [dont_retime](#) on page 842

 (design) [retime](#) on page 910

 (subdesign) [retime](#) on page 910

[retime_async_reset](#) on page 817

retime_reg_naming_suffix

```
retime_reg_naming_suffix string
```

Default: `_reg`

Read-write root attribute. Marks those registers that were moved due to retiming optimization with the specified suffix.

Example

The following example instructs RTL Compiler to add the `_retimed_reg` suffix to all registers that are moved during retiming optimization:

```
rc:/> set_attribute retime_reg_naming_suffix _retimed_reg  
Setting attribute of root /: 'retime_reg_naming_suffix' = _retimed_reg
```

The affected registers could look like the following example:

```
(n_124)); D_F_LPH0002_H retime_16_reg(.E (ck), .D (n_118), .L2N (n_159)); INVERT_J  
g48(.A (n_167), .Z (n_119)); D_F_LPH0001_E retime_17_reg(.E (ck), .D (n_118), .L2  
(n_158)); D_F_LPH0002_E retime_8_reg(.E (ck), .D (n_112), .L2N (n_165)); INVERT_H  
g52(.A (n_116), .Z (n_117)); INVERT_H g55(.A (n_104), .Z (n_115));
```

Related Information

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

Affects these commands: [retime](#)

[synthesize -to_map](#)

Related attributes [dont_retime](#) on page 842

[\(design\) retime](#) on page 910

[\(subdesign\) retime](#) on page 910

[retime_async_reset](#) on page 817

[retime_original_registers](#) on page 951

[trace_retime](#) on page 857

retime_verification_flow

```
retime_verification_flow {true | false}
```

Default: true

Read-write [root](#) attribute. Enables the retiming verification with the Encounter® Conformal® Equivalence Checking tool. Set this attribute to true before you use the [synthesize](#) or [retime](#) command.

Related Information

Recommended Flow in Interfacing between Encounter RTL Compiler and Encounter Conformal.

Affects these commands:	retime synthesize -to_map
Related attributes	dont_retime on page 842 (design) retime on page 910 (subdesign) retime on page 910 retime_async_reset on page 817 retime_original_registers on page 951 trace_retime on page 857

retiming_clocks

`retiming_clocks clocks`

Read-write [root](#) attribute. Specifies to perform retiming on the registers clocked by the specified clocks.

Related Information

Retiming the Design in Using Encounter RTL Compiler

Affects these commands:	retime synthesize -to_map
Affected by these attributes:	dont_retime on page 842 (design) retime on page 910 (subdesign) retime on page 910
Related attributes:	retime_async_reset on page 817 retime_original_registers on page 951 trace_retime on page 857

stop_at_iopc_state

`stop_at_iopc_state IOPT_state`

Read-write root attribute. Specifies the state at which incremental optimization stopped. If you use the `cntrl-c` key sequence in the middle of incremental optimization, RTL Compiler will stop, bring you back to the RTL Compiler command line, and issue a warning message with an IOPT state. Next time you enter a RTL Compiler session (with the same commands, constraints, script, etc. that preceded the `cntrl-c` halt) you can use specify the IOPT state at which you stopped with the `stop_at_iopc_state` attribute. RTL Compiler will continue with the netlist it had generated at the specified state.

Related Information

Affects these commands: [synthesize -incr](#)
 [synthesize -to_map](#)

tns_opto

`tns_opto {true | false}`

Default: true

Read-write root attribute. Controls whether timing slack is optimized only on the most critical path or also on other paths with negative slack.

By default, the `synthesize -to_mapped` command focuses on the critical path to produce a smaller design. When the critical path has negative slack, other near critical paths can also be relaxed to recover area.

When the `tns_opto` attribute is set to `true`, it forces the tool to consider all the endpoints for the optimization. The appropriate weight is given to the slack of each endpoint during the optimization. More drastic area recovery is not performed on violating paths to prevent worsening the total negative slack (TNS).

Related Information

Affects this command: [synthesize](#)

ultra_global_mapping

`ultra_global_mapping {false | true}`

Default: false

Read-write [root](#) attribute. Controls the use of some enhanced optimization algorithms during global mapping which could improve the QoR on some designs. When enabled, this will have an impact on runtime due to the iterative nature of the algorithms.

Note: These enhanced optimization algorithms are only available when you use `synthesize -effort high`.

Related Information

Affects this command: [synthesize](#)

uniquify_naming_style

`uniquify_naming_style string`

Default: `%s_%d`

Read-write `root` attribute. Specifies the naming style of uniquified subdesigns. This attribute must be set before issuing the `elaborate` command, otherwise the attribute value will not be honored.

Note: This attribute is supported in the RTL flow and the structural flow.

Example

In the following example, the top module is named `sable_top`. The submodule name is `sable_sub`. The default naming style of `%s_%d` will cause the uniquified submodule to be named `sable_sub_1`.

```
rc:> edit_netlist uniquify sable_top
rc:> ls /designs/sable_top/subdesigns/
./      sable_sub/      sable_sub_1/
```

The naming style has now been changed to `%s_uniqued_%d`:

```
rc:> set_attribute uniquify_naming_style %s_uniqued_%d
rc:> elaborate
rc:> edit_netlist uniquify sable_top
rc:> ls /designs/sable_top/subdesigns/
./      sable_sub/      sable_sub_uniqued_1
```

Related Information

Related command: [edit_netlist uniquify](#)
[read_netlist](#)

use_compatibility_based_grouping

`use_compatibility_based_grouping {true | false}`

Default: true

Read-write root attribute. Controls the mapping of single-bit sequential, combinational, and tristate cells into multibit cells based on common control signals.

By default, the tool will create groups of compatible single-bit cells that have common control signals. Merging is only done within a group of compatible cells.

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize](#) -incremental

use_multibit_cells

```
use_multibit_cells {false | true}
```

Default: false

Read-write root attribute. If set to true, single-bit cells are mapped to an appropriate multibit cell from the technology library.

For example, four single-bit flops:

```
ff_reg[0] ff_reg[1] ff_reg[2] ff_reg[3]
```

can be merged into one four-bit flop

```
ff_reg[0_3]
```

Single-bit cells that can be merged include: flip-flops, latches, tristate buffers, and combinatorial cells.

When merging flops, the shared input is the clock line. When merging latches, the shared input is the gate line. When merging tri-state cells, the shared input is the enable line.

Note: Multibit cell inferencing is performed during incremental optimization rather than during the mapping stage. While cells that are marked unusable by RTL Compiler cannot be used during mapping, they can be used during incremental optimization.

This attribute controls both logical and physical-aware multibit mapping. If the design is placed, physical-aware multibit mapping is performed, otherwise logical multibit mapping is performed.

Related Information

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

[Mapping to Multi-Bit Scan Cells in Design for Test in Encounter RTL Compiler](#)

Affects this command: [synthesize -incremental](#)

Affects these attributes: [force merge combos into multibit cells](#) on page 718

[force merge seqs into multibit cells](#) on page 719

[use multibit combo cells](#) on page 827

[use multibit seq and tristate cells](#) on page 828

Affected by this attribute: [multibit seqs members naming style](#) on page 805

Related attributes: [cell bitwidth](#) on page 191

[multibit cells from different busses](#) on page 798

use_multibit_combo_cells

```
use_multibit_combo_cells {false | true}
```

Default: false

Read-write root attribute. Controls the mapping of single-bit combinational cells to appropriate multibit combinational cells from the technology library.

By default, this attribute inherits the last setting of the `use_multibit_cells` attribute, but you can explicitly override this setting to enable or disable combinational cell mapping.

Note: Multibit cell inferencing is performed during incremental optimization rather than during the mapping stage. While cells that are marked unusable by RTL Compiler cannot be used during mapping, they can be used during incremental optimization.

Example

```
rc:/> get_attr use_multibit_cells /  
false  
rc:/> get_attr use_multibit_combo_cells  
false  
rc:/> set_attr use_multibit_cells true /  
Setting attribute of root '/': 'use_multibit_cells' = true  
rc:/> get_attr use_multibit_combo_cells  
true  
rc:/> set_attr use_multibit_combo_cells false /  
Setting attribute of root '/': 'use_multibit_combo_cells' = false  
rc:/> set_attr use_multibit_cells true /  
Setting attribute of root '/': 'use_multibit_cells' = true  
rc:/> get_attr use_multibit_combo_cells  
true
```

Related Information

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

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [use_multibit_cells](#) on page 826

Related attribute: [cell_bitwidth](#) on page 191

use_multibit_seq_and_tristate_cells

```
use_multibit_seq_and_tristate_cells {false | true}
```

Default: false

Read-write root attribute. Controls the mapping of single-bit sequential and tristate cells to appropriate multibit cells from the technology library.

By default, this attribute inherits the last setting of the `use_multibit_cells` attribute, but you can explicitly override this setting to enable or disable sequential and tristate cell mapping.

Note: Multibit cell inferencing is performed during incremental optimization rather than during the mapping stage. While cells that are marked unusable by RTL Compiler cannot be used during mapping, they can be used during incremental optimization.

Example

```
rc:/> get_attr use_multibit_cells /  
false  
rc:/> get_attr use_multibit_seq_and_tristate_cells  
false  
rc:/> set_attr use_multibit_cells true /  
  Setting attribute of root '/': 'use_multibit_cells' = true  
rc:/> get_attr use_multibit_seq_and_tristate_cells  
true  
rc:/> set_attr use_multibit_seq_and_tristate_cells false /  
  Setting attribute of root '/': 'use_multibit_seq_and_tristate_cells' = false  
rc:/> set_attr use_multibit_cells true /  
  Setting attribute of root '/': 'use_multibit_cells' = true  
rc:/> get_attr use_multibit_seq_and_tristate_cells  
true
```

Related Information

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

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [use_multibit_cells](#) on page 826

Related attribute: [cell_bitwidth](#) on page 191

use_nextstate_type_only_to_assign_sync_ctrls

```
use_nextstate_type_only_to_assign_sync_ctrls {false | true}
```

Default: false

Read-write root attribute. Forces the tool to use the `nextstate_type.lib` pin attribute (when available in the library) to assign synchronous control pins.

By default, the tool uses the `nextstate_type.lib` pin attribute along with some heuristics based on pin names to assign synchronous control pins.

Related Information

Affects this command: [synthesize](#)

Affects this attribute: [sync_clear](#) on page 207

use_scan_seqs_for_non_dft

```
use_scan_seqs_for_non_dft {true|false|degenerated_only}
```

Default: true

Read-write root attribute. Controls the mapping of registers to scan flops for functional purposes. This attribute can have the following values:

degenerated_only	Allows mapping to a degenerated scan flip-flop (with the scan-related pins tied off) if the flip-flops do not pass the DFT rules checker.
false	Prevents mapping to scan flip-flops if the registers do not pass the DFT rules checker.
true	Maps any registers not targeted to scan for DFT to scan flops for functional use.

Note: A flop can only be mapped to scan for DFT if you run the `check_dft_rules` command and the flop passes the DFT rules check or if prior to mapping, you set the `dft_scan_map_mode` design attribute to `force_all`.

Related Information

Affects this command: [synthesize](#)

Related attributes: [dft_mapped](#) on page 1135
[dft_scan_map_mode](#) on page 1126

use_tiehilo_for_const

`use_tiehilo_for_const {none | duplicate | unique}`

Default: none

Read-write [root](#) attribute. Determines whether a constant assignment should be replaced with a tie cell in the netlist. The attribute can have the following values:

duplicate	Allows each constant assignment to be replaced with a tie cell.
none	Prevents the replacement of constants in the netlist with tie cells.
unique	Allows only one unique tie cell in the netlist. Treatment of the remaining constant assignments depends on settings of the <code>remove_assigns</code> attribute and the <code>set_remove_assign_options</code> command.

Note: When tie-cell insertion is done as part of incremental optimization, hierarchical constant connected pins which are not used inside the module are by default skipped during insertion of tie high and tie low cells. If you use the `insert_tiehilo_cells` command to insert tie high and tie low cells, these hierarchical pins are not skipped by default and you can use the `-skip_unused_hier_pins` option to skip them.

Examples

For the following examples, assume the following constant assignments in the RTL code:

```
assign out1 = 1'b0;  
assign out2 = 1'b0;
```

In addition, assume the `remove_assigns` attribute is set to false.

- If the `use_tiehilo_for_const` attribute is set to `none`, the netlist will remain unchanged.

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Root Attributes

- If the `use_tiehilo_for_const` attribute is set to `duplicate`, the netlist will look like:

```
TIELO tie_0_cell(.Y (out1));  
TIELO tie_0_cell(.Y (out2));
```

- If the `use_tiehilo_for_const` attribute is set to `unique`, the netlist will look like:

```
TIELO tie_0_cell(.Y (out1));  
assign out2 = out1;
```

Related Information

Affects this command: [synthesize](#)

Affected by this command: [set_remove_assign_options](#)

Affected by these attribute: [ignore_preserve_in_tiecell_insertion](#) on page 784
[remove_assigns](#) on page 816

Design Attributes

Contain information about the specified design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design
```

or

```
set_attribute attribute_name attribute_value [find /des* -design name]
```

- To get a design attribute value, type

```
get_attribute attribute_name design
```

control_logic_optimization

```
control_logic_optimization {inherited | none | basic | advanced}
```

Default: inherited

Read-write design attribute. Controls the optimization of control logic (described in the RTL through conditional constructs like case statements, if-then-else statements, conditional selects, and so on) in the design during generic synthesis. You can specify any of the following values:

inherited	Inherits the value from the <code>control_logic_optimization</code> root attribute.
advanced	Applies advanced level optimization
basic	Applies basic optimization.
none	Turns off optimization.

Note: All control optimization transformations are verifiable. The advanced transformations might result in better QoR but can also increase the runtime. For best results, set this attribute after you have elaborated the design, but before generic synthesis.

Related Information

Affects this command: [synthesize_to_generic](#)

Related attributes: (root) [control_logic_optimization](#) on page 707

(subdesign) [control_logic_optimization](#) on page 899

delete_unloaded_seqs

```
delete_unloaded_seqs {inherited | false | true}
```

Default: inherited

Read-write design attribute. Controls the deletion of unloaded sequential instances in the design. You can specify one of the following values:

false	Prevents the deletion of unloaded sequential instances.
inherited	Inherits the value from the <code>delete_unloaded_seqs</code> root attribute.
true	Removes flip-flops and logic if they are not transitively fanning out to output ports.

Related Information

Affects this command:	synthesize
Affected by these attributes:	(root) delete_unloaded_seqs on page 709
Related attributes:	(root) delete_unloaded_insts on page 708 (subdesign) delete_unloaded_insts on page 900 (subdesign) delete_unloaded_seqs on page 901 hdl_preserve_unused_registers on page 759

dp_csa

```
dp_csa {inherited | basic | none}
```

Default: inherited

Read-write design attribute. Controls the carry-save adder (CSA) transformations in within the design. You can specify one of the following values:

basic	Applies basic transformation.
inherited	Inherits the value from the <code>dp_csa</code> root attribute.
none	Turns off CSA transformation.

Related Information

[Controlling CSA Transformations in Datapath Synthesis in Encounter RTL Compiler](#)

Affects this command: [synthesize -to_generic](#)

Related attributes: (root) [dp_csa](#) on page 711

 (subdesign) [dp_csa](#) on page 901

dp_rewriting

`dp_rewriting {inherited | none | basic | advanced}`

Default: inherited

Read-write [design](#) attribute. Controls how the datapath rewriting optimization is applied on the specified module during `synthesize -to_generic -effort high`. If the value of this attribute is `inherited`, the effort level of the optimization on the design will be identical to (“inherited from”) the effort specified at the root.

<code>inherited</code>	Inherits the value from the <code>dp_rewriting</code> root attribute.
<code>advanced</code>	Applies advanced level optimization
<code>basic</code>	Applies basic optimization.
<code>none</code>	Turns off optimization.

Related Information

Affects this command: [synthesize -to_generic -effort high](#)

Related attributes: (root) [dp_rewriting](#) on page 712

 (subdesign) [dp_rewriting](#) on page 902

dp_sharing

```
dp_sharing {inherited | none | basic | advanced}
```

Default: inherited

Read-write design attribute. Controls resource sharing in datapath on the design during `synthesize -to_generic -effort high`. If the value is set to `inherited`, the effort level of the optimization on the design will be identical to (“inherited from”) the effort specified at the root.

inherited	Inherits the value from the <code>dp_sharing</code> root attribute.
advanced	Applies advanced level optimization
basic	Applies basic optimization.
none	Turns off optimization.

Example

The following command sets the level of RTL sharing transformations to `basic`:

```
rc:/> set_attribute dp_sharing basic /designs/test
```

Related Information

Controlling Sharing Transformations in Datapath Synthesis in Encounter RTL Compiler

Affects this command: [synthesize -to_generic -effort high](#)

Related attributes: (root) [dp_sharing](#) on page 713

 (subdesign) [dp_sharing](#) on page 903

dp_speculation

```
dp_speculation {inherited | basic | none}
```

Default: inherited

Read-write design attribute. Controls RTL speculation (unsharing) transformations within a particular subdesign. You can specify one of the following values:

basic	Applies basic RTL speculation transformation.
inherited	Inherits the value from the dp_speculation root attribute.
none	Turns off RTL speculation transformation.

Example

The following command turns off RTL speculation transformations within design test:

```
rc:/> set_attribute dp_speculation none [find /designs -design test]
```

Related Information

[Controlling Speculation Transformations in Datapath Synthesis in Encounter RTL Compiler](#)

Affects this command:	<u>synthesize -to_generic -effort high</u>
Related attributes:	(root) <u>dp_speculation</u> on page 714
	(subdesign) <u>dp_speculation</u> on page 904

dp_verify_ok

```
dp_verify_ok {false | true}
```

Default: false

Read-write design attribute. When enabled, eases verification by limiting few optimizations on CSA modules such as boundary optimization on CSA hierarchies and avoiding ungrouping of simple CSA and modules in the fanout of CSA trees. Due to potential QoR impact, set this attribute to true only on specific verification-challenged user-subdesigns which contain CSA logic. This will enable the Encounter® Conformal® Logical Equivalence Checking (LEC) tool to do a better job of datapath learning and successful verification.

Example

Consider the following module.

```
module csa_add_mux_add(a, b, c, d, e, f, g, s, z);
  input [7:0] a, b, c, d, e, f, g;
  input s;
  output [10:0] z;
  wire [9:0] tmp;

  assign tmp = s ? a+b+f : c+d+g;
  assign z = tmp + e;

endmodule
```

When you use the default setting, all CSA trees will be ungrouped after mapping. When you set this attribute to true, none of the CSA trees will be ungrouped.

Related Information

Affects this command: [synthesize -to_generic -effort {medium| high}](#)

Related attribute: [\(subdesign\) dp_verify_ok on page 904](#)

hdl_cw_list

```
hdl_cw_list {{language library component}...}
```

Read-write [design](#) attribute. Returns a Tcl list of Tcl lists. There can be as many Tcl lists as there are types of Chipware components in the top module. Each Tcl list contains three elements:

- The *language* of the module in which the ChipWare component is instantiated
- The name of the ChipWare *library* that contains the ChipWare component
- The name of the ChipWare *component*

Note: Each ChipWare component used in the design appears just once in the `hdl_cw_list` attribute, no matter how many times it is instantiated.

Example

In the following example, component `DW01_add` is instantiated in design `test`, a Verilog-2001 design.

```
rc:/designs/test> get_attribute hdl_cw_list
{-v2001 DW01 DW01_add}
```

Related Information

Affected by this command: [elaborate](#)

Related attribute: (subdesign) [hdl_cw_list](#)

preserve

```
preserve {false | true | const_prop_delete_ok | const_prop_size_delete_ok  
| delete_ok | map_size_ok | size_ok | size_delete_ok}
```

Default: false

Read-write [design](#) attribute. Controls the optimization of the design. You can set the following options:

const_prop_delete_ok	Allows deleting a mapped subdesign and its instances, and constant propagation through the mapped subdesign and its instances, but not resizing, renaming or remapping.
const_prop_size_delete_ok	Allows deleting and resizing a mapped subdesign and its instances, or constant propagation through a mapped subdesign and its instances, but not renaming or remapping.
delete_ok	Allows deleting a mapped subdesign or child instance during optimization, but not resizing, renaming, or remapping it.
false	Allows logic changes to any object in the design during optimization.
map_size_ok	Allows resizing, unmapping, and remapping of a mapped sequential instance during optimization, but not renaming or deleting it.
size_delete_ok	Allows resizing or deleting a mapped subdesign or child instance during optimization, but not renaming or remapping it.
size_ok	Allows resizing a mapped subdesign or child instance during optimization, but not deleting, renaming, or remapping it.
true	Prevents logic changes to any object in the design during optimization.

Related Information

Affects this command:	synthesize
Affects these attributes:	(instance) preserve on page 855
	(net) preserve on page 876
	(pin) preserve on page 886
	(subdesign) preserve on page 909

retime

`retime {false | true}`

Default: false

Read-write [design](#) attribute. Marks the specified design to be retimed during synthesis. This attribute must be set after the `elaborate` command but before the `synthesize` command.

Example

The following marks the `m1` design for retiming:

```
rc:/> set_attribute retime true [find / -design m1]
Setting attribute of m1: 'retime' = true
```

Related Information

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

Affects this command:	synthesize
Affected by these commands:	dont_retime on page 842 retime_hard_region on page 910 retime_reg_naming_suffix on page 819
Related attribute:	(subdesign) retime on page 910

undesirable_libcells

`undesirable_libcells libcells`

Read-write design attribute. Specifies a list of libcells that the tool should avoid during synthesis and optimization of the design. The tool can still use some of these libcells if there are no other libcells available to synthesize the design.

Related Information

Affects this command: [synthesize](#)

Affects these attributes: (instance) [undesirable_libcells](#) on page 858

Instance Attributes

Contain information about a combinational, sequential, or hierarchical instance in the specified design. Instance objects are found in the `instances_comb`, `instances_seq`, and `instances_hier` directories.

- To set an instance attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -instance name]
```

- To get an instance attribute value, type

```
get_attribute attribute_name [find /des*/design -instance name]
```

dont_infer_multibit

```
dont_infer_multibit {false | true}
```

Default: false

Read-write instance attribute. Controls whether the instance should be avoided during multibit merging.

As shown in the following table, the tool only avoids multibit merging for the instance when the `infer_multibit` for this instance is not set.

dont_infer_multibit	infer_multibit	Multibit merging is
false	false	attempted if <code>use_multibit_cells</code> is set to true.
false	true	attempted.
true	false	not attempted.
true	true	attempted. The <code>dont_infer_multibit</code> attribute setting is ignored.

Note: This attribute applies only to sequential and tristate instances.

Related Information

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

[Specifying Pragmas to Control Mapping to Multi-Bit Registers in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [infer_multibit](#) on page 844

Related attribute: [use_multibit_cells](#) on page 826

dont_retime

`dont_retime {false | true}`

Default: false

Read-write [instance](#) attribute. Instructs RTL Compiler not to move any of the specified instances for retiming. This attribute is only available on registers.

Example

The following example marks the U2 instance as immovable for retiming.

```
rc:/> set_attribute dont_retime true [find / -instance U2]
```

Related Information

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

Affects this command: [retime](#)

Affects these attributes: (design) [retime](#) on page 839

 (subdesign) [retime](#) on page 910

Related attributes: [retime_hard_region](#) on page 910

[retime_reg_naming_suffix](#) on page 819

dont_split_multibit

`dont_split_multibit {false | true}`

Default: false

Read-write [instance](#) attribute. Controls whether the multibit cell used for this instance can be split during incremental optimization.

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command: [synthesize -incremental](#)

Related attribute: [use_multibit_cells](#) on page 826

dont_use_qbar_pin

`dont_use_qbar_pin {false | true}`

Default: false

Read-write [instance](#) attribute. Controls the use of the Qbar output pin of this sequential instance during optimization if other possibilities exist. By default, these pins can be used.

Related Information

Affects this command: [synthesize](#)

Related attribute: (root) [dont_use_qbar_seq_pins](#) on page 710

hdl_proc_name

`hdl_proc_name string`

Read-write [instance](#) attribute. If the `hdl_enable_proc_name` attribute is set to true, specifies for sequential elements either

- The Verilog block identifier of the named always block that infers this sequential element
- The VHDL label of the process that infers this sequential element

If no name was given to the Verilog block or VHDL process, a tool-generated name is given.

Note: This attribute is created during elaboration. After elaboration, it has no value for hierarchical instances, or for instances that are not sequential elements.

Related Information

Affects this command: [elaborate](#)

Affected by this attribute: [hdl_enable_proc_name](#) on page 732

infer_multibit

```
infer_multibit {false | true}
```

Default: false

Read-write [instance](#) attribute. Controls whether the instance can be considered for multibit merging. If set, the instance can be considered.

The following table shows when the tool considers multibit merging for the instance.

dont_infer_multibit	infer_multibit	Multibit merging is
false	false	attempted if <code>use_multibit_cells</code> is set to true.
false	true	attempted.
true	false	not attempted.
true	true	attempted. The <code>dont_infer_multibit</code> attribute setting is ignored.

Note: This attribute applies only to sequential and tristate instances.

Related Information

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

[Specifying Pragmas to Control Mapping to Multi-Bit Registers in *HDL Modeling in Encounter RTL Compiler*](#)

Affects this command: [synthesize -incremental](#)

Affected by this attribute: [dont_infer_multibit](#) on page 841

Related attribute: [use_multibit_cells](#) on page 826

inherited_preserve

```
inherited_preserve {const_prop_delete_ok | const_prop_size_delete_ok | delete_ok |
    false | true | map_size_ok | size_ok | size_delete_ok}
```

Read-only instance attribute. Returns the effective *preserve* value that was either explicitly set on this instance, or that this instance inherited from its parent module or hierarchical instance. The attribute can have the following values:

const_prop_delete_ok	Allows deleting a mapped instance and constant propagation through the mapped instance, but not resizing, renaming or remapping it.
const_prop_size_delete_ok	Allows deleting and resizing a mapped instance, or constant propagation through a mapped instance, but not renaming or remapping it.
delete_ok	Allows deleting a mapped instance during optimization, but not resizing, renaming, or remapping it.
false	Allows logic changes to a mapped instance in the design during optimization.
map_size_ok	Allows mapping, resizing, and remapping of unmapped sequential instances during optimization, but not renaming or deleting them.
size_delete_ok	Allows resizing or deleting a mapped instance during optimization, but not renaming or remapping it.
size_ok	Allows resizing a mapped instance during optimization, but not deleting, renaming, or remapping it.
true	Prevents logic changes to a mapped instance in the design during optimization.

When the *preserve* attribute is set on an instance and its parent module, the most restrictive value is inherited.

Example

Assuming that *preserve* was set to *delete_ok* on an instance and set to *size_ok* on the parent module, the value of *inherited_preserve* would be *true*.

Related Information

Affects this command: [synthesize](#)

Affected by this attribute: [preserve](#) on page 855

map_to_multibit_bank_label

`map_to_multibit_bank_label string`

Default: default_bank

Read-write `instance` attribute. Defines a bank label for a sequential instance. Instances with the same label can be considered for multibit mapping to the same bank of multibit instance.

If this attribute is set on a flop but no value is given to the `map_to_multibit_register` attribute, the flop is considered for *regular* multibit cell inferencing.

If several instances have the same bank label and the same `map_to_multibit_register` attribute value, those instances will be considered to be mapped to the same bank of the multibit cells specified by the `map_to_multibit_register` attribute.

Example

The following commands first define a label for a set of flops, then specify all library cells that can be used for multibit mapping of the set.

```
rc:/> set_attr map_to_multibit_bank_label bank_1 [find /des* -inst Outbus_reg[1?\]]  
      Setting attribute of instance 'Outbus_reg[10]': 'map_to_multibit_bank_label' =  
      bank_1  
...  
      Setting attribute of instance 'OutbusintCK_reg[19]': 'map_to_multibit_bank_label'  
      = bank_1  
rc:/> set_attr map_to_multibit_register HS65V_GSH_4DFPQX18 [find /des* -inst  
      Outbus_reg[1?\]]  
      Setting attribute of instance 'Outbus_reg[10]': 'map_to_multibit_register'  
      = /libraries/COR65/libcells/HS65V_4DFPQX18  
...  
      Setting attribute of instance 'Outbus_reg[19]': 'map_to_multibit_register' = /  
      libraries/COR65/libcells/HS65V_4DFPQX18
```

Related Information

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

Affects this command: [synthesize](#)

Related attribute: [map_to_multibit_register](#) on page 847

map_to_multibit_register

`map_to_multibit_register libcell_list`

Read-write instance attribute. Enables predefined multibit cell inferencing (MBCI) for the sequential instance and limits the multibit mapping of this instance to the specified multibit library cells. Since this attribute enables *predefined* multibit cell inferencing, it implies a forced type of mapping and is therefore not QoR-driven.

All the specified library cells must

- Have same bit width.
A bit width mismatch implies no multibit cell inferencing for this bank-label.
- Be functionally swappable.
A mismatch will not allow you to set the value for this attribute.

All instances with the same bank label (set through the `map_to_multibit_bank_label` attribute) and the same `map_to_multibit_register` attribute value, are considered to be mapped to the same bank of multibit cells.

Example

The following example does not specify a specific bank label on the instances, only the list of all library cells that can be used for multibit mapping of the set.

```
rc:/> set_attr map_to_multibit_register HS65V_4DFPQX18 [find /des* -inst Outbus_reg\[1?\]]  
Setting attribute of instance 'Outbus_reg[10]': 'map_to_multibit_register'  
= /libraries/COR65/libcells/HS65V_4DFPQX18  
...  
Setting attribute of instance 'Outbus_reg[19]': 'map_to_multibit_register' = /  
libraries/COR65/libcells/HS65V_4DFPQX18
```

Related Information

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

Affects this command: [synthesize](#)

Affected by this attribute: [map_to_multibit_bank_label](#) on page 846

Related attribute: [bank_based_multibit_inferencing](#) on page 698

map_to_mux

```
map_to_mux {false | true}
```

Default: false

Read-write instance attribute. When set to true, maps the instance to a binary mux cell. If the `map_to_mux` pragma is found in the RTL, and there is a way to infer a mux cell from the RTL annotated by this pragma, then elaboration creates a binary mux instance. Examining the `map_to_mux` pragma of specific instances let you find out whether they are honored by RTL Compiler.

Note: If you want to apply this attribute to this instance only, you must first uniquify the corresponding subdesign, otherwise it affects all instantiations of the subdesign.

Note: Using this attribute may affect the QoR.

Related Information

Affects this command: [synthesize](#)

Related attribute: [input map_to_mux pragma](#) on page 268

map_to_register

```
map_to_register libcell_list
```

Read-write instance attribute. Lists the libcells that can be used for mapping the specified sequential cell.

Examples

- The following command forces instance A3 to be mapped to libcell DFPQX1, while all other flops in the design can be mapped to any usable flop.

```
set_attribute map_to_register [find /libraries/tut/ -libcell DFPQX1] \  
[find /des*/top -instance A3]
```

- The following command forces instances A1, A2, and A3 to be mapped to either DFPQX1, DFPQX2, or DFPQX3 while all other flops in the design can be mapped to any usable flop.

```
set_attribute map_to_register {DFPQX1 DFPQX2 DFPQX3} {A1 A2 A3}
```

- The following command forces any instance starting with `q_reg` to be mapped to any libcell starting with SDFFSX.

```
set_attr map_to_register [find / -libcell SDFFSX*] [find / -inst u1/q_reg*]
```

Related Information

Affects this command: [synthesize](#)

merge_combinational_hier_instance

`merge_combinational_hier_instance {inherited | false | true}`

Default: inherited

Read-write [instance](#) attribute. Controls the merging of this combinational hierarchical instance. You can specify the following values:

false	Prevents merging of this combinational hierarchical instance.
inherited	If the instance is a combinational hierarchical instance, it inherits the value of the <code>merge_combinational_hier_instances</code> root attribute.
true	Allows merging of this combinational hierarchical instance.

Note: This attribute applies only to combinational hierarchical instances.

Related Information

Affects these commands: [synthesize -to_generic](#)

[synthesize -to_mapped](#)

Affected by this attribute: [merge_combinational_hier_instances](#) on page 794

multibit_allow_async_phase_map

`multibit_allow_async_phase_map {inherited | true | false}`

Default: inherited

Read-write [instance](#) attribute. Controls whether phase inversion is allowed during multibit cell inferencing for this instance. You can specify one of the following values:

false	Prevents phase inversion on the instance during multibit mapping.
inherited	Inherits the value from the <code>multibit_allow_async_phase_map</code> subdesign attribute.

true Allows phase inversion on the instance during multibit mapping.

Examples

Consider the following netlist after mapping and before incremental optimization: The library contains the multibit cell DUALDFSF.

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    DFCF cnt_0_reg[0] (.CDN (resetn), .CP (clk), .D(in[0]), .Q (out[0]));
    DFCF cnt_0_reg[1] (.CDN (resetn), .CP (clk), .D(in[1]), .Q (out[1]));
    DFCF cnt_0_reg[2] (.CDN (resetn), .CP (clk), .D(in[2]), .Q (out[2]));
    DFCF cnt_0_reg[3] (.CDN (resetn), .CP (clk), .D(in[3]), .Q (out[3]));
endmodule
module top(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    test t(resetn,clk,in,out);
endmodule
```

- In this example, the `multibit_allow_async_phase_map` attribute is not set for the instances, so the instance attributes inherit the value from the `multibit_allow_async_phase_map` subdesign attribute. Assuming the attribute is not set for subdesign `test`, the subdesign attribute inherits its value from the root attribute, which defaults to `true`.

With phase inversion allowed, the netlist is modified as follows:

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    wire resetn, clk;
    wire [3:0] in;
    wire [3:0] out;
    DUALDFSF \CDN_MBIT_cnt_0_reg[0_1] (.SDN (resetn), .CP (clk), .D1(in[0]),
        .D2 (n_31), .Q1 (out[0]), .Q2 (n_32));
    INV g5(.I (in[1]), .ZN (n_31));
    INV g6(.I (n_32), .ZN (out[1]));
    DUALDFSF \CDN_MBIT_cnt_0_reg[2_3] (.SDN (resetn), .CP (clk), .D1(n_43),
        .D2 (in[3]), .Q1 (n_44), .Q2 (out[3]));
    INV g23(.I (in[2]), .ZN (n_43));
    INV g24(.I (n_44), .ZN (out[2]));
endmodule
```

- In the following example, the `multibit_allow_async_phase_map` attribute is explicitly set to `false` for two instances.

```
set_attr multibit_allow_async_phase_map false [find / -inst cnt_0_reg[0]]
set_attr multibit_allow_async_phase_map false [find / -inst cnt_0_reg[1]]
```

Phase inversion is now only allowed for instances, `cnt_0_reg[2]` and `cnt_0_reg[3]`, as their instance attribute is not set, and neither is the subdesign attribute, nor the root attribute. Following is the resulting netlist:

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Instance Attributes

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    wire resetn, clk;
    wire [3:0] in;
    wire [3:0] out;
    DFCF cnt_0_reg[0] (.CDN (resetn), .CP (clk), .D(in[0]), .Q (out[0]));
    DFCF cnt_0_reg[1] (.CDN (resetn), .CP (clk), .D(in[1]), .Q (out[1]));
    DUALDFSF \CDN_MBIT_cnt_0_reg[2_3] (.SDN (resetn), .CP (clk), .D1(n_43),
        .D2 (in[3]), .Q1 (n_44), .Q2 (out[3]));
    INV g23(.I (in[2]), .ZN (n_43));
    INV g24(.I (n_44), .ZN (out[2]));
endmodule
```

Related Information

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

Affects this command: [synthesize -incremental](#)

Related attributes: [cell_bitwidth](#) on page 191

(root) [multibit allow async phase map](#) on page 796

(subdesign) [multibit allow async phase map](#) on page 907

[multibit_prefix_string](#) on page 801

[use_multibit_cells](#) on page 826

optimize_constant_0_seq

```
optimize_constant_0_seq {inherited | false | true}
```

Default: inherited

Read-write instance attribute. Controls constant 0 propagation through this sequential instance. You can specify the following values:

false	Prevents constant 0 propagation through this sequential instance.
inherited	If the sequential instance is a flop, it inherits the value of the <u>optimize_constant_0_flops</u> root attribute. If the sequential instance is a latch, it inherits the value of the <u>optimize_constant_latches</u> root attribute.
true	Allows constant 0 propagation through the sequential instance, and thus allows removal of the instance.

Note: This attribute applies only to sequential instances.

Related Information

Affects this command: [synthesize](#)

Affected by this attribute: [optimize_constant_0_flops on page 810](#)

[optimize_constant_latches on page 810](#)

optimize_constant_1_seq

optimize_constant_1_seq {inherited | false | true}

Default: inherited

Read-write instance attribute. Controls constant 1 propagation through this sequential instance. You can specify the following values:

false	Prevents constant 1 propagation through this sequential instance.
inherited	If the sequential instance is a flop, it inherits the value of the optimize_constant_1_flops root attribute. If the sequential instance is a latch, it inherits the value of the optimize_constant_latches root attribute.
true	Allows constant 1 propagation through the sequential instance, and thus allows removal of the instance.

Note: This attribute applies only to sequential instances.

Related Information

Affects this command:	synthesize
Affected by this attribute:	optimize_constant_0_flops on page 810 optimize_constant_latches on page 810

optimize_constant_feedback_seq

optimize_constant_feedback_seq {true | false}

Default: true

Read-write instance attribute. Controls constant propagation through a sequential cell that has a feedback loop. You can specify the following values:

false	Prevents constant propagation through this sequential instance.
true	Allows constant propagation through the sequential instance, and thus allows removal of the instance.

Note: This attribute applies only to sequential instances.

Related Information

Affects this command: [synthesize](#)

optimize_merge_seq

`optimize_merge_seq {inherited | true | false}`

Default: inherited

Read-write [instance](#) attribute. Controls merging of this instance with equivalent sequential instances. You can specify the following values:

false	Prevents merging of this sequential instance with other equivalent sequential instances.
inherited	If the sequential instance is a flop, it inherits the value of the <code>optimize_merge_flops</code> root attribute. If the sequential instance is a latch, it inherits the value of the <code>optimize_merge_latches</code> root attribute.
true	Allows merging of this sequential instance with other equivalent sequential instances.

Note: This attribute applies only to sequential instances.

Related Information

[When to Write out Intermediate Netlist](#) in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects these commands: [synthesize](#)

[write_hdl](#) -lec

Affected by these attributes: [optimize_merge_flops](#) on page 811

[optimize_merge_latches](#) on page 811

preserve

```
preserve {false | true | const_prop_delete_ok | const_prop_size_delete_ok  
| delete_ok | map_size_ok | size_ok | size_delete_ok}
```

Default: false

Read-write instance attribute. Controls optimization of a mapped instance. You can set the following options:

const_prop_delete_ok	Allows deleting a mapped instance and constant propagation through the mapped instance, but not resizing, renaming or remapping it.
const_prop_size_delete_ok	Allows deleting and resizing a mapped instance, or constant propagation through a mapped instance, but not renaming or remapping it.
delete_ok	Allows deleting a mapped instance during optimization, but not resizing, renaming, or remapping it.
false	Allows logic changes to an instance in the design during optimization.
map_size_ok	Allows resizing, unmapping, and remapping of a sequential instance during optimization, but not renaming or deleting it.
size_delete_ok	Allows resizing or deleting a mapped instance during optimization, but not renaming or remapping it.
size_ok	Allows resizing a mapped instance during optimization, but not deleting, renaming, or remapping it.
true	Prevents logic changes to a mapped instance in the design during optimization.

Related Information

Affects this command:	synthesize
Affects this attribute:	inherited_preserve on page 845
Related attributes:	(design) preserve on page 838 (net) preserve on page 876 (pin) preserve on page 886 (subdesign) preserve on page 909

propagate_constant_from_timing_model

`propagate_constant_from_timing_model {inherited | false | true}`

Default: inherited

Read-write instance attribute. Controls constant propagation from this timing model instance.

Note: This attribute applies to instances of timing models whose output function is defined as constant 0 or 1.

You can specify the following values:

false (or 0)	Prevents constant propagation from this timing model instance.
inherited	Inherits the value of the <code>propagate_constant_from_timing_model</code> root attribute.
true (or 1)	Allows constant propagation from this timing model instance.

Related Information

Affects this command: [synthesize](#)

Related attribute: (root) [propagate_constant_from_timing_model](#) on page 812

trace_retime

```
trace_retime {false | true}
```

Default: false

Read-write [instance](#) attribute. When set to true, the specified register is “marked” so that it can be retrieved and identified after retiming optimization. All registers that were marked with this attribute can be retrieved with the [retime_original_registers](#) attribute.



This attribute is meant to be used for debugging purposes only due to potential performance implications.

Example

The following example marks all the registers in the design and then retrieves the retimed instances with the [retime_original_registers](#) command:

```
rc:/> set_attribute trace_retime true [find / -instance *seq/*]  
...  
rc:/> retime -min_delay  
rc:/> get_attribute retime_original_registers [find / -instance *seq/*]
```

Related Information

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

Related commands: [retime](#)

Affects this attribute: [retime_original_registers](#) on page 951

Related attributes: [dont_retime](#) on page 842

(design) [retime](#) on page 839

(subdesign) [retime](#) on page 910

[retime_hard_region](#) on page 910

[retime_reg_naming_suffix](#) on page 819

undesirable_libcells

`undesirable_libcells libcells`

Read-write instance attribute. Specifies a list of libcells that the tool should avoid during synthesis and optimization of this hierarchical instance. The tool can still use some of these libcells if there are no other libcells available to synthesize the design.

By default, the instance attribute inherits the value of the parent instance attribute. If the values of this attribute differ for the hierarchical instances of a subdesign, these hierarchical instances get automatically unqualified. You cannot set this attribute on hierarchical instances generated by RTL Compiler (such as muxes, datapath operators, and so on).

Related Information

Affects this command: [synthesize](#)

Affected by these attributes: (design) [undesirable_libcells](#) on page 840

ungroup_ok

`ungroup_ok {true | false}`

Default: true

Read-write instance attribute. Controls whether this hierarchical instance should be ungrouped. Set the attribute to `false` to prevent the ungrouping.

Related Information

[Automatic Ungrouping in Using Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

Affected by these attributes: [auto_ungroup](#) on page 697

 (subdesign) [ungroup_ok](#) on page 911

unresolved

unresolved {false | true}

Default: false

Read-write instance attribute. Controls whether the instance should be treated as an unresolved macro in the design and whether to preserve the logical gates driving this instance.

Set this attribute to true on a hard macro to treat it as a blackbox without timing characteristics. This prevents the logic that drives the hard macro pins from being deleted during synthesis because transitive signal flow cannot be detected across the hard block in the absence of a timing library model in the technology library. (This is part of boundary optimization in RTL Compiler).



When you set the unresolved attribute to true on a hierarchical instance, the instance becomes a blackbox. The attribute setting is irreversible for the session.

Example

The following command sets the unresolved attribute to true on instance u2.

```
rc:/> set_attr unresolved true u2
Info      : Deleting the contents of subdesign. [TUI-246]
            : The subdesign is 'ibox2'.
            : The subdesign is gone and cannot be brought back.
Setting attribute of instance 'u2': 'unresolved' = true
```

Related Information

Affects this command: [synthesize](#)

hdl_arch Attributes

Contain information about user-defined modules (or entities).

- To set an hdl_arch attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_arch name]
```

- To get a an hdl_arch attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_arch name]
```

blackbox

```
blackbox {false | true}
```

Default: false

Read-write hdl_arch attribute. When set to true, this architecture will not be elaborated causing it to be written out as a blackbox (empty module with complete port information).

Note: This attribute is supported only in the RTL flow.

Related Information

[Specifying black_box Pragmas \(Verilog\) in HDL Modeling in Encounter RTL Compiler](#)

Affects this command: [elaborate](#)

Related attribute: (root) [hdl_error_on_blackbox](#) on page 733

hdl_error_on_blackbox

```
hdl_error_on_blackbox {false | true}
```

Default: false

Read-write hdl_arch attribute. When set to true, issues an error message if there is an unresolved reference (black-box) during elaboration of a particular module or hdl_architecture.

In case where the root and hdl_arch attributes are set to different values, the last specification takes precedence. See the example RTL and the corresponding schematic in [HDL-Related Attributes](#) of the *HDL Modeling in Encounter RTL Compiler* manual.

Note: This attribute is supported only in the RTL flow.

Related Information

Affects this command: [elaborate](#)

Related attribute: (root) [hdl_error_on_blackbox](#) on page 733

hdl_error_on_latch

`hdl_error_on_latch {false | true}`

Default: false

Read-write [hdl_arch](#) attribute. When set to true, issues an error message if a latch is inferred for a module/hdl_architecture.

In case where the `root` and `hdl_arch` attributes are set to different values, the last specification takes precedence. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

Note: This attribute is supported only in the RTL flow.

Related Information

Affects these commands: [elaborate](#)

Related attribute: (root) [hdl_error_on_latch](#) on page 733

hdl_error_on_logic_abstract

`hdl_error_on_logic_abstract {false | true}`

Default: false

Read-write [root](#) attribute. When set to true, issues an error message if a logic abstract is inferred for a for a module/hdl_architecture.

Related Information

[Global and User Control of Elaboration](#) in *HDL Modeling in Encounter RTL Compiler*

Affects these commands: [elaborate](#)

Related attribute: (root) [hdl_error_on_logic_abstract](#) on page 734

hdl_error_on_negedge

hdl_error_on_negedge {false | true}

Default: false

Read-write `hdl_arch` attribute. When set to `true`, issues an error message if a module or entity infers a flip-flop that is triggered by a falling clock edge.

In case the root and `hdl_arch` `hdl_error_on_negedge` attributes are set to different values, the last one takes precedence. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

Note: This attribute is supported only in the RTL flow.

Related Information

Affects these commands: [elaborate](#)

Related attribute: (root) [hdl_error_on_negedge](#) on page 735

hdl_ff_keep_explicit_feedback

hdl_ff_keep_explicit_feedback {true | false}

Default: true

Read-write `hdl_arch` attribute. Controls how flip-flop stable states are implemented for feedback assignments that are explicitly specified in the RTL. This attribute for this object type (`hdl_arch`) will only affect the `module` or `entity` represented by the particular `hdl_arch`.

In case the root and `hdl_arch` `hdl_ff_keep_explicit_feedback` attributes are set to different values, the last one takes precedence. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

Note: This attribute is ignored, and all flip-flop feedback is removed when the `lp_insert_clock_gating` attribute is set to `true`.

This attribute is supported only in the RTL flow.

Related Information

Affects this command: [elaborate](#)

Related attribute (root) [hdl_ff_keep_explicit_feedback](#) on page 735

hdl_ff_keep_feedback

```
hdl_ff_keep_feedback {false | true}
```

Default: false

Read-write `hdl_arch` attribute. Controls how flip-flop stable states are implemented. When set to `true`, implements a feedback path from the Q output to the D input. When set to `false`, uses a synchronous flip-flop enable signal to implement the stable states. This attribute for this object type (`hdl_arch`) will only affect the module or entity represented by the particular `hdl_arch`.

In case the root and `hdl_arch` `hdl_ff_keep_feedback` attributes are set to different values, the last one takes precedence. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

Note: This attribute is ignored, and all flip-flop feedback is removed when the `lp_insert_clock_gating` attribute is set to `true`.

This attribute is supported only in the RTL flow. See the example RTL and the corresponding schematic in the [HDL-Related Attributes](#) section of the *HDL Modeling in Encounter RTL Compiler* manual.

Related Information

Affects these commands: [elaborate](#)

[read_hdl](#)

Related attribute (root) [hdl_ff_keep_feedback](#) on page 736

hdl_latch_keep_feedback

```
hdl_latch_keep_feedback {false | true}
```

Default: false

Read-write `hdl_arch` attribute. Controls how explicitly-specified latch stable states (for example, `q <= q`) are implemented. When set to `true`, implements a feedback path from the Q output to the D input, resulting in a combinational loop. When set to `false`, implements a latch with an enable signal. This attribute for this object type (`hdl_arch`) will only affect the module or entity represented by the particular `hdl_arch`. See the example RTL and the corresponding schematic in [HDL-Related Attributes](#) in *HDL Modeling in Encounter RTL Compiler*.

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—hdl_arch Attributes

In case the root and hdl_arch `hdl_latch_keep_feedback` attributes are set to different values, the last one takes precedence. See the example RTL and the corresponding schematic in [HDL-Related Attributes](#) in *HDL Modeling in Encounter RTL Compiler* manual.

Note: This attribute is supported only in the RTL flow.

Related Information

Affects these commands: [elaborate](#)

[read_hdl](#)

Related attribute: (root) [hdl_latch_keep_feedback](#) on page 748

hdl_preserve_unused_flop

`hdl_preserve_unused_flop {false | true}`

Default: false

Read-write [hdl_arch](#) attribute. When set to `true`, RTL Compiler does not remove flip-flops that do not, directly or indirectly, affect any outputs. This can be used, for example, to keep flops that are only used to observe internal nets through scan chains in test mode.

This attribute only affects flops that are inferred by elaboration. It does not affect flops that are explicitly instantiated in the HDL code.

Note: This attribute is ignored in the structural flow.

Related Information

Affects these commands: [elaborate](#)

[read_hdl](#)

Related attribute: (root) [hdl_preserve_unused_flop](#) on page 755

hdl_preserve_unused_latch

```
hdl_preserve_unused_latch {false | true}
```

Default: false

Read-write `hdl_arch` attribute. When set to `true`, RTL Compiler does not remove latches that do not, directly or indirectly, affect any outputs. This can be used, for example, to keep registers that are only used to observe internal nets through scan chains in test mode.

This attribute only affects latches that are inferred by elaboration. It does not affect latches that are explicitly instantiated in the HDL code.

Note: This attribute is ignored in the structural flow.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

Related attribute: (root) [hdl_preserve_unused_latch](#) on page 757

hdl_preserve_unused_registers

```
hdl_preserve_unused_registers {false | true}
```

Default: false

Read-write `hdl_arch` attribute. When set to `true`, RTL Compiler does not remove registers (latches and flip-flops) that do not, directly or indirectly, affect any outputs. This can be used, for example, to keep registers that are only used to observe internal nets through scan chains in test mode. This attribute for this object type (`hdl_arch`) will only affect the module or entity represented by the particular `hdl_arch`.

This attribute only affects registers that are inferred by elaboration. It does not affect registers that are explicitly instantiated in the HDL code.

Note: This attribute is only supported in the RTL flow.

Related Information

Affects these commands: [elaborate](#)
 [read_hdl](#)

Related attribute: (root) [hdl_preserve_unused_registers](#) on page 759

ungroup

```
ungroup {false | true}
```

Default: false

Read-write hdl_arch attribute. Determines whether all instances of the specified entity (in VHDL) or module (in Verilog) should be inlined during elaboration.

Note: This attribute is supported only in the RTL flow.

Example

The following example specifies that all instances of module blackwidow should be inlined during elaboration:

```
rc:/> set_attribute ungroup \
      true /hdl_libraries/default/architectures/blackwidow/
```

Related Information

Related attributes: (hdl_comp) [ungroup](#) on page 869
 (hdl_impl) [ungroup](#) on page 870
 (hdl_inst) [ungroup](#) on page 871

hdl_block Attributes

Contain information about HDL blocks.

- To set an hdl_block attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_block name]
```

- To get a an hdl_block attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_block name]
```

group

```
group string
```

Read-write hdl_block attribute. In VHDL, the group attribute specifies the name of the group to which this particular VHDL block belongs. During elaboration, a level of design hierarchy is created for each group. All VHDL blocks in an entity that belong to a particular group are put into a subdesign created for that group. If this attribute is an empty string, the VHDL block remains at the same level in the design hierarchy described in the RTL code.

After using the `read_hdl` command and before using the `elaborate` command, you can change the value of `group` attributes to instruct elaboration to create an extra level of design hierarchy.

Note: This attribute is supported only in the RTL flow.

Example

Using the following RTL code:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
    port (y : out std_logic_vector (3 downto 0);
          a, b, c :in std_logic_vector (3 downto 0);
          clk : in std_logic);
end;
architecture rtl of test is
    signal p : std_logic_vector (3 downto 0);
begin
    blok : block
    begin
        p <= a and b;
    end
```

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—hdl_block Attributes

```
block;
  y <= p or c;
end;
```

And using the following commands:

```
rc> set_attr library tutorial.lbr
rc> read_hdl -vhdl test.vhd
rc> set_attr group extra [find / -hdl_block blok]
rc> elaborate
rc> write_hdl
```

Results in the following post-elaboration netlist:

```
module tst_extra (p0, a0, b0);
  input [3:0] a0, b0;
  output [3:0] p0;
  and g1 (p0[0], a0[0], b0[0]);
  and g2 (p0[1], a0[1], b0[1]);
  and g3 (p0[2], a0[2], b0[2]);
  and g4 (p0[3], a0[3], b0[3]);
endmodule

module tst (y, a, b, c);
  input [3:0] a, b, c;
  output [3:0] y;
  wire \p[0], \p[1], \p[2], \p[3];
  tst_extra tst_extra (.p0({\p[3], [\p2], \p[0]}), .a0(a), .b0(b));
  or g1 (y[0], \p[0], c[0]);
  or g2 (y[1], \p[1], c[1]);
  or g3 (y[2], \p[2], c[2]);
  or g4 (y[3], \p[3], c[3]);
endmodule
```

Related Information

Related attributes: [\(hdl_proc\) group](#) on page 872

hdl_comp Attributes

Contain information about ChipWare components.

- To set an hdl_comp attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_comp name]
```

- To get a an hdl_comp attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_comp name]
```

ungroup

```
ungroup {false | true}
```

Default: false

Read-write hdl_comp attribute. Determines whether all instances of the specified component should be inlined during elaboration.

This attribute is supported only in the RTL flow.

Example

The following example specifies that the CW_absval ChipWare component should be inlined during elaboration:

```
rc:/> set_attribute ungroup \
      true /hdl_libraries/CW/components/CW_absval
```

Related Information

Related attributes:	(hdl_arch) ungroup on page 866
	(hdl_impl) ungroup on page 870
	(hdl_inst) ungroup on page 871

hdl_impl Attributes

Contain information about implementation within the ChipWare Developer framework.

- To set an hdl_impl attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_impl name]
```

- To get a an hdl_impl attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_impl name]
```

ungroup

```
ungroup {false | true}
```

Default: false

Read-write hdl_impl attribute. Determines whether all instances of the specified component architecture should be ungrouped during elaboration.

Related Information

Related attributes:	(hdl_arch) ungroup on page 866
	(hdl_comp) ungroup on page 869
	(hdl_inst) ungroup on page 871

hdl_inst Attributes

Contain information about HDL instances.

- To set an hdl_inst attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_inst name]
```

- To get a an hdl_inst attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_inst name]
```

ungroup

```
ungroup {false | true}
```

Default: false

Read-write hdl_inst attribute. Determines whether a particular instance should be inlined during elaboration.

Note: This attribute is supported only in the RTL flow.

Related Information

Related attributes:	(hdl_arch) ungroup on page 866
	(hdl_comp) ungroup on page 869
	((hdl_impl) ungroup on page 870

hdl_proc Attributes

Contain information about the HDL processes.

- To set an hdl_proc attribute, type

```
set_attribute attribute_name attribute_value \
[find /hdl_libraries/library -hdl_proc name]
```

- To get a an hdl_proc attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_proc name]
```

group

`group string`

Read-write `hdl_proc` attribute. Specifies the name of the group to which this particular Verilog or VHDL process belongs.

In VHDL, the `group` attribute specifies the name of the group to which the specified process belongs. During elaboration a level of design hierarchy is created for each group. All processes in an entity that belong to a particular group are put into a subdesign created for that group. If this attribute is an empty string, then this process remains at the same level in the design hierarchy described in the RTL code.

In Verilog, the `group` attribute specifies the name of the group to which the specified `always` construct belongs. During elaboration a level of design hierarchy is created for each group. All `always` constructs in a module that belong to a particular group are put into a subdesign created for that group. If this attribute is an empty string, then the `always` construct remains at the same level in the design hierarchy described in the RTL code.

After using the `read_hdl` command and before using the `elaborate` command, you can change the value of `group` attributes to instruct elaboration to create an extra level of design hierarchy.

Note: This attribute is supported only in the RTL flow.

Examples

The following example shows how to use the `hdl_proc` attribute to produce a netlist for Verilog named blocks.

- Using the following Verilog RTL:

```
module test (y, a, b, c);
    input [3:0] a, b, c;
    reg [3:0] p;
    output [3:0] y;

    always @(a or b)
    begin : blok
        p = a & b;
    end
    assign y = p | c;
endmodule
```

And the following commands:

```
rc:/> set_attribute library tutorial.lbr
rc:/> read_hdl test.v
rc:/> set_attribute group extra [find / -hdl_proc blok]
rc:/> elaborate
rc:/> write_hdl
```

Provides the following post-elaboration netlist:

```
module test_extra (p0, a0, b0);
    input [3:0] a0, b0;
    output [3:0] p0;
    and g1 (p0[0], a0[0], b0[0]);
    and g2 (p0[1], a0[1], b0[1]);
    and g3 (p0[2], a0[2], b0[2]);
    and g4 (p0[3], a0[3], b0[3]);
endmodule

module test (y, a, b, c);
    input [3:0] a, b, c;
    output [3:0] y;
    wire \p[0], \p[1], \p[2], \p[3];
    test_extra test_extra (.p0({\p[3], \p[2], \p[1], \p[0]}), .a0(a), .b0(b));
    or g1 (y[0], \p[0], c[0]);
    or g2 (y[1], \p[1], c[1]);
    or g3 (y[2], \p[2], c[2]);
    or g4 (y[3], \p[3], c[3]);
endmodule
```

- As shown in the following example, if a VHDL process is labelled (given a name in the VHDL code), then it becomes an `hdl_proc` object of that name after using the `read_hdl` command. Each `hdl_proc` object has a string attribute called `group`, whose default value is an empty string (""). After using the `read_hdl` command and before elaboration, you can use the `set_attribute` command to change the value of the `group` attribute of the `hdl_proc` objects. During elaboration, all `hdl_proc` objects whose `group` attribute carries the same name are grouped as a level in the design hierarchy.

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—hdl_proc Attributes

Using the following RTL

```
library ieee;
use ieee.std_logic_1164.all;

entity tst is
    port (y : out std_logic_vector (3 downto 0);
          a, b, c : in std_logic_vector (3 downto 0) );
end;

architecture rtl of tst is
    signal p : std_logic_vector (3 downto 0);
begin
    blok : process (a, b)
    begin
        p <= a and b;
        end process;
        y <= p or c;
    end;
endmodule
```

And the following commands:

```
rc:/> set_attribute library tutorial.lbr
rc:/> read_hdl -vhdl test.vhd
rc:/> set_attribute group extra [find / -hdl_proc blk]
rc:/> elaborate
rc:/> write_hdl
```

The **find** command returns the following:

```
/hdl_libraries/default/architectures/ex3(rtl)/processes/blk
```

The **write_hdl** command writes the following post-elaboration netlist:

```
module test_extra (p0, a0, b0);
    input [3:0] a0, b0;
    output [3:0] p0;
    and g1 (p0[0], a0[0], b0[0]);
    and g2 (p0[1], a0[1], b0[1]);
    and g3 (p0[2], a0[2], b0[2]);
    and g4 (p0[3], a0[3], b0[3]);
endmodule

module tst (y, a, b, c);
    input [3:0] a, b, c;
    output [3:0] y;
    wire \p[0], \p[1], \p[2], \p[3];
    tst_extra_tst_extra (.p0({\p[3], \p[2], .a0(a), .b0(b));
    or g1 (y[0], \p[0], c[0]);
    or g2 (y[1], \p[1], c[1]);
    or g3 (y[2], \p[3], c[3]);
    or g4 (y[3], \p[3], c[3]);
endmodule
```

Related Information

Related attributes:

(**hdl_block**) **group** on page 867

hdl_subp Attributes

Contain information about the HDL programs.

- To get a an hdl_subp attribute value, type

```
get_attribute attribute_name \
[find /hdl_libraries/library -hdl_subp name]
```

map_to_module

`map_to_module string`

Read-only `hdl_subp` attribute. For valid modules, the specified function, task, or procedure is mapped to the module during elaboration. If the specified function, task, or procedure has a valid `map_to_module` pragma, this attributes keeps the name of the module specified in the pragma. It is a null string if this function, task, procedure does not carry a `map_to_module` pragma.

Note: This attribute is supported only in the RTL flow.

map_to_operator

`map_to_operator string`

Read-only `hdl_subp` attribute. For valid synthetic operators, the specified function, task, or procedure is mapped to the synthetic operator during elaboration. If the specified function, task, or procedure has a valid `map_to_operator` pragma, this attributes keeps the name of the synthetic operator specified in the pragma. It is a null if string if this function, task, or procedure does not carry a `map_to_operator` pragma.

Note: This attribute is supported only in the RTL flow.

return_port

`return_port string`

Read-only `hdl_subp` attribute. The operator output pin will supply a return value for the specified function. If the specified function has a valid `return_port_name` pragma, this attributes keeps the name of the output pin of the synthetic operator. It is a null string if this function does not carry a `return_port_name` pragma.

This attribute is supported only in the RTL flow.

Net Attributes

Contains information about the specified net. Net objects are stored in the `nets` directory.

- To set a net attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -net net_name]
```

- To get a net attribute value, type

```
get_attribute attribute_name \
[find /des*/design -net net_name]
```

preserve

```
preserve {false | true | delete_ok}
```

Default: false

Read-write `net` attribute. Controls the deletion of the net segment during synthesis. The preservation will not propagate across hierarchies unless you specify as such. You can set the following options:

delete_ok	Allows the specified net segment to be deleted during the removal of undriven or floating logic.
-----------	--

false	Allows the specified net segment to be optimized during synthesis.
-------	--

true	Prevents the specified net from being optimized during synthesis.
------	---

This setting also prevents the deletion of the logic driven by this net, excluding flip-flop and hierarchical instances.

Note: The deletion of flip-flops and hierarchical instances driven by a preserved net can be disabled by setting the `delete_flops_on_preserved_net` and `delete_hier_insts_on_preserved_net` attributes to `false`.

Note: Nets with the `preserve` attribute set to `true` could be duplicated during synthesis. Since the names and locations of nets can change during synthesis, use the `user_defined` attribute to assign names to the nets. After synthesis, query the names to find them. Hierarchical nets will not be duplicated nor will their names change during synthesis.

Related Information

[Preserving Instances and Modules in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (pin) [preserve](#) on page 886
 [user_defined](#) on page 1476

Related attributes: (design) [preserve](#) on page 838
 (instance) [preserve](#) on page 855
 (pin) [preserve](#) on page 886
 (subdesign) [preserve](#) on page 909

Pin Attributes

Contain information about the specified instance pin. A pin is a single bit connection point on an instance. Pin objects are stored in `pins_in` and `pins_out` directories.

- To set a pin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pin instance/pin]
```

- To get a pin attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pin instance/pin]
```

Note: You may need to specify more than one instance to serve as the search path that will uniquely identify the relevant pin.

boundary_optimize_constant_hier_pins

```
boundary_optimize_constant_hier_pins {inherited | true | false}
```

Default: inherited

Read-write [pin](#) attribute. Controls constant propagation through this hierarchical boundary pin. You can specify the following values:

false	Prevents constant propagation.
inherited	Inherits the value of the <code>boundary_optimize_constant_hier_pins</code> subdesign attribute.
true	Allows constant propagation.

This attribute applies only to pins of hierarchical instances.

Note: If the `boundary_opto` attribute on the corresponding subdesign is set to `false`, the setting of this attribute is ignored.

Related Information

[Setting Boundary Optimization in Using Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

Affected by these attributes: [boundary_opto](#) on page 892

(root) [boundary optimize constant hier pins](#) on page 699

(subdesign) [boundary optimize constant hier pins](#) on page 893

Related attributes: (pin) [boundary optimize equal opposite hier pins](#) on page 879

(root) [boundary optimize equal opposite hier pins](#) on page 700

(subdesign) [boundary optimize equal opposite hier pins](#) on page 894

(pin) [boundary optimize feedthrough hier pins](#) on page 881

(root) [boundary optimize feedthrough hier pins](#) on page 701

(subdesign) [boundary optimize feedthrough hier pins](#) on page 896

(pin) [boundary optimize invert hier pins](#) on page 883

(root) [boundary optimize invert hier pins](#) on page 702

(subdesign) [boundary optimize invert hier pins](#) on page 897

boundary_optimize_equal_opposite_hier_pins

`boundary_optimize_equal_opposite_hier_pins {inherited | true | false}`

Default: inherited

Read-write [pin](#) attribute. Controls collapsing of equal and opposite hierarchical boundary pins. Two hierarchical boundary pins are considered equal (opposite), if the tool determines that they always have the same (opposite or inverse) logic value. You can specify the following values:

false	Prevents collapsing of the hierarchical boundary pins.
inherited	Inherits the value of the <code>boundary_optimize_equal_opposite_hier_pins</code> subdesign attribute.
true	Allows collapsing of the hierarchical boundary pins.

This attribute applies only to pins of hierarchical instances.

Note: If the `boundary_opto` attribute on the corresponding subdesign is set to `false`, the setting of this attribute is ignored.

Related Information

[Setting Boundary Optimization](#) in *Using Encounter RTL Compiler*

Affects this command: [synthesize](#)

Affected by these attributes: [boundary_opto](#) on page 892

(root) [boundary_optimize_equal_opposite_hier_pins](#) on page 700

(subdesign) [boundary_optimize_equal_opposite_hier_pins](#) on page 894

Related attributes: [\(pin\) boundary_optimize_constant_hier_pins](#) on page 878

(root) [boundary_optimize_constant_hier_pins](#) on page 699

(subdesign) [boundary_optimize_constant_hier_pins](#) on page 893

(pin) [boundary_optimize_feedthrough_hier_pins](#) on page 881

(root) [boundary_optimize_feedthrough_hier_pins](#) on page 701

(subdesign) [boundary_optimize_feedthrough_hier_pins](#) on page 896

(pin) [boundary_optimize_invert_hier_pins](#) on page 883

(root) [boundary_optimize_invert_hier_pins](#) on page 702

(subdesign) [boundary_optimize_invert_hier_pins](#) on page 897

boundary_optimize_feedthrough_hier_pins

`boundary_optimize_feedthrough_hier_pins {inherited | true | false}`

Default: inherited

Read-write [pin](#) attribute. Controls optimization of feedthrough pins on this hierarchical pin. Hierarchical boundary pins are feedthrough pins, if output pins have always the same (or inverted) logic value as an input pin. Such feedthrough pins can be routed around the subdesign and no connections or logic is needed inside the subdesign for these pins. You can specify the following values:

false	Prevents optimization of feedthrough pins.
inherited	Inherits the value of the <code>boundary_optimize_feedthrough_hier_pins</code> subdesign attribute.
true	Allows optimization of feedthrough pins.

This attribute applies only to pins of hierarchical instances.

Note: If the `boundary_opto` attribute on the corresponding subdesign is set to `false`, the setting of this attribute is ignored.

Related Information

[Setting Boundary Optimization in Using Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

Affected by these attributes: [boundary_opto](#) on page 892

(root) [boundary_optimize_feedthrough_hier_pins](#) on page 701

(subdesign) [boundary_optimize_feedthrough_hier_pins](#) on page 896

Related attributes: (pin) [boundary_optimize_constant_hier_pins](#) on page 878

(root) [boundary_optimize_constant_hier_pins](#) on page 699

(subdesign) [boundary_optimize_constant_hier_pins](#) on page 893

(pin) [boundary_optimize_equal_opposite_hier_pins](#) on page 879

(root) [boundary optimize equal opposite hier pins](#) on page 700

(subdesign) [boundary optimize equal opposite hier pins](#) on page 894

(pin) [boundary optimize invert hier pins](#) on page 883

(root) [boundary optimize invert hier pins](#) on page 702

(subdesign) [boundary optimize invert hier pins](#) on page 897

boundary_optimize_hier_pin_invertible

`boundary_optimize_hier_pin_invertible {true | false}`

Read-only [pin](#) attribute. Indicates whether the pin can be inverted during boundary optimization.

Preserved pins, pins of preserved nets, and pins of preserved instances are not invertible. In addition, there are internal restrictions (such as multibit ports/busses, pins with timing exceptions or pins where a clock is defined on) that do not allow inversion of a pin.

Note: This attribute is set to `false` for pins that are not hierarchical boundary pins.

Related Information

Affects this command: [synthesize](#)

Affected by these attributes: [\(pin\) boundary optimize invert hier pins](#) on page 883

[\(root\) boundary optimize invert hier pins](#) on page 702

[\(subdesign\) boundary optimize invert hier pins](#) on page 897

Related attribute: [\(subport\) boundary optimize hier pin invertible](#) on page 913

boundary_optimize_invert_hier_pins

```
boundary_optimize_invert_hier_pins {inherited | true | false}
```

Default: inherited

Read-write [pin](#) attribute. Controls hierarchical boundary pin inversion on this hierarchical pin. You can specify the following values:

false	Prevents boundary pin inversion.
inherited	Inherits the value of the <code>boundary_optimize_invert_hier_pins</code> subdesign attribute.
true	Allows boundary pin inversion.

This attribute applies only to pins of hierarchical instances.

Note: If the `boundary_opto` attribute on the corresponding subdesign is set to `false`, the setting of this attribute is ignored.

Note: If this attribute is set to `true`, the `write_do_lec` command adds the following LEC command to the dofile:

```
SET Naming Rule _BAR -Inverted_pin_extension
```

This LEC command specifies the string that is appended to pin names and net names by RTL Compiler when hierarchical boundary pins are inverted during synthesis. To modify this string use the `boundary_optimize_invert_hier_pins_renaming_extension root` attribute. In the command above, the `_BAR` value corresponds to the default value of this attribute.

Related Information

[Setting Boundary Optimization in Using Encounter RTL Compiler](#)

[Interfacing with Encounter Conformal Logical Equivalence Checker in Interfacing between Encounter RTL Compiler and Encounter Conformal](#)

Affects these commands: [synthesize](#)

[write_do_lec](#)

Affected by these attributes: [boundary_opto](#) on page 892

(root) [boundary_optimize_invert_hier_pins](#) on page 702

(subdesign) [boundary optimize invert hier pins](#) on page 897

Related attributes:

(pin) [boundary optimize constant hier pins](#) on page 878

(root) [boundary optimize constant hier pins](#) on page 699

(subdesign) [boundary optimize constant hier pins](#) on page 893

(pin) [boundary optimize equal opposite hier pins](#) on page 879

(root) [boundary optimize equal opposite hier pins](#) on page 700

(subdesign) [boundary optimize equal opposite hier pins](#) on page 894

(pin) [boundary optimize feedthrough hier pins](#) on page 881

(root) [boundary optimize feedthrough hier pins](#) on page 701

(subdesign) [boundary optimize feedthrough hier pins](#) on page 896

[boundary optimize invert hier pins rename nets](#) on page 703

[boundary optimize invert hier pins renaming extension](#) on page 704

iopt_avoid_tiecell_replacement

`iopt_avoid_tiecell_replacement {false | true}`

Default: false

Read-write [pin](#) attribute. Controls whether a constant assignment on this pin can be replaced with a tie cell during incremental optimization. Set this attribute to `true` to prevent this replacement.

Related Information

- Affects this command: [synthesize -incremental](#)
- Related attributes: (port) [iopt_avoid_tiecell_replacement](#) on page 890
 (subport) [iopf_avoid_tiecell_replacement](#) on page 914

lssd_master_clock

`lssd_master_clock master_clock`

Read-write [pin](#) attribute. Identifies two pins on a hierarchical instance as the master clock and slave clock pins. The pin on which you set this attribute is the slave clock pin, while the pin you specify as the value of this attribute is the master clock pin. You must set this attribute before mapping. During mapping, RTL Compiler will make connections from the master clock pin to the master clock pins of all master-slave flip-flops driven by this slave clock.

Example

```
set_attribute lssd_master_clock i_pads/pins_out/clkm i_pads/pins_out/clks
```

This example specifies hierarchical pin `i_pads/pins_out/clkm` as the master clock, and hierarchical pin `i_pads/pins_out/clks` as the slave clock.

Related Information

- Affects this command: [synthesize](#)
- Affected by this attribute: [map_to_master_slave_lssd](#) on page 793
- Related attributes: (port) [lssd_master_clock](#) on page 890
 (subport) [lssd_master_clock](#) on page 914

preserve

```
preserve {false | true | delete_ok}
```

Default: false

Read-write [pin](#) attribute. Controls the deletion of the pin during synthesis. You can set the following options:

delete_ok	Allows the specified pin to be deleted during the removal of undriven or floating logic.
false	Allows the specified pin to be optimized during synthesis.
true	Prevents the specified pin from being optimized during synthesis.

Note: Pins with the `preserve` attribute set to `true` could be duplicated during synthesis. Since the names and locations of pins can change during synthesis, use the `user_defined` attribute to assign names to the pins. After synthesis, query the names to find them. Hierarchical pins will not be duplicated nor will their names change during synthesis.

Example

```
rc:/> set_attribute preserve true [find /des*/design -pin instance/pin]
```

Related Information

[Preserving Instances and Modules in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: ([net](#)) [preserve](#) on page 876

[user_defined](#) on page 1476

Related attributes: ([design](#)) [preserve](#) on page 838

 ([instance](#)) [preserve](#) on page 855

 ([pgpin](#)) [preserve](#) on page 888

 ([subdesign](#)) [preserve](#) on page 909

prune_unused_logic

`prune_unused_logic {true | false}`

Default: true

Read-write [pin](#) attribute. Allows pruning (removing) of logic driving a hierarchical pin if the hierarchical pin does not drive any loads.

Related Information

Affects this command: [synthesize](#)

Affects this attribute: [boundary_opto](#) on page 892

Pgpin Attributes

Contain information about power and ground instance pins.

- To set a pgpin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pgpin instance/pin]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pgpin instance/pin]
```

preserve

```
preserve {false | true | delete_ok}
```

Default: false

Read-write pgpin attribute. Controls the deletion of the power or ground pin during synthesis. You can set the following options:

delete_ok Allows the specified pin to be deleted during the removal of undriven or floating logic.

false Allows the specified pin to be optimized during synthesis.

true Prevents the specified pin from being optimized during synthesis.

Note: Pins with the preserve attribute set to true could be duplicated during synthesis. Since the names and locations of pins can change during synthesis, use the user_defined attribute to assign names to the pins. After synthesis, query the names to find them. Hierarchical pins will not be duplicated nor will their names change during synthesis.

Example

```
rc:/designs/top> get_attr preserve m1/g3/vdd
false
```

Related Information

[Preserving Instances and Modules in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by this attribute: (net) [preserve](#) on page 876

[user_defined](#) on page 1476

Related attributes: (design) [preserve](#) on page 838

 (instance) [preserve](#) on page 855

 (pin) [preserve](#) on page 886

 (subdesign) [preserve](#) on page 909

Port Attributes

Contain information about single bit connection points in the specified top-level design. Port objects are stored in `ports_in` and `ports_out` directories.

- To set a port attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -port name]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -port name]
```

iop_tiecell_replacement

```
iop_tiecell_replacement {false | true}
```

Default: false

Read-write port attribute. Controls whether a constant assignment on this port can be replaced with a tie cell during incremental optimization. Set this attribute to `true` to prevent this replacement.

Related Information

Affects this command: [synthesize -incremental](#)

Related attributes: (pin) [iop_tiecell_replacement](#) on page 884

 (subport) [iop_tiecell_replacement](#) on page 914

lssd_master_clock

```
lssd_master_clock master_clock
```

Read-write port attribute. Identifies two ports on a module as the master clock and slave clock ports. The port on which you set this attribute is the slave clock port, while the port you specify as the value of this attribute is the master clock port. You must set this attribute before mapping. During mapping, RTL Compiler will make connections from the master clock port to the master clock pins of all master-slave flip-flops driven by this slave clock.

Example

```
set_attribute lssd_master_clock clkm clk
```

This example specifies `clkm` as the master clock port and `clk` as the slave clock port.

Related Information

Affects this command:	synthesize
Affected by this attribute:	map_to_master_slave_lssd on page 793
Related attributes:	(pin) lssd_master_clock on page 885 (subport) lssd_master_clock on page 914

Subdesign Attributes

Contain information about the subdesigns in the specified design. Subdesigns correspond to Verilog modules or VHDL entities instantiated in the top-level Verilog module or top-level VHDL entity.

- To set a subdesign attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subdesign name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subdesign name]
```

boundary_opto

`boundary_opto {true | false}`

Default: true

Read-write subdesign attribute. Controls boundary optimization on the subdesign and hierarchical pin inversion.

By default, RTL Compiler performs boundary optimization during synthesis for all subdesigns in the design.

To preserve the input and output pins of a subdesign, you can turn off the boundary optimization. In this case, no hierarchical pin inversion will be done either for this subdesign.

Note: To exclude individual pins from boundary optimization, use the `preserve` attribute

Note: Setting this attribute to `false` prevents ungrouping of the subdesign.

Related Information

[Setting Boundary Optimization in Using Encounter RTL Compiler](#)

Affects this command:

[synthesize](#)

Affected by this attribute:

[prune_unused_logic](#) on page 887

Affects these attributes:

[boundary_change](#) on page 960

[boundary_optimize_constant_hier_pins](#) on page 699

[boundary_optimize_equal_opposite_hier_pins](#) on page 700

[boundary optimize feedthrough hier pins](#) on page 701

[boundary optimize invert hier pins](#) on page 702

boundary_optimize_constant_hier_pins

`boundary_optimize_constant_hier_pins {inherited | true | false}`

Default: inherited

Read-write [subdesign](#) attribute. Controls constant propagation through the hierarchical boundary pins of this subdesign.

You can specify the following values:

false	Prevents constant propagation.
inherited	Inherits the value of the <code>boundary_optimize_constant_hier_pins</code> root attribute.
true	Allows constant propagation.

Note: If the `boundary_opto` attribute on the corresponding subdesign is set to `false`, the setting of this attribute is ignored. If the `boundary_optimize_constant_hier_pins` attribute on an instance pin is set to `false` or `true` (does not have the `inherited` value), the setting of this subdesign attribute is ignored for that pin.

Related Information

[Setting Boundary Optimization](#) in *Using Encounter RTL Compiler*

Affects this command: [synthesize](#)

Affected by these attributes: [boundary_opto](#) on page 892

(root) [boundary_optimize_constant_hier_pins](#) on page 699

Affects this attribute: [\(pin\) boundary_optimize_constant_hier_pins](#) on page 878

Related attributes: [\(pin\) boundary_optimize_equal_opposite_hier_pins](#) on page 879

(root) [boundary_optimize_equal_opposite_hier_pins](#) on page 700

(subdesign) [boundary optimize equal opposite hier pins](#) on page 894

(pin) [boundary optimize feedthrough hier pins](#) on page 881

(root) [boundary optimize feedthrough hier pins](#) on page 701

(subdesign) [boundary optimize feedthrough hier pins](#) on page 896

(pin) [boundary optimize invert hier pins](#) on page 883

(root) [boundary optimize invert hier pins](#) on page 702

(subdesign) [boundary optimize invert hier pins](#) on page 897

boundary_optimize_equal_opposite_hier_pins

`boundary_optimize_equal_opposite_hier_pins {inherited | true | false}`

Default: inherited

Read-write [subdesign](#) attribute. Controls collapsing of equal and opposite hierarchical boundary pins of this subdesign. Two hierarchical boundary pins are considered equal (opposite), if the tool determines that they always have the same (opposite or inverse) logic value. You can specify the following values:

false	Prevents collapsing of the hierarchical boundary pins.
inherited	Inherits the value of the <code>boundary_optimize_equal_opposite_hier_pins</code> root attribute.
true	Allows collapsing of the hierarchical boundary pins.

Note: If the `boundary_opto` attribute on the corresponding subdesign is set to `false`, the setting of this attribute is ignored. If the `boundary_optimize_equal_opposite_hier_pins` attribute on an instance pin is set to `false` or `true` (does not have the `inherited` value), the setting of this subdesign attribute is ignored for that pin.

Related Information

[Setting Boundary Optimization](#) in *Using Encounter RTL Compiler*

Affects this command: [synthesize](#)

Affected by these attributes: [boundary_opto](#) on page 892

(root) [boundary optimize equal opposite hier_pins](#) on
page 894

Affects this attribute: [\(pin\) boundary optimize equal opposite hier_pins](#) on
page 879

Related attributes: [\(pin\) boundary optimize constant hier_pins](#) on page 878

(root) [boundary optimize constant hier_pins](#) on page 699

(subdesign) [boundary optimize constant hier_pins](#) on
page 893

(pin) [boundary optimize feedthrough hier_pins](#) on
page 881

(root) [boundary optimize feedthrough hier_pins](#) on
page 701

(subdesign) [boundary optimize feedthrough hier_pins](#) on
page 896

(pin) [boundary optimize invert hier_pins](#) on page 883

(root) [boundary optimize invert hier_pins](#) on page 702

(subdesign) [boundary optimize invert hier_pins](#) on
page 897

boundary_optimize_feedthrough_hier_pins

`boundary_optimize_feedthrough_hier_pins {inherited | true | false}`

Default: inherited

Read-write subdesign attribute. Controls optimization of feedthrough pins of this subdesign. Hierarchical boundary pins are feedthrough pins, if output pins have always the same (or inverted) logic value as an input pin. Such feedthrough pins can be routed around the subdesign and no connections or logic is needed inside the subdesign for these pins. You can specify the following values:

false	Prevents optimization of feedthrough pins.
inherited	Inherits the value of the <code>boundary_optimize_feedthrough_hier_pins</code> root attribute.
true	Allows optimization of feedthrough pins.

Note: If the `boundary_opto` attribute on the corresponding subdesign is set to `false`, the setting of this attribute is ignored. If the `boundary_optimize_feedthrough_hier_pins` attribute on an instance pin is set to `false` or `true` (does not have the `inherited` value), the setting of this subdesign attribute is ignored for that pin.

Related Information

Setting Boundary Optimization in *Using Encounter RTL Compiler*

Affects this command: [synthesize](#)

Affected by these attributes: [boundary_opto](#) on page 892

 (root) [boundary_optimize_feedthrough_hier_pins](#) on
 page 701

Affects this attribute: (pin) [boundary_optimize_feedthrough_hier_pins](#) on
 page 881

Related attributes: (pin) [boundary_optimize_constant_hier_pins](#) on page 878
 (root) [boundary_optimize_constant_hier_pins](#) on page 699
 (subdesign) [boundary_optimize_constant_hier_pins](#) on
 page 893

(pin) [boundary_optimize_equal_opposite_hier_pins](#) on page 879

(root) [boundary_optimize_equal_opposite_hier_pins](#) on page 700

(subdesign) [boundary_optimize_equal_opposite_hier_pins](#) on page 894

(pin) [boundary_optimize_invert_hier_pins](#) on page 883

(root) [boundary_optimize_invert_hier_pins](#) on page 702

(subdesign) [boundary_optimize_invert_hier_pins](#) on page 897

boundary_optimize_invert_hier_pins

`boundary_optimize_invert_hier_pins {inherited | true | false}`

Default: inherited

Read-write [subdesign](#) attribute. Controls hierarchical boundary pin inversion of this subdesign. You can specify the following values:

false	Prevents boundary pin inversion.
inherited	Inherits the value of the <code>boundary_optimize_invert_hier_pins</code> root attribute.
true	Allows boundary pin inversion.

Note: If the `boundary_opto` attribute on the corresponding subdesign is set to `false`, the setting of this attribute is ignored. If the `boundary_optimize_invert_hier_pins` attribute on an instance pin is set to `false` or `true` (does not have the `inherited` value), the setting of this subdesign attribute is ignored for that pin.

Note: If this attribute is set to `true`, the `write_do_lec` command adds the following LEC command to the dofile:

```
SET Naming Rule _BAR -Inverted_pin_extension
```

This LEC command specifies the string that is appended to pin names and net names by RTL Compiler when hierarchical boundary pins are inverted during synthesis. To modify this string use the `boundary_optimize_invert_hier_pins_renaming_extension` root attribute. In the previous command, the `_BAR` value corresponds to the default value of this attribute.

Related Information

[Setting Boundary Optimization](#) in *Using Encounter RTL Compiler*

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

Affects these commands: [synthesize](#)

[write_dolec](#)

Affected by these attributes: [boundary_opto](#) on page 892

(root) [boundary_optimize_invert_hier_pins](#) on page 702

Affects this attribute: [\(pin\) boundary_optimize_invert_hier_pins](#) on page 883

Related attributes: [\(pin\) boundary_optimize_constant_hier_pins](#) on page 878

(root) [boundary_optimize_constant_hier_pins](#) on page 699

(subdesign) [boundary_optimize_constant_hier_pins](#) on page 893

(pin) [boundary_optimize_equal_opposite_hier_pins](#) on page 879

(root) [boundary_optimize_equal_opposite_hier_pins](#) on page 700

(subdesign) [boundary_optimize_equal_opposite_hier_pins](#) on page 894

(pin) [boundary_optimize_feedthrough_hier_pins](#) on page 881

(root) [boundary_optimize_feedthrough_hier_pins](#) on page 701

(subdesign) [boundary_optimize_feedthrough_hier_pins](#) on page 896

[boundary_optimize_invert_hier_pins_rename_nets](#) on page 703

[boundary_optimize_invert_hier_pins_renaming_extension](#) on page 704

control_logic_optimization

```
control_logic_optimization {inherited | none | basic | advanced}
```

Default: inherited

Read-write subdesign attribute. Controls the optimization of control logic (described in the RTL through conditional constructs like case statements, if-then-else statements, conditional selects, and so on) in this subdesign during generic synthesis. You can specify any of the following values:

inherited	Inherits the value from the <code>control_logic_optimization</code> root attribute.
advanced	Applies advanced level optimization
basic	Applies basic optimization.
none	Turns off optimization.

Note: All control optimization transformations are verifiable. The advanced transformations might result in better QoR but can also increase the runtime. For best results, set this attribute after you have elaborated the design, but before generic synthesis.

Related Information

- Affects this command: [synthesize -to_generic](#)
Related attributes: (root) [control_logic_optimization](#) on page 707
 (design) [control_logic_optimization](#) on page 832

delete_unloaded_insts

`delete_unloaded_insts {inherited | false | true}`

Default: inherited

Read-write subdesign attribute. Controls the deletion of unloaded hierarchical instances. You can specify one of the following values:

false	Preserves any pre-existing hierarchical instances.
	Note: If you just want to maintain mapped and unmapped sequential instances, set the <code>delete_unloaded_seqs</code> attribute to <code>false</code> . Unmapped (Boolean) non-hierarchical instances cannot be rescued if they are unloaded.
inherited	Inherits the value from the <code>delete_unloaded_insts</code> root attribute.
true	Removes logic if none of the outputs are connected.

Related Information

Affects this command:	<u>synthesize</u>
Affected by this attribute:	(root) <u>delete_unloaded_insts</u> on page 708
Related attributes:	(root) <u>delete_unloaded_seqs</u> on page 709
	(design) <u>delete_unloaded_seqs</u> on page 833
	(subdesign) <u>delete_unloaded_seqs</u> on page 901

delete_unloaded_seqs

`delete_unloaded_seqs {inherited | false | true}`

Default: inherited

Read-write subdesign attribute. Controls the deletion of unloaded sequential instances in this subdesign. You can specify one of the following values:

false	Prevents the deletion of unloaded sequential instances.
inherited	Inherits the value from the <code>delete_unloaded_seqs</code> root attribute.
true	Removes flip-flops and logic if they are not transitively fanning out to output ports.

Related Information

Affects this command:	<u>synthesize</u>
Affected by this attribute:	(root) <u>delete_unloaded_seqs</u> on page 709
Related attributes:	(root) <u>delete_unloaded_insts</u> on page 708 (design) <u>delete_unloaded_seqs</u> on page 833 (subdesign) <u>delete_unloaded_insts</u> on page 900 <u>hdl_preserve_unused_registers</u> on page 759

dp_csa

`dp_csa {inherited | basic | none}`

Default: inherited

Read-write subdesign attribute. Controls the carry save adder (CSA) transformations within the subdesign. CSA opportunities outside the specified subdesign are unaffected. You can specify one of the following values:

basic	Applies basic transformation.
inherited	Inherits the value from the <code>dp_csa</code> design attribute.
none	Turns off CSA transformation.

Related Information

[Controlling CSA Transformations in Datapath Synthesis in Encounter RTL Compiler](#)

Affects this command: [synthesize -to_generic](#)

Related attributes: (design) [dp_csa](#) on page 833
(root) [dp_csa](#) on page 711

dp_rewriting

`dp_rewriting {inherited | none | basic | advanced}`

Default: inherited

Read-write [subdesign](#) attribute. Controls how the datapath rewriting optimization is applied onto this module during `synthesize -to_generic -effort high`. If the value of this attribute is `inherited`, the effort level of the optimization at the specified module will be identical to (“inherited from”) the root level attribute of the same name.

advanced	Applies advanced level optimization
basic	Applies basic optimization.
inherited	Inherits the value from the <code>dp_rewriting</code> root attribute.
none	Turns off optimization.

Examples

- The following example turns on this optimization only on certain set of modules:

```
rc:/> set_attribute dp_rewriting none /  
rc:/> set_attribute dp_rewriting basic $list_of_subd_vdirs
```

- The following example turns off this optimization only on a certain set of modules:

```
rc:/> set_attribute dp_rewriting none $list_of_subd_vdirs
```

Related Information

Affects this command: [synthesize -to_generic -effort high](#)

Related attributes: (root) [dp_rewriting](#) on page 712
(design) [dp_rewriting](#) on page 834

dp_sharing

```
dp_sharing {inherited | none | basic | advanced}
```

Default: inherited

Read-write subdesign attribute. Controls resource sharing on the subdesign during synthesize -to_generic -effort high. If the value of this attribute is inherited, the effort level of the optimization on the design will be identical to (“inherited from”) the root level attribute.

inherited	Inherits the value from the root level value of <u>dp_sharing</u> .
advanced	Applies advanced level optimization
basic	Applies basic optimization.
none	Turns off optimization.

Example

The following command turns RTL sharing transformations off on subdesign mid:

```
rc:/> set_attribute dp_sharing none {find / -subdesign mid}
```

Related Information

Controlling Sharing Transformations in Datapath Synthesis in Encounter RTL Compiler

Affects this command: synthesize -to_generic -effort high

Related attributes: (root) dp_sharing on page 713

 (design) dp_sharing on page 835

dp_speculation

```
dp_speculation {inherited | basic | none}
```

Default: inherited

Read-write subdesign attribute. Controls RTL speculation (unsharing) transformations within a particular subdesign. You can specify one of the following values:

basic	Applies basic RTL speculation transformation.
inherited	Inherits the value from the dp_speculation design attribute.
none	Turns off RTL speculation transformation.

Example

The following command turns off RTL speculation transformations within subdesign mid:

```
rc:/> set_attribute dp_speculation none [find /designs* -subdesign mid]
```

Related Information

[Controlling Speculation Transformations in Datapath Synthesis in Encounter RTL Compiler](#)

Affects this command: [synthesize -to_generic -effort high](#)

Related attributes: [\(design\) dp_speculation](#) on page 836

[\(root\) dp_speculation](#) on page 714

dp_verify_ok

```
dp_verify_ok {false | true}
```

Default: false

Read-write subdesign attribute. When enabled, eases verification by limiting few optimizations on CSA modules such as boundary optimization on CSA hierarchies and avoiding ungrouping of simple CSA and modules in the fanout of CSA trees. Due to potential QoR impact, set this attribute to true only on specific verification-challenged user-subdesigns which contain CSA logic. This will enable the Encounter® Conformal® Logical Equivalence Checking (LEC) tool to do a better job of datapath learning and successful verification.

Example

Consider the following module.

```
module csa_add_mux_add(a, b, c, d, e, f, g, s, z);
  input [7:0] a, b, c, d, e, f, g;
  input s;
  output [10:0] z;
  wire [9:0] tmp;

  assign tmp = s ? a+b+f : c+d+g;
  assign z = tmp + e;

endmodule
```

When you use the default setting, all CSA trees will be ungrouped after mapping. When you set this attribute to `true`, none of the CSA trees will be ungrouped.

Related Information

Affects this command: [synthesize -to_generic -effort {medium| high}](#)
Related attribute: (design) [dp_verify_ok](#) on page 836

hdl_cw_list

```
hdl_cw_list {{language library component}...}
```

Read-write `subdesign` attribute. Returns a Tcl list of Tcl lists. There can be as many Tcl lists as there are types of Chipware components in the subdesign or module. Each Tcl list contains three elements:

- The `language` of the module in which the ChipWare component is instantiated
- The name of the ChipWare `library` that contains the ChipWare component
- The name of the ChipWare `component`

Note: Each ChipWare component used in the subdesign appears just once in the `hdl_cw_list` attribute, no matter how many times it is instantiated.

Example

In the following example, component `DW01_add` is instantiated in module `test`, written in Verilog-2001.

```
rc:/designs/mydesign/subdesigns/test> get_attribute hdl_cw_list
{-v2001 DW01 DW01_add}
```

Related Information

Affected by this command: [elaborate](#)

Related attribute: (design) [hdl_cw_list](#)

minimize_uniquify

`minimize_uniquify {false | true}`

Default: false

Read-write [subdesign](#) attribute. Controls uniquification of this subdesign.

By default, the tool uniquifies multiple instantiations with different context (timing, constants, and so on) to deliver the best QoR.

To limit the scenarios of uniqueness of a specific subdesign with a trade-off against QoR, set this attribute to true. This will not necessarily prevent uniqueness due to algorithmic considerations based on the design topology.

Related Information

Affects this command: [synthesize -incr](#)

Affected by this attribute: (root) [minimize_uniquify](#) on page 796

multibit_allow_async_phase_map

```
multibit_allow_async_phase_map {inherited | true | false}
```

Default: inherited

Read-write subdesign attribute. Controls whether phase inversion is allowed during multibit cell inferencing in this subdesign. You can specify one of the following values:

false	Prevents phase inversion on the subdesign during multibit mapping.
inherited	Inherits the value from the <code>multibit_allow_async_phase_map</code> root attribute.
true	Allows phase inversion on the subdesign during multibit mapping.

Examples

Consider the following netlist after mapping and before incremental optimization: The library contains the multibit cell DUALDFSF.

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    DFCF cnt_0_reg[0] (.CDN (resetn), .CP (clk), .D(in[0]), .Q (out[0]));
    DFCF cnt_0_reg[1] (.CDN (resetn), .CP (clk), .D(in[1]), .Q (out[1]));
    DFCF cnt_0_reg[2] (.CDN (resetn), .CP (clk), .D(in[2]), .Q (out[2]));
    DFCF cnt_0_reg[3] (.CDN (resetn), .CP (clk), .D(in[3]), .Q (out[3]));
endmodule
module top(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    test t(resetn,clk,in,out);
endmodule
```

- When the `multibit_allow_async_phase_map` attribute is not set for subdesign `test`, the subdesign inherits the value from the root attribute (default is `true`). With phase inversion allowed, the netlist is modified as follows:

```
module test(resetn, clk, in, out);
    input resetn, clk;
    input [3:0] in;
    output [3:0] out;
    wire resetn, clk;
    wire [3:0] in;
    wire [3:0] out;
    DUALDFSF \CDN_MBIT_cnt_0_reg[0_1] (.SDN (resetn), .CP (clk), .D1(in[0]),
        .D2 (n_31), .Q1 (out[0]), .Q2 (n_32));
    INV g5(.I (in[1]), .ZN (n_31));
    INV g6(.I (n_32), .ZN (out[1]));
    DUALDFSF \CDN_MBIT_cnt_0_reg[2_3] (.SDN (resetn), .CP (clk), .D1(n_43),
        .D2 (in[3]), .Q1 (n_44), .Q2 (out[3]));
    INV g23(.I (in[2]), .ZN (n_43));
    INV g24(.I (n_44), .ZN (out[2]));
endmodule
```

- When the `multibit_allow_async_phase_map` subdesign attribute is explicitly set to `false`, the netlist does not change because phase inversion is prevented in this subdesign during multibit mapping.

Related Information

Mapping to Multibit Cells in *Encounter RTL Compiler Synthesis Flows*

Affects this command:	synthesize -incremental
Related attributes:	cell_bitwidth on page 191
	(root) multibit allow async phase map on page 796
	(instance) multibit allow async phase map on page 849
	multibit_prefix_string on page 801
	use_multibit_cells on page 826

preserve

```
preserve {false | true | const_prop_delete_ok | const_prop_size_delete_ok  
| delete_ok | map_size_ok | size_ok | size_delete_ok}
```

Default: false

Read-write subdesign attribute. Controls the optimization of the subdesign. You can set these options:

const_prop_delete_ok	Allows deleting a mapped subdesign and its instances, and constant propagation through the mapped subdesign and its instances, but not resizing, renaming or remapping.
const_prop_size_delete_ok	Allows deleting and resizing a mapped subdesign and its instances, or constant propagation through a mapped subdesign and its instances, but not renaming or remapping.
delete_ok	Allows deleting a mapped subdesign and its instances during optimization, but not resizing, renaming, or remapping it.
false	Allows logic changes to the subdesign and its instances during optimization.
map_size_ok	Allows resizing, unmapping, and remapping of the mapped sequential instances in the subdesign during optimization, but not renaming or deleting them.
size_delete_ok	Allows resizing or deleting a subdesign and its instances during optimization, but not renaming or remapping it.
size_ok	Allows resizing a mapped subdesign and its instances during optimization, but not deleting, renaming, or remapping it.
true	Prevents any logic changes in the mapped subdesign while allowing mapping optimizations to be performed in surrounding logic.

Related Information

Affects this command:

[synthesize](#)

Affected by this attribute:

(design) [preserve](#) on page 838

retime

```
retime {false | true}
```

Default: false

Read-write subdesign attribute. Marks the specified subdesign to be retimed during synthesis. This attribute must be set after the `elaborate` command but before the `synthesize` command.

Example

The following marks the `m1_sub` design for pipelining:

```
rc:/> set_attribute retime true [find / -subdesign m1_sub]  
Setting attribute of m1_sub: 'retime' = true
```

Related Information

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

Affects these commands:

[retime](#)

[synthesize](#)

Affects these attributes:

[dont_retime](#) on page 842

[\(design\) retime](#) on page 839

[retime_hard_region](#) on page 910

[retime_reg_naming_suffix](#) on page 819

retime_hard_region

```
retime_hard_region {false | true}
```

Default: false

Read-write subdesign attribute. Prevents registers from being moved across the subdesign boundaries during retiming optimization.

Example

The following example prevents all the registers in all the subdesigns from being moved across their respective boundaries:

```
rc:/> set_attribute retime_hard_region true [find / -subdesign *]
```

Related Information

Retiming the Design in *Using Encounter RTL Compiler*

Affects this command:	retime
Affects these attributes:	dont_retime on page 842 (design) retime on page 839 (subdesign) retime on page 910
Related attribute:	retime_reg_naming_suffix on page 819

user_sub_arch

`user_sub_arch {booth | non_booth | radix8}`

Read-write [subdesign](#) attribute. Controls the partial product encoding scheme that you want the tool to use for a multiplier, which affects the number of partial products generated.

Example

```
rc:/> set_attribute user_sub_arch booth [find /designs* -subdesign name]
```

Related Information

Controlling Sub-Architecture Selection in *Datapath Synthesis in Encounter RTL Compiler*

Affects this command:	synthesize
Affects this attribute:	sub_arch on page 172

ungroup_ok

`ungroup_ok {true | false}`

Default: true

Read-write [subdesign](#) attribute. Controls ungrouping of the hierarchical instances of this subdesign during automatic ungrouping. Set the attribute to `false` to prevent the ungrouping of the subdesign and its instances.

Related Information

Automatic Ungrouping in *Using Encounter RTL Compiler*

Affects this command: [synthesize](#)

Affects this attribute: (instance) [ungroup_ok](#) on page 858

Affected by this attribute: [auto_ungroup](#) on page 697

Subport Attributes

Contain information about subports in the specified design. A subport is a single bit connection point within a hierarchical instance, that is a module or entity that has been instantiated.

- To set a subport attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subport hier_instance/name]
```

- To get a subport attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subport hier_instance/name]
```

Note: You may need to specify more than one (hierarchical) instance to serve as the search path that will uniquely identify the relevant subport.

boundary_optimize_hier_pin_invertible

```
boundary_optimize_hier_pin_invertible {true | false}
```

Read-only subport attribute. Indicates whether the subport can be inverted during boundary optimization.

Preserved pins, pins of preserved nets, and pins of preserved subdesigns are not invertible. In addition, there are internal restrictions (such as multibit ports/busses, pins with timing exceptions or pins where a clock is defined on) that do not allow inversion of a pin.

Related Information

Affects this command:	<u>synthesize</u>
Affected by these attributes:	<u>boundary_opto</u> on page 892 (pin) <u>boundary_optimize_invert_hier_pins</u> on page 883 (root) <u>boundary_optimize_invert_hier_pins</u> on page 702 subdesign) <u>boundary_optimize_invert_hier_pins</u> on page 897
Related attribute:	(pin) <u>boundary_optimize_hier_pin_invertible</u> on page 882

iopt_avoid_tiecell_replacement

```
iopt_avoid_tiecell_replacement {false | true}
```

Default: false

Read-write subport attribute. Controls whether a constant assignment on this subport can be replaced with a tie cell during incremental optimization. Set this attribute to `true` to prevent this replacement.

Related Information

Affects this command: [synthesize -incremental](#)

Related attributes: (pin) [iopt_avoid_tiecell_replacement](#) on page 884

 (port) [iopt_avoid_tiecell_replacement](#) on page 890

lssd_master_clock

```
lssd_master_clock master_clock
```

Read-write subport attribute. Identifies two subports on a module as the master clock and slave clock ports. The subport on which you set this attribute is the slave clock port, while the subport you specify as the value of this attribute is the master clock port. You must set this attribute before mapping. During mapping, RTL Compiler will make connections from the master clock port to the master clock pins of all master-slave flip-flops in the module driven by this slave clock.

Example

```
set_attribute lssd_master_clock clk_m clk
```

This example specifies `clk_m` as the master clock port and `clk` as the slave clock port.

Related Information

Affects this command: [synthesize](#)

Affected by this attribute: [map_to_master_slave_lssd](#) on page 793

Related attributes: (pin) [lssd_master_clock](#) on page 885

 (port) [lssd_master_clock](#) on page 890

Clock Attributes

Contain information about clock objects in the specified design.

- To set a `clock` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -clock myclock]
```

- To get a `clock` attribute value, type

```
get_attribute attribute_name [find /des*/design -clock myclock]
```

`clock_library_cells`

```
clock_library_cells cell_list
```

Read-write `clock` attribute. Defines the list of library cells that can be used in the clock-path logic of the specified clock(s).

Set this attribute before you run the `remap_to_dedicated_clock_library` command.

Example

- The following command specifies that libcell `x1` can be used in the clock-path logic for clock `CLK` in `design test`.

```
set_attribute clock_library_cells x1 \
/designs/test/timing/clock_domains/domain_1/CLK
```

- The following commands specify that libcells `x1`, `x2`, and `x3` can be used in the clock-path logic for clocks `CLK` and `CLK1` in `design test`.

```
set_attribute clock_library_cells {x1 x2 x3} \
/designs/test/timing/clock_domains/domain_1/CLK
set_attribute clock_library_cells {x1 x2 x3} \
/designs/test/timing/clock_domains/domain_1/CLK1
```

or you can use one command:

```
set_attribute clock_library_cells {x1 x2 x3} \
/designs/test/timing/clock_domains/domain_1/CLK \
/designs/test/timing/clock_domains/domain_1/CLK1
```

Related Information

Affects this command:

[remap_to_dedicated_clock_library](#)

Attribute Reference for Encounter RTL Compiler

Elaboration and Synthesis—Clock Attributes

Analysis

Root Attributes

- [active operating conditions](#) on page 928
- [report timing show wire length](#) on page 928
- [show wns in log](#) on page 929

Design Attributes

- [arch_name](#) on page 931
- [area](#) on page 931
- [average_net_length](#) on page 932
- [cell_area](#) on page 932
- [cell_count](#) on page 932
- [constant_0_nets](#) on page 933
- [constant_1_nets](#) on page 933
- [entity_name](#) on page 933
- [hdl_elab_command_params](#) on page 934
- [hdl_parameters](#) on page 935
- [hdl_vdp_list](#) on page 936
- [language](#) on page 937
- [library_name](#) on page 937
- [logic_abstract](#) on page 937
- [max_cap_cost](#) on page 938

Attribute Reference for Encounter RTL Compiler

Analysis—List

- [max_fanout_cost](#) on page 938
- [max_trans_cost](#) on page 939
- [min_cap_cost](#) on page 939
- [min_fanout_cost](#) on page 940
- [min_trans_cost](#) on page 940
- [net_area](#) on page 940
- [physical_cell_area](#) on page 941
- [pin_count](#) on page 941
- [seq_reason_deleted](#) on page 941
- [slack](#) on page 942
- [slack_by_mode](#) on page 943
- [tns](#) on page 944
- [total_net_length](#) on page 945
- [wireload](#) on page 945

Instance Attributes

- [buffer](#) on page 946
- [cell_area](#) on page 946
- [combinational](#) on page 947
- [exceptions](#) on page 947
- [flop](#) on page 948
- [hdl_instantiated](#) on page 948
- [hierarchical](#) on page 948
- [inverted_phase](#) on page 948
- [inverter](#) on page 949
- [latch](#) on page 949
- [libcell](#) on page 949

Attribute Reference for Encounter RTL Compiler

Analysis—List

- [loop breaker](#) is ignored for all modes on page 950
- [module](#) on page 950
- [negative edge clock](#) on page 950
- [pin count](#) on page 951
- [primitive function](#) on page 951
- [retime original registers](#) on page 951
- [sequential](#) on page 952
- [slack](#) on page 952
- [subdesign](#) on page 953
- [timing case disabled arcs](#) on page 953
- [timing case disabled arcs by mode](#) on page 954
- [timing model](#) on page 955
- [tristate](#) on page 955
- [unique versions](#) on page 956

Constant Attributes

- [pin capacitance](#) on page 957
- [unique versions](#) on page 958
- [wire capacitance](#) on page 958
- [wire length](#) on page 959
- [wire resistance](#) on page 959

Pin Attributes

- [boundary change](#) on page 960
- [capturer](#) on page 962
- [causes ideal net](#) on page 962
- [clock sense negative](#) on page 963

Attribute Reference for Encounter RTL Compiler

Analysis—List

- [clock_sense_positive](#) on page 964
- [clock_sense_stop_propagation](#) on page 964
- [clock_sources_inverted](#) on page 965
- [clock_sources_non_inverted](#) on page 966
- [connect_delay](#) on page 966
- [direction](#) on page 967
- [drivers](#) on page 967
- [endpoint](#) on page 967
- [exceptions](#) on page 968
- [external_delays](#) on page 968
- [external_delays_by_mode](#) on page 969
- [has_min_delay](#) on page 971
- [endpoint](#) on page 968
- [launcher](#) on page 971
- [libpin](#) on page 972
- [loads](#) on page 972
- [min_slew](#) on page 972
- [min_timing_arcs](#) on page 973
- [net](#) on page 973
- [pin_capacitance](#) on page 974
- [propagated_clocks](#) on page 974
- [propagated_clocks_by_mode](#) on page 976
- [propagated_ideal_network](#) on page 978
- [rf_slack](#) on page 979
- [slack](#) on page 980
- [slack_by_mode](#) on page 980
- [slew](#) on page 982

Attribute Reference for Encounter RTL Compiler

Analysis—List

- [slew_by_mode](#) on page 982
- [startpoint](#) on page 983
- [timing_arcs](#) on page 983
- [timing_case_computed_value](#) on page 983
- [timing_case_computed_value_by_mode](#) on page 985
- [timing_info](#) on page 987
- [timing_info_favor_startpoint](#) on page 990
- [unique_versions](#) on page 991
- [wire_capacitance](#) on page 991
- [wire_length](#) on page 992
- [wire_resistance](#) on page 992
- [wireload_model](#) on page 992

Pgpin Attributes

- [direction](#) on page 993
- [drivers](#) on page 993
- [libpin](#) on page 994
- [loads](#) on page 994
- [net](#) on page 994
- [unique_versions](#) on page 994

Pin Bus Attributes

- [bits](#) on page 996

Net Attributes

- [constant](#) on page 997
- [driven_by_supply0](#) on page 997
- [driven_by_supply1](#) on page 998

Attribute Reference for Encounter RTL Compiler

Analysis—List

- [drivers](#) on page 999
- [ideal](#) on page 999
- [is net part of bus](#) on page 1000
- [loads](#) on page 1000
- [num drivers](#) on page 1000
- [num loads](#) on page 1000
- [type](#) on page 1001
- [unique versions](#) on page 1001

Port Attributes

- [bus](#) on page 1002
- [capturer](#) on page 1002
- [causes ideal net](#) on page 1003
- [clock sources inverted](#) on page 1004
- [clock sources non inverted](#) on page 1004
- [connect delay](#) on page 1005
- [direction](#) on page 1005
- [drivers](#) on page 1005
- [endpoint](#) on page 1006
- [exceptions](#) on page 1006
- [external delays](#) on page 1007
- [external delays by mode](#) on page 1007
- [has min delay](#) on page 1009
- [launcher](#) on page 1009
- [loads](#) on page 1010
- [min slew](#) on page 1010
- [min port delay](#) on page 1011

Attribute Reference for Encounter RTL Compiler

Analysis—List

- [net](#) on page 1011
- [pin capacitance](#) on page 1012
- [port delay](#) on page 1012
- [propagated clocks](#) on page 1013
- [propagated clocks by mode](#) on page 1015
- [rf slack](#) on page 1017
- [slack](#) on page 1017
- [slack by mode](#) on page 1018
- [slew](#) on page 1019
- [slew by mode](#) on page 1019
- [startpoint](#) on page 1020
- [timing case computed value](#) on page 1020
- [timing case computed value by mode](#) on page 1021
- [timing info](#) on page 1023
- [timing info favor startpoint](#) on page 1025
- [unique versions](#) on page 1025
- [wire capacitance](#) on page 1026
- [wire length](#) on page 1026
- [wire resistance](#) on page 1027

Port Bus Attributes

- [bits](#) on page 1028
- [direction](#) on page 1028
- [order](#) on page 1029

Subdesign Attributes

- [arch name](#) on page 1030

Attribute Reference for Encounter RTL Compiler

Analysis—List

- [area](#) on page 1030
- [cell_area](#) on page 1031
- [cell_count](#) on page 1031
- [constant_0_nets](#) on page 1031
- [constant_1_nets](#) on page 1032
- [entity_name](#) on page 1109
- [hdl_elab_param_override](#) on page 1033
- [hdl_parameters](#) on page 1033
- [hdl_pipeline_comp](#) on page 1035
- [hdl_vdp_list](#) on page 1113
- [instances](#) on page 1115
- [language](#) on page 1115
- [library_name](#) on page 1036
- [logic_abstract](#) on page 1036
- [logical_hier](#) on page 1037
- [net_area](#) on page 1037
- [physical_cell_area](#) on page 1037
- [pin_count](#) on page 1038
- [wireload](#) on page 1038

Subport Attributes

- [bus](#) on page 1039
- [causes_ideal_net](#) on page 1039
- [direction](#) on page 1040
- [drivers](#) on page 1040
- [loads](#) on page 1041
- [net](#) on page 1041

Attribute Reference for Encounter RTL Compiler

Analysis—List

- [pin_capacitance](#) on page 1041
- [unique_versions](#) on page 1042
- [wire_capacitance](#) on page 1042
- [wire_length](#) on page 1043
- [wire_resistance](#) on page 1043

Support Bus Attributes

- [bits](#) on page 1044
- [direction](#) on page 1044
- [order](#) on page 1045

Timing Bin Attributes

- [is_sub_bin](#) on page 1046
- [path_count](#) on page 1046
- [root](#) on page 1047

Timing Path Attributes

- [bin](#) on page 1048
- [end_point](#) on page 1048
- [exceptions](#) on page 1049
- [mode](#) on page 1049
- [slack](#) on page 1050
- [startpoint](#) on page 1050

Clock Attributes

- [clock_sense_negative](#) on page 1051
- [clock_sense_positive](#) on page 1052
- [clock_sense_stop_propagation](#) on page 1052

Attribute Reference for Encounter RTL Compiler

Analysis—List

- [divide_fall](#) on page 1053
- [divide_period](#) on page 1053
- [divide_rise](#) on page 1054
- [divide_waveform](#) on page 1054
- [exceptions](#) on page 1055
- [fall](#) on page 1056
- [period](#) on page 1056
- [rise](#) on page 1056
- [waveform](#) on page 1057

Clock Domain Attributes

- [exceptions](#) on page 1058

Cost Group Attributes

- [exceptions](#) on page 1059
- [slack](#) on page 1060
- [slack_by_mode](#) on page 1060
- [tns](#) on page 1062

Exception Attributes

- [adjust_value](#) on page 1063
- [cost_group](#) on page 1063
- [delay_value](#) on page 1063
- [domain](#) on page 1064
- [exception_type](#) on page 1064
- [from_points](#) on page 1065
- [lenient](#) on page 1065
- [paths](#) on page 1066

Attribute Reference for Encounter RTL Compiler

Analysis—List

- [precluded_path](#) [adjusts](#) on page 1066
- [priority](#) on page 1066
- [shift_capture](#) on page 1067
- [shift_launch](#) on page 1067
- [through_points](#) on page 1067
- [to_points](#) on page 1068

External Delay Attributes

- [clock](#) on page 1069
- [clock_rise](#) on page 1069
- [delay](#) on page 1069
- [exceptions](#) on page 1070
- [external_delay_pins](#) on page 1070
- [input_delay](#) on page 1071
- [level_sensitive](#) on page 1071

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

active_operating_conditions

`active_operating_conditions string`

Read-only `root` attribute. Returns the complete path to the operating conditions that were set with the `operating_conditions` attribute and forces the tool to load the operating conditions if they were not loaded yet or if they became invalid. Any problems found while loading are reported.

If the operating conditions cannot be loaded, then the attribute returns an empty string. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Affects these commands: [synthesize](#)

[report timing](#)

Affected by these attributes: [library](#) on page 273

[operating_conditions](#) on page 533

Related attribute: (library_domain) [active_operating_conditions](#) on page 1418

report_timing_show_wire_length

`report_timing_show_wire_length {false | true}`

Default: `false`

Read-write `root` attribute. Controls the display of a wire length column in the physical timing report.

Attribute Reference for Encounter RTL Compiler

Analysis—Root Attributes

Related Information

Affects this command: [report timing -physical](#)

show_wns_in_log

```
show_wns_in_log {true | false}
```

Default: true

Read-write root attribute. Controls the display of additional columns for the worst negative slack (WNS) value and the corresponding cost-group to the global mapping status, global incremental optimization status, and incremental optimization status in the log file during synthesis of a design with multiple cost-groups.

Example

The following is an extract of the log file for a design with four cost groups.

```
Global mapping target info
=====
Cost Group 'cg4' target slack: -130 ps
Target path end-point (Port: add/out[31])
...
...
Cost Group 'cg1' target slack: -1340 ps
Target path end-point (Port: add/out[14])

Global mapping status
=====


| Operation  | Area | Group |       |       | Group | Worst            | Path |
|------------|------|-------|-------|-------|-------|------------------|------|
|            |      | Total | Worst | Neg   |       |                  |      |
| global_map |      | 456   | -3163 | -1357 | cg1   | a[1] --> out[14] |      |



Global incremental target info
=====
Cost Group 'cg4' target slack: -224 ps
Target path end-point (Port: add/out[31])
...
...
Global incremental optimization status
=====


| Operation  | Area | Group |       |       | Group | Worst            | Path |
|------------|------|-------|-------|-------|-------|------------------|------|
|            |      | Total | Worst | Neg   |       |                  |      |
| global_inc |      | 458   | -3136 | -1350 | cg1   | a[1] --> out[14] |      |


```

Attribute Reference for Encounter RTL Compiler

Analysis—Root Attributes

Incremental optimization status

Group							
Operation	Total	DRC	Total	Worst	Worst		
	Area	Worst	Max	Neg	Cost	Group	Worst Path
init_iopt	458	-3136	0	-1350	cg1 a[1] --> out[14]		

Incremental optimization status

Group							
Operation	Total	DRC	Total	Worst	Worst		
	Area	Worst	Max	Neg	Cost	Group	Worst Path
init_delay	466	-3034	0	-1338	cg1 a[1] --> out[12]		
init_drc	466	-3034	0	-1338	cg1 a[1] --> out[12]		
init_area	466	-3034	0	-1338	cg1 a[1] --> out[12]		
....							

Related Information

Affects this command: [synthesize](#)

Design Attributes

Contain information about the specified design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design
```

or

```
set_attribute attribute_name attribute_value [find /des* -design name]
```

- To get a design attribute value, type

```
get_attribute attribute_name design
```

arch_name

`arch_name string`

Read-only `design` attribute. Returns the name of the Verilog module or the VHDL architecture from which the design is derived.

Related Information

Related command: [read_hdl](#)

Related attribute: (subdesign) [arch_name](#) on page 1030

area

`area float`

Read-only `design` attribute. Computes the total area of the design, including the net area. The area is specified in terms of the library units. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report area](#)

Related attributes: (libcell) [area](#) on page 188

(subdesign) [area](#) on page 1030

average_net_length

`average_net_length float`

Read-only design attribute. Computes the average net length of the design in microns. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related commands:

[generate_reports](#)
[report_qor](#)
[summary_table](#)

cell_area

`cell_area string`

Read-only design attribute. Returns the cell area of a design, which is specified in library units. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command:

[report_area](#)

Related attributes:

(libcell) [area](#) on page 188
(instance) [cell_area](#) on page 947
(subdesign) [cell_area](#) on page 1031

cell_count

`cell_count integer`

Read-only design attribute. Returns the cell count of the design. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report area](#)

Related attributes: (subdesign) [cell_count](#) on page 1031

constant_0_nets

`constant_0_nets list_of_nets`

Read-only [design](#) attribute. Returns a list of nets driven by constant 0.

Related Information

Related attributes: (subdesign) [constant_0_nets](#) on page 1031

constant_1_nets

`constant_1_nets list_of_nets`

Read-only [design](#) attribute. Returns a list of nets driven by constant 1.

Related Information

Related attributes: (subdesign) [constant_1_nets](#) on page 1032

entity_name

`entity_name string`

Read-only [design](#) attribute. Returns the name of the Verilog module or the VHDL entity from which the design is derived.

Related Information

Related command: [read_hdl](#)

Related attribute: (subdesign) [entity_name](#) on page 1032

hdl_elab_command_params

`hdl_elab_command_params {{parameter value}...}`

Read-only design attribute. Returns a Tcl-list of the {parameter value} pairs that were specified as value for the -parameters option of the most recently executed elaborate command.

Example

```
rc:/> elaborate -parameters {width 4} test
....
rc:/> get_attribute hdl_elab_command_params /designs/test
{{width 4}}
```

Related Information

Affects this command: [elaborate](#)

Related attribute: (subdesign) [hdl_elab_command_params](#) on page 1032

hdl_parameters

```
hdl_parameters {{parameter current_value cmdset default_value}...}
```

Read-only design attribute. Returns a Tcl list of Tcl lists. There can be as many Tcl lists as there are parameters and localparams in the top module. Each Tcl list contains four elements:

- the parameter or localparam name
- the current parameter value
- a binary number

If the binary number is 1, it indicates that the parameter value was set through the elaborate -parameters command. For localparam, the *status* value will always be 0 because it cannot be overwritten.

- The default value of the parameter before it is overridden during instantiation or using the elaborate command.

Example

The following example RTL defines a local parameter *w* and a parameter *y*.

```
module topdes (z, i, a, b);
    localparam w = 8;
    parameter y = 8;
    defparam u1.W = w;
    output [w-1:0] z;
    input [w-1:0] i;
    output [y-1:0] b;
    input [y-1:0] a;
    subdes u1 (.Z(z), .I(i));
    subdes #(y) u2 (.Z(b), .I(a));
endmodule

module subdes (Z,I);
    parameter W = 8;
    parameter LS = 1;
    localparam LSB = 1;
    output [W-LSB:0] Z;
    input [W-LS:0] I;
    assign Z = I + 1;
endmodule // subdes
```

Using the following script the value of parameter *y* is changed from 8 to 4.

```
set_attr library tutorial.lbr
read_hdl -v2001 {subdes.v topdes.v}
elaborate -param {{y 4}} topdes
```

The following command shows that parameter *y* now has the value 4 and the value was set at elaboration. The parameter *w* has the value 8 and was not overwritten.

```
rc:/> get_att hdl_parameters [find / -design *]
{w 8 0 8} {y 4 1 8}
```

Related Information

Affected by this command: [elaborate](#)

Related attribute: (subdesign) [hdl_parameters](#) on page 1033

hdl_vdp_list

`hdl_vdp_list string`

Read-only [design](#) attribute. Returns the list of RTL Compiler-supported datapath function primitives which have been instantiated in the design.

Example

Consider the following RTL.

```
module top(in1, out1, out2, out3);
    input signed [7:0] in1;
    output [7:0] out1;
    parameter bits = 4;
    output [3:0] out2;
    output [3:0] out3;

    assign out1 = $round(in1, bits);
    sub s1(in1, out2, out3);
endmodule //top

module sub(a, b, c);
    input signed [7:0] a;
    output [3:0] b;
    output [3:0] c;

    assign b = $lead0(a);
    assign c = $lead1(a);
endmodule //sub
```

After reading in the RTL and elaborating the design, you can check the datapath extension functions instantiated in the design using the following command:

```
rc:/> get_attr hdl_vdp_list /des*/top
round
```

Related Information

Affected by this command: [elaborate](#)

Related attribute: (subdesign) [hdl_vdp_list](#) on page 1034

language

`language string`

Read-only design attribute. Specifies whether the design is a Verilog module or VHDL entity.

Related Information

Related command: [read_hdl](#)

Related attribute: (subdesign) language on page 1035

library_name

`library_name string`

Read-only design attribute. Returns the name of the Verilog or VHDL library in which the module or entity definition is stored. This name corresponds to the name specified with the `-lib` option of the `read_hdl` command. If this option was not specified, the name defaults to default.

Related Information

Related command: [read_hdl](#)

Related attribute: (subdesign) library_name on page 1036

logic_abstract

`logic_abstract {false | true}`

Read-only design attribute. Specifies whether the design is inferred as a logic abstract from an empty module, an entity without an architecture, or an entity whose architecture is empty in the input design description.

- If the attribute value is `true`, the design is considered an empty module with port directions and bit widths, and is treated as an unresolved reference.
- If the attribute value is `false`, the design can be optimized by the generic optimization engine.

Note: This attribute is supported in the RTL and structural flows.

Related Information

- Affected by this command: [elaborate](#)
Related attribute: (subdesign) [logic_abstract](#) on page 1036

max_cap_cost

`max_cap_cost string`

Read-only [design](#) attribute. Returns the sum of the capacitance violations on all pins and ports in the design. A violation is flagged when the actual capacitance on a pin or port is larger than the `max_capacitance` constraint (attribute).

Related Information

- Related command: [report design rules](#)
Affected by these attributes: (design) [max_capacitance](#) on page 553
(libpin) [max_capacitance](#) on page 220
(port) [max_capacitance](#) on page 647
Related attributes: (design) [max_fanout_cost](#) on page 940
(design) [max_trans_cost](#) on page 940

max_fanout_cost

`max_fanout_cost string`

Read-only [design](#) attribute. Returns the sum of the fanout violations on all ports in the design. A violation is flagged when the actual fanout on a port is larger than the `max_fanout` constraint (attribute).

Related Information

- Related command: [report design rules](#)
Affected by these attributes: (design) [max_fanout](#) on page 554
(libpin) [max_fanout](#) on page 220
(port) [max_fanout](#) on page 648

Related attributes: (design) [max_cap_cost](#) on page 939
(design) [max_trans_cost](#) on page 940

max_trans_cost

`max_trans_cost string`

Read-only [design](#) attribute. Returns the sum of the transition time violations on all ports in the design. A violation is flagged when the actual transition time on a port is larger than the `max_transition` constraint (attribute).

Related Information

Related command: [report design rules](#)
Affected by these attributes: (design) [max_transition](#) on page 555
(libpin) [max_transition](#) on page 220
(port) [max_transition](#) on page 649
Related attributes: (design) [max_cap_cost](#) on page 939
(design) [max_fanout_cost](#) on page 940

min_cap_cost

`min_cap_cost {no_value | float}`

Read-only [design](#) attribute. Returns the sum of the capacitance violations on all pins and ports in the design. A violation is flagged when the actual capacitance on a pin or port is smaller than the `min_capacitance` constraint (attribute).

Related Information

Affected by this attribute: [min_capacitance](#) on page 221
Related attributes: (design) [min_fanout_cost](#) on page 941
(design) [min_trans_cost](#) on page 941

min_fanout_cost

`min_fanout_cost {no_value | float}`

Read-only design attribute. Returns the sum of the fanout violations on all ports in the design. A violation is flagged when the actual fanout on a port is smaller than the `min_fanout` constraint (attribute).

Related Information

Affected by this attribute: [min_fanout](#) on page 221

Related attributes: (design) [min_cap_cost](#) on page 941

(design) [min_trans_cost](#) on page 941

min_trans_cost

`min_trans_cost {no_value | float}`

Read-only design attribute. Returns the sum of the tranistion time violations on all ports in the design. A violation is flagged when the actual tranistion time on a port is smaller than the `min_transition` constraint (attribute).

Related Information

Affected by this attribute: [min_transition](#) on page 221

Related attributes: (design) [min_cap_cost](#) on page 941

(design) [min_fanout_cost](#) on page 941

net_area

`net_area string`

Read-only design attribute. Returns the net area of the design. The area is specified in terms of the units specified in the library. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report area](#)

Related attribute: (subdesign) [net_area](#) on page 1037

physical_cell_area

```
physical_cell_area {no_value | float}
```

Read-only design attribute. Computes the cell area of the design using the LEF area values. The area is specified in terms of the units specified in the library.

Related Information

Related command: [report area](#)

Related attribute: (subdesign) [physical_cell_area](#) on page 1037

pin_count

```
pin_count integer
```

Read-only design attribute. Returns the number of pins in the design.

Related Information

Related attributes: (gcell) [pin_count](#) on page 436

(instance) [pin_count](#) on page 952

(subdesign) [pin_count](#) on page 1037

seq_reason_deleted

```
seq_reason_deleted {{instance reason}...}
```

Read-write design attribute. Returns a Tcl list of Tcl lists. Each Tcl list contains the name of a sequential instance that was deleted during optimization and the reason why the instance was deleted.

If the `delete_unloaded_seqs` root attribute is set to `false`, the reason might be appended with an asterisk (*) to indicate that the sequential instance is optimized for the reason reported but not deleted. In this case, the flop output pin will be dangling, not driving any loads. The loads driven by the flop before optimization, will be replaced/driven by a constant, or by the output pins of the merged flops pin.

Examples

```
rc;/> get_attr seq_reason_deleted cr4dpu_swizzle_load
{{ctl_wr_reg[36] } {{merged with ctl_wr_reg[0]}}} {{ctl_wr_reg[46] } {{merged with
```

Attribute Reference for Encounter RTL Compiler

Analysis—Design Attributes

```
ctl_wr_reg[10]} } {{ctl_wr_reg[47] } {{merged with ctl_wr_reg[11]}}}  
.....  
{ {ctl_wr_reg[32] } unloaded} {{ctl_wr_reg[12] } unloaded} {{ctl_wr_reg[20] }  
unloaded} {{ctl_wr_reg[28] } unloaded}
```

If the `delete_unloaded_seqs` root attribute is set to `false`, the result may look like:

```
rc:/> get_attr seq_reason_deleted cr4dpu_swizzle_load  
ctl_wr_reg[ 36 ] } {{merged(*) with ctl_wr_reg[ 0 ]} ctl_wr_reg[ 46 ] }  
{{merged(*) with ctl_wr_reg[ 10 ]} ctl_wr_reg[ 47 ] } {{merged(*) with ctl_wr_reg[  
11 ]}}
```

Related Information

Related command: [report sequential -deleted_seqs](#)

Affected by this attribute: [delete_unloaded_seqs](#) on page 705

slack

```
slack {no_value | float}
```

Read-only [design](#) attribute. Returns the worst slack (over all paths) in the design in picoseconds. The resolution is 1. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report timing](#)

Related attributes: (instance) [slack](#) on page 953

(pin) [slack](#) on page 981

(port) [slack](#) on page 1018

(cost group) [slack](#) on page 1060

slack_by_mode

```
slack_by_mode {{mode_name_1 delay_value} [{mode_name_2 delay_value}]...}
```

Read-only design attribute. Returns a Tcl list of lists with slack values in picoseconds for each timing mode in a design.

Example

The following example shows that the `top` design has two modes. Mode `a` has a `-374.5` ps negative slack value and mode `b` has a `-314.5` ps negative slack value:

```
rc:/> get_attribute slack_by_mode/designs/top/
{/designs/top/modes/b -374.5} {/designs/top/modes/a -314.5}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands: [report clocks](#)

[report qor](#)

[report timing](#)

[write encounter](#)

[write sdc](#)

[write script](#)

Related commands: [create mode](#)

[read sdc](#)

Related attributes: (instance) [disabled arcs by mode](#) on page 561

(pin) [external delays by mode](#) on page 970

(port) [external delays by mode](#) on page 1008

(pin) [propagated clocks by mode](#) on page 977

(port) [propagated clocks by mode](#) on page 1016

(pin) [slack by mode](#) on page 981

(port) [slack by mode](#) on page 1019

(cost group) [slack by mode](#) on page 1060

(pin) [timing case computed value by mode](#) on page 986

(port) [timing_case_computed_value_by_mode](#) on page 1022
instance) [timing_case_disabled_arcs](#) on page 954
(instance) [timing_case_disabled_arcs_by_mode](#) on page 955
(pin) [timing_case_logic_value_by_mode](#) on page 609
(port) [timing_case_logic_value_by_mode](#) on page 660

tns

`tns string`

Read-only [design](#) attribute. Returns the sum (or total) of all worst negative slacks of all endpoints in the design in picoseconds. If the worst slack at an endpoint is positive, it is not considered. If all endpoints in the design have a positive worst slack, a value of 0 will be reported. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Note: An endpoint can belong to several cost groups and can have a different worst slack value in each cost group. For the calculation of this value, the worst slack of each endpoint is used.

Example

Assume a design has two endpoints, A and B, whose worst slack is -10 and +15 respectively. In this case, the `tns` value of the design returns 10, because the slack value of +15 is treated as zero slack.

Related Information

Related command: [report timing](#)

Related attribute: [\(cost_group\) tns](#) on page 1062

total_net_length

`total_net_length float`

Read-only design attribute. Computes the total net length of the design in microns. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related commands:

- [generate_reports](#)
- [report_qor](#)
- [summary_table](#)

wireload

`wireload string`

Read-only design attribute. Returns the current wire-load model for the design.

Related Information

Related attribute:

- (subdesign) [wireload](#) on page 1038

Instance Attributes

Contain information about a combinational, sequential, or hierarchical instance in the specified design. Instance objects are found in the `instances_comb`, `instances_seq`, and `instances_hier` directories.

- To set an instance attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -instance name]
```

- To get an instance attribute value, type

```
get_attribute attribute_name [find /des*/design -instance name]
```

buffer

`buffer {true | false}`

Read-only instance attribute. Indicates if the instance is a buffer.

Related Information

Related attributes: (libcell) buffer on page 190

cell_area

`cell_area string`

Read-only instance attribute. Returns the cell area of the instance, which is specified in library units. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: report area

Related attributes: (libcell) area on page 188

(design) cell_area on page 935

(subdesign) cell_area on page 1031

combinational

```
combinational {false | true}
```

Read-only instance attribute. Indicates if the instance is a combinational or tristate instance.

Related Information

Related attributes: (libcell) combinational on page 194

exceptions

```
exceptions string
```

Read-only instance attribute. Returns a list of all the timing exceptions that were applied to the specified instance.

Related Information

Affected by these commands:

multi_cycle
path_adjust
path_delay
path_disable
path_group

Related attributes:

(clock) exceptions on page 1055
(clock_domain) exceptions on page 1058
(cost_group) exceptions on page 1059
(external_delay) exceptions on page 1070
(pin) exceptions on page 969
(port) exceptions on page 1007

flop

```
flop {true | false}
```

Read-only instance attribute. Indicates if the instance is a flip-flop (or edge-triggered sequential element).

Related Information

Related attribute: (libcell) [flop](#) on page 196

hdl_instantiated

```
hdl_instantiated {true | false}
```

Read-only instance attribute. Indicates whether the instance is present in the input RTL description (true) or generated by RTL Compiler during synthesis (false).

Related Information

Set by these commands: [elaborate](#)
[read_netlist](#)

hierarchical

```
hierarchical {true | false}
```

Read-only instance attribute. Indicates if an instance is hierarchical.

inverted_phase

```
inverted_phase {true | false}
```

Read-only instance attribute. Indicates whether a transformation (as described for the lbr_seq_in_out_phase_opto attribute) did occur for this instance. Transformations can only occur when you enabled the lbr_seq_in_out_phase_opto root attribute.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the ls command by default.

Note: This attribute applies only to mapped sequential instances.

Related Information

Affected by this attribute: [lbr_seq_in_out_phase_opto](#) on page 786

inverter

`inverter {true | false}`

Read-only [instance](#) attribute. Indicates if the instance is an inverter.

Related Information

Related attribute: [\(libcell\) inverter](#) on page 197

latch

`latch {true | false}`

Read-only [instance](#) attribute. Indicates if the instance is a latch (or level-sensitive sequential element).

Related Information

Related attribute: [\(libcell\) latch](#) on page 199

libcell

`libcell string`

Read-write [instance](#) attribute. Specifies the library cell to which the instance is mapped.

Note: To replace one cell with another, the pin mappings must be identical.

Related Information

Set by this command: [synthesize -to_mapped](#)

loop_breaker_is_ignored_for_all_modes

```
loop_breaker_is_ignored_for_all_modes {false | true}
```

Read-only instance attribute. Indicates for loop breaker instances whether the path tracer traverses the loop breaker instance.

If set to `true`, the path tracer traverses the loop breaker and the tool reports the timing paths through the loop breaker. The loop breaker is still necessary for breaking the topological loop in the timing graph.

This attribute value is set to `true` if all following conditions are met:

- The design is in multi-mode (several modes have been created).
- For each of the modes, the loop through this loop breaker instance is broken differently by mode with SDC `set_disable_timing` constraints.

This attribute is set to `false` if the loop is not broken in all modes.

module

```
module string
```

Read-only instance attribute. Returns the name of the module (subdesign) that is instantiated.

negative_edge_clock

```
negative_edge_clock {false | true}
```

Read-only instance attribute. When `true`, indicates that the flop instance was inferred from an RTL statement triggered by a `negedge` (Verilog) or `falling_edge` (VHDL) condition.

Example

Assume your RTL description has the following:

```
assign nclk = ~clk;
always @(posedge nclk)
begin
    ou1 <= in;
end

always @(negedge clk)
begin
    ou2 <= in;
end
```

After elaborate, the attribute values are:

```
rc:/> get_attribute negative_edge_clock ou1_reg  
false  
rc:/> get_attribute negative_edge_clock ou2_reg  
true
```

pin_count

`pin_count integer`

Read-only instance attribute. Returns the number of logical pins of the instance.

Related Information

Related attributes:	(design) pin_count on page 942
	(gcell) pin_count on page 436
	(subdesign) pin_count on page 1037

primitive_function

`primitive_function string`

Read-only instance attribute. Returns the primitive function if the instance is a primitive. Possible values are and, nand, or, nor, buf, not, d_flop, latch, mux, bmx, xor, xnor, bufif0, bufif1, notif0, notif1, and dc.

Related Information

Related command: [edit_netlist new_primitive](#)

retime_original_registers

`retime_original_registers register_name`

Read-only instance attribute. Returns the original name of the specified retimed register. Registers must first be marked with the `trace_retime` attribute before retiming optimization.

Example

The following example marks all the registers in the design and then retrieves the retimed instances with the `retime_original_registers` command:

```
set_attribute trace_retime true [find / -instance *seq/*]  
...  
retime -min_delay  
rc:/> get_att retime_original_registers  
/designs/TOP/instances_hier/U1/instances_hier/mul_9/instances_seq/retime_94_reg  
U1/mul_9_14/retime_94_reg
```

Related Information

Related command: [retime](#)

Affected by this attribute: [trace_retime](#) on page 853

Related attributes: [dont_retime](#) on page 838

(design) [retime](#) on page 835

(subdesign) [retime](#) on page 906

[retime_hard_region](#) on page 906

[retime_reg_naming_suffix](#) on page 815

sequential

```
sequential {true | false}
```

Read-only [instance](#) attribute. Indicates if an instance is a sequential element (latch or flip-flop)

Related Information

Related attribute: (libcell) [sequential](#) on page 207

slack

```
slack {no_value | float}
```

Read-only [instance](#) attribute. Returns the slack of the instance in picoseconds. The resolution is 1. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command:	report timing
Related attributes:	(design) slack on page 944
	(pin) slack on page 981
	(port) slack on page 1018
	(cost_group) slack on page 1060

subdesign

`subdesign string`

Read-only [instance](#) attribute. Identifies the subdesign that the instance instantiates.

timing_case_disabled_arcs

`timing_case_disabled_arcs {libarc_a libarc_b...}`

Read-only [instance](#) attribute. Identifies the timing arcs that are disabled due to case analysis. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Affects these commands:	report clocks report qor read sdc report timing write sdc write script
Affects this attribute:	timing_case_disabled_arcs_by_mode on page 955
Related attributes:	(pin/port) timing_case_logic_value on page 608

timing_case_disabled_arcs_by_mode

```
timing_case_disabled_arcs_by_mode {mode_name_1 {libarc_a libarc_b...}}  
[mode_name_2 {libarc_c libarc_d...}]....}
```

Read-only [instance](#) attribute. Returns a Tcl list of lists with timing arcs of library cells for an instance that is disabled due to case analysis for each timing mode.

Example

The `timing_case_disabled_arcs_by_mode` attribute returns the timing arcs for disabled instances when the SDC `set_case_analysis` constraint is set on the input pin of an instance. The following example returns the timing arcs where the SDC constraint was applied on the A pin of the `nand_inst` instance for mode a:

```
rc:/>get_attribute timing_case_disabled_arcs_by_mode nand_inst  
{/designs/top/modes/a {/libraries/typical/libcells/NAND2X1/Y/inarcs/B_Y_n60  
/libraries/typical/libcells/NAND2X1/Yinarcs/A_Y_n60}}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:

[report_clocks](#)
[report_qor](#)
[read_sdc](#)
[report_timing](#)
[write_sdc](#)
[write_script](#)

Related commands:

[create_mode](#)
[read_sdc](#)

Related attributes:

[disabled_arcs_by_mode](#) on page 561
(pin) [external_delays_by_mode](#) on page 970
(port) [external_delays_by_mode](#) on page 1008
(design) [latch_borrow_by_mode](#) on page 548
(instance) [latch_borrow_by_mode](#) on page 565
(design) [latch_max_borrow_by_mode](#) on page 551

(instance) [latch_max_borrow_by_mode](#) on page 568
(pin) [latch_max_borrow_by_mode](#) on page 598
(pin) [propagated_clocks_by_mode](#) on page 977
(port) [propagated_clocks_by_mode](#) on page 1016
(design) [slack_by_mode](#) on page 944
(pin) [slack_by_mode](#) on page 981
(port) [slack_by_mode](#) on page 1019
(cost group) [slack_by_mode](#) on page 1060
(pin) [timing_case_computed_value_by_mode](#) on page 986
(port) [timing_case_computed_value_by_mode](#) on page 1022
instance) [timing_case_disabled_arcs](#) on page 954
(pin) [timing_case_logic_value_by_mode](#) on page 609
(port) [timing_case_logic_value_by_mode](#) on page 660

timing_model

`timing_model {true | false}`

Read-only [instance](#) attribute. Indicates if the instance has only a timing model. A timing model is a library cell for which timing data is available, but no functional information.

Related Information

Related attribute: [\(libcell\) timing_model](#) on page 208

tristate

`tristate {true | false}`

Read-only [instance](#) attribute. Indicates if the instance has a tristate output pin.

Related Information

Related attribute: [\(libcell\) tristate](#) on page 209

unique_versions

`unique_versions string`

Read-only `instance` attribute. Returns a list of the unique uses of an instance from a collection of instances. An object can have multiple object versions in the design. Use this attribute to get all the objects it represents.

Example

The following command returns a list of the unique uses of an instance:

```
get_attribute unique_versions [find /des/* -instance inst_name]
```

Related Information

Related command: [report timing](#)

Related attributes: (constant) [unique_versions](#) on page 959

(pin) [unique_versions](#) on page 992

(net) [unique_versions](#) on page 1002

(port) [unique_versions](#) on page 1026

(subport) [unique_versions](#) on page 1042

Constant Attributes

Contain information about the specified constant. Each level of hierarchy has its own dedicated logic constants that can only be connected to other objects within that level of hierarchy, such as `logic0` and `logic1` pins. They are in the `constants` directory and are called 1 and 0.

- To get a constant attribute value, type

```
get_attribute attribute_name [find  
/des*/design_name/instances_hier/instance_name/constants -constant 0]
```

or

```
get_attribute attribute_name [find  
/des*/design_name/instances_hier/instance_name/constants -constant 1]
```

pin_capacitance

`pin_capacitance float float`

Read-only constant attribute. Returns the rise and fall pin capacitances of a constant's net in femtofarads.

Example

The following example returns the pin capacitance of the constant's net:

```
rc:/> get_attribute pin_capacitance /designs/top/instances_hier/inst1/constants/0  
0.0 0.0
```

Related Information

Related attributes:	(pin) pin_capacitance on page 975
	(port) pin_capacitance on page 1013
	(subport) pin_capacitance on page 1041

unique_versions

`unique_versions string`

Read-only constant attribute. Returns a list of the unique uses of a constant from a collection of constants. An object can have multiple object versions in the design. Use this attribute to get all the objects it represents.

Example

The following command returns a list of the unique uses of the constant:

```
rc:/> get_attribute unique_versions /designs/top/instances_hier/inst1/constants/0  
/designs/top/instances_hier/inst2/constants/0  
/designs/top/instances_hier/inst1/constants/0
```

Related Information

Related attributes:	(instance) unique_versions on page 957
	(pin) unique_versions on page 992
	(net) unique_versions on page 1002
	(port) unique_versions on page 1026
	(subport) unique_versions on page 1042

wire_capacitance

`wire_capacitance float`

Read-only constant attribute. Returns the wire-capacitance of a constant's net.

Example

The following command returns the wire capacitance of the constant's net:

```
get_attribute wire_capacitance /designs/top/instances_hier/inst1/constants/0  
0.0
```

Related Information

Related attributes:	(pin) wire_capacitance on page 992
	(port) wire_capacitance on page 1027
	(subport) wire_capacitance on page 1042

wire_length

wire_length {no_value | *float*}

Read-only constant attribute. Returns the wire length of the net connected to the given constant.

Note: The value for constant objects is zero.

Related Information

Related attributes: (pin) [wire_length](#) on page 993
(port) [wire_length](#) on page 1027
(subport) [wire_length](#) on page 1043

wire_resistance

wire_resistance {no_value | *float*}

Read-only constant attribute. Returns the net resistance of a constant's net in kilohms. Resolution is 1/1000. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes: (pin) [wire_resistance](#) on page 993
(port) [wire_resistance](#) on page 1028
(subport) [wire_resistance](#) on page 1043

Pin Attributes

Contain information about the specified instance pin. A pin is a single bit connection point on an instance. Pin objects are stored in `pins_in` and `pins_out` directories.

- To set a pin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pin instance/pin]
```

- To get a pin attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pin instance/pin]
```

Note: You may need to specify more than one instance to serve as the search path that will uniquely identify the relevant pin.

Note: These attributes are located at `/designs/design/instances_*/pins_in` and `/designs/design/instances_*/pins_out`

boundary_change

```
boundary_change { constant 0 propagated | constant 1 propagated
| gobbled | inverted
| pushed equal signal through pin XYZ
| pushed opposite signal through pin XYZ
| routed equal signal through pin XYZ around instance ABC
| routed opposite signal through pin XYZ around instance ABC
none}
```

Read-only `pin` attribute. Returns the type of boundary optimization that was performed on the specified hierarchical pin. The types of boundary optimization are:

- `constant 0 propagated`—Indicates that a constant 0 was propagated through the specified pin.
- `constant 1 propagated`—Indicates that a constant 1 was propagated through the specified pin.
- `gobbled`—The driving logic of the specified pin has been removed because it does not load anything.
- `inverted`—Indicates that the hierarchical instance pin was inverted during boundary optimization.
- `pushed equal signal through pin`—Specifies that the specified pin is equal to the returned pin and all connections from the specified pin are now connected through the returned pin.

Attribute Reference for Encounter RTL Compiler

Analysis—Pin Attributes

- pushed opposite signal through pin—Specifies that the specified pin is the opposite of the returned pin and all connections from the specified pin are now connected through the returned pin through inverter.
- routed equal signal through pin around instance—Indicates that the feedthrough from the returned pin to the specified pin is being routed around the returned instance.
- routed opposite signal through pin around instance—Indicates that the inverted feedthrough from the returned pin to the specified pin is being routed around the returned instance.
- none—No boundary change was performed on the specified pin.

Example

Take this input RTL:

```
module top(a,b,out,out1);
    input a,b;
    output out,out1;
    child inst(a,b,b,out,out1);
endmodule

module child(in1,in2,in3,out,out1);
    input in1,in2,in3;
    output out,out1;

    and u2(out,n_1,in1);
    assign out1 = ~in1;
endmodule
```

After the `synthesize -to_map` command, the original RTL would look like the following RTL:

```
module child(in1, in2, in3, out, out1);
    input in1, in2, in3;
    output out, out1;
    wire in1, in2, in3;
    wire out, out1;
    wire n_0;
    assign out1 = 1'b0;
    invl g6(.A (n_0), .Y (out));
    nand2 g7(.A (in1), .B (in3), .Y (n_0));
endmodule

module top(a, b, out, out1);
    input a, b;
    output out, out1;
    wire a, b;
    wire out, out1;
    wire UNCONNECTED;
    child inst(a, 1'b0, b, out, UNCONNECTED);
    invl g2(.A (a), .Y (out1));
endmodule
```

Here are the results for various pins:

```
rc:/> get_attribute boundary_change /designs/top/instances_hier/inst/pins_in/in1
none
rc:/> get_attribute boundary_change /designs/top/instances_hier/inst/pins_in/in2
pushed equal signal through 'in3'
rc:/> get_attribute boundary_change \
/designs/top/instances_hier/inst/pins_out/out1
routed opposite signal through 'in1' around 'inst'
```

Related Information

[Setting Boundary Optimization in Using Encounter RTL Compiler](#)

Affected by this attribute: [boundary_opto](#) on page 888

capturer

```
capturer {true | false}
```

Read-only [pin](#) attribute. Indicates if the pin's required time was derived through static analysis from a setup check (`false`) or user-defined output delay constraint (`true`). This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report_timing](#)

Related attribute [\(port\) capturer](#) on page 1003

causes_ideal_net

```
causes_ideal_net {true | false}
```

Read-only [pin](#) attribute. Indicates whether the specified pin causes its net to become an ideal net.

RTL Compiler implicitly sets an ideal net if some pin is

- A pin of a sequential element that is an asynchronous set or reset. You can check for this attribute by using the `ls -long -attribute` command on the pin or libcell.
- A pin of a sequential element that is a clock pin. You can check for this attribute by using the `ls -long -attribute` command on the pin or libcell.

- Any input pin of a sequential element that has no setup arc. That is, any sequential element that may be asynchronous.

Example

The following example queries whether the `in1[1]` input pin causes its net to become ideal:

```
get_attribute causes_ideal_net \
  {/designs/trance/instances_hier/inst1/pins_in/in1[1]}
false
```

Related Information

Related attributes:	(port) causes ideal net on page 1004
	(subport) causes ideal net on page 1039
	ideal seq async pins on page 544
	time recovery arcs on page 535

clock_sense_negative

`clock_sense_negative clock...`

Read-only pin attribute. Returns the clocks that are propagated with a negative clock sense at this pin.

Example

Assume that the SDC file you read in contains the following command:

```
set_clock_sense -negative [get_pins {x1/Y}] -clock [get_clocks {CLK1}]
```

To verify which clocks are propagated with negative sense on pin `x1/Y`, specify the following command:

```
rc:/> get_attr clock_sense_negative [find / -pin x1/Y]
/designs/top/instances_comb/x1/pins_out/Y
```

Related Information

Set by this command:	dc::set_clock_sense -negative
Related attributes:	(clock) clock sense negative on page 1051
	(clock) clock sense positive on page 1052
	(pin) clock sense positive on page 965

(clock) [clock sense stop propagation](#) on page 1052

(pin) [clock sense stop propagation](#) on page 965

(pin) [propagated clocks](#) on page 975

clock_sense_positive

`clock_sense_positive clock...`

Read-only pin attribute. Returns the clocks that are propagated with a positive clock sense at this pin.

Example

Assume that the SDC file you read in contains the following command:

```
set_clock_sense -positive [get_pins {x1/Y}] -clock [get_clocks {CLK1}]
```

To verify which clocks are propagated with positive sense on pin `x1/Y`, specify the following command:

```
rc:/> get_attr clock_sense_positive [find / -pin x1/Y]  
/designs/top/instances_comb/x1/pins_out/Y
```

Related Information

Set by this command:

[dc::set_clock_sense -positive](#)

Related attributes:

(clock) [clock sense negative](#) on page 1051

(pin) [clock sense negative](#) on page 964

(clock) [clock sense positive](#) on page 1052

(clock) [clock sense stop propagation](#) on page 1052

(pin) [clock sense stop propagation](#) on page 965

(pin) [propagated clocks](#) on page 975

clock_sense_stop_propagation

`clock_sense_stop_propagation clock...`

Read-only pin attribute. Returns the clocks that stop propagating at this pin.

Example

Assume that the SDC file you read in contains the following command:

```
set_clock_sense -stop_propagation [get_pins {x1/Y}] -clock [get_clocks {CLK1}]
```

To verify which clocks stop propagating on pin x1/Y, specify the following command:

```
rc:/> get_attr clock_sense_stop_propagation [find / -pin x1/Y]  
/designs/top/timing/clock_domains/domain_1/CLK1
```

Related Information

Set by this command:

[dc::set_clock_sense -stop_propagation](#)

Related attributes:

(clock) [clock sense negative](#) on page 1051

(pin) [clock sense negative](#) on page 964

(clock) [clock sense positive](#) on page 1052

(pin) [clock sense positive](#) on page 965

(clock) [clock sense stop propagation](#) on page 1052

(pin) [propagated_clocks](#) on page 975

clock_sources_inverted

`clock_sources_inverted string`

Read-only [pin](#) attribute. Returns a Tcl list of clock objects that were applied in the inverted sense to the specified pin.

Related Information

Related attributes

[inverted_sources](#) on page 673

(port) [clock_sources_inverted](#) on page 1005

(pin) [clock_sources_non_inverted](#) on page 967

(port) [clock_sources_non_inverted](#) on page 1005

(pin) [propagated_clocks](#) on page 975

(port) [propagated_clocks](#) on page 1014

(pin) [propagated_clocks_by_mode](#) on page 977

(port) [propagated_clocks_by_mode](#) on page 1016

clock_sources_non_inverted

`clock_sources_non_inverted string`

Read-only [pin](#) attribute. Returns a Tcl list of clock objects that were applied in the non-inverted sense to the specified pin or port.

Related Information

Related attributes	non_inverted_sources on page 673 (pin) clock_sources_inverted on page 966 (port) clock_sources_inverted on page 1005 (port) clock_sources_non_inverted on page 1005 (pin) propagated_clocks on page 975 (port) propagated_clocks on page 1014 (pin) propagated_clocks_by_mode on page 977 (port) propagated_clocks_by_mode on page 1016
--------------------	--

connect_delay

`connect_delay { {no_value no_value} | {float float}}`

Read-only [pin](#) attribute. Returns the rise and fall connect delay to this pin in picoseconds. Resolution is 1. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command:	report_timing
Related attribute:	(port) connect_delay on page 1006

direction

```
direction {in | out | inout}
```

Read-only pin attribute. Returns the direction of the pin.

Related Information

Related attributes:	(port) direction on page 1006
	(port bus) direction on page 1029
	(subport) direction on page 1040
	(subport bus) direction on page 1044

drivers

```
drivers {constant | pin | port | subport}
```

Read-only pin attribute. Returns the drivers for the pin.

Related Information

Related attributes:	(net) drivers on page 1000
	(pgpin) drivers on page 994
	(port) drivers on page 1006
	(subport) drivers on page 1040

endpoint

```
endpoint {true | false}
```

Read-only pin attribute. Indicates if the pin is the endpoint of a timing path. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command:	report timing
Related attribute:	(port) endpoint on page 1007

exceptions

`exceptions string`

Read-only `pin` attribute. Returns a list of all the timing exceptions that were applied to the specified pin.

Related Information

Affected by these commands:

- [path_adjust](#)
- [path_delay](#)
- [path_disable](#)
- [path_group](#)
- [multi_cycle](#)

Related attributes:

- (clock) [exceptions](#) on page 1055
- (clock_domain) [exceptions](#) on page 1058
- (cost_group) [exceptions](#) on page 1059
- (external_delay) [exceptions](#) on page 1070
- (instance) [exceptions](#) on page 948
- (port) [exceptions](#) on page 1007

external_delays

`external_delays string`

Read-only `pin` attribute. Returns a list of the external delay objects associated with this pin.

Related information

Set by this command:

- [external_delay](#)

Related Attribute:

- [external_delays](#) on page 1008

external_delays_by_mode

```
external_delays_by_mode {mode external_delay} [{mode external_delay}]...
```

Read-only pin attribute. Returns a Tcl list of external delays on a pin for each mode.

Examples

- The following example shows that an input delay was annotated on the `inv_inst/Y` pin in mode a but not in mode b:

```
rc:/>get_attribute external_delays_by_mode inv_inst/Y  
{/designs/top/modes/b {} }{/designs/top/modes/a  
/designs/top/modes/a/external_delays/in_del}
```

- You can then use the `ls -al` command on the external delay for mode a to see the associated attributes, as shown in the following example:

```
rc:/>ls -al /designs/top/modes/a/external_delays/in_del  
/designs/top/modes/a/external_delays/in_del (external_delay)  
  All attributes:  
    clock = /designs/top/modes/a/clock_domains/domain_1/clock  
    clock_network_latency_included = false  
    clock_rise = true  
    clock_source_latency_included = false  
    delay = 50.0 50.0 50.0 50.0  
    exceptions =  
    external_delay_pins = /designs/top/instances_comb/inv_inst/pins_out/Y  
    input_delay = true  
    level_sensitive = false
```



Important

The timing path will get snipped (segmented) if the external delay is set in the middle of a timing path. In other words, the pin, which has the applied external delay constraints, will become a timing endpoint or startpoint once the external delay is annotated on it.

Note: When a path is segmented, unless an `input_delay` is applied at the segmentation endpoint, the rest of the path will be unconstrained.

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands:

[report clocks](#)

[report qor](#)

[report timing](#)

Attribute Reference for Encounter RTL Compiler

Analysis—Pin Attributes

[write encounter](#)

[write sdc](#)

[write script](#)

Related commands: [create mode](#)

[read sdc](#)

Related attributes:

- [disabled arcs by mode](#) on page 561
- (port) [external delays by mode](#) on page 1008
- (design) [latch borrow by mode](#) on page 548
- (instance) [latch borrow by mode](#) on page 565
- (design) [latch max borrow by mode](#) on page 551
- (instance) [latch max borrow by mode](#) on page 568
- (pin) [latch max borrow by mode](#) on page 598
- (pin) [propagated clocks by mode](#) on page 977
- (port) [propagated clocks by mode](#) on page 1016
- (design) [slack by mode](#) on page 944
- (pin) [slack by mode](#) on page 981
- (port) [slack by mode](#) on page 1019
- (cost group) [slack by mode](#) on page 1060
- (pin) [timing case computed value by mode](#) on page 986
- (port) [timing case computed value by mode](#) on page 1022
- instance) [timing case disabled arcs by mode](#) on page 955
- (pin) [timing case logic value by mode](#) on page 609
- (port) [timing case logic value by mode](#) on page 660

has_min_delay

```
has_min_delay {true | false}
```

Read-only pin attribute. Indicates whether the minimum delays were computed for the pin to honor the data-to-data checks timing constraints. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the ls command by default.

Example

The following command checks whether minimum delays were computed for pin SE of instances out_reg[1]. The returned value indicates that the minimum delays were computed.

```
rc:/> get_attribute has_min_delay [find /des*/top -pin out_reg[1]/pins_in/SE]  
true
```

Related Information

Related command: [report timing](#)

Related attributes: (port) [has_min_delay](#) on page 1010

(pin) [min_slew](#) on page 973

(port) [min_slew](#) on page 1011

(pin) [min_timing_arcs](#) on page 974

launcher

```
launcher {true | false}
```

Read-only pin attribute. Indicates if the arrival time at this pin was derived through static analysis from a clock signal arriving at an edge-triggered or level-sensitive point, or from a user-defined input delay. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the ls command by default.

Related Information

Related command: [report timing](#)

Related attribute: (port) [launcher](#) on page 1010

libpin

`libpin string`

Read-only pin attribute. Returns the library pin associated with a pin of a mapped instance.

loads

`loads { pin | port | subport }`

Read-only pin attribute. Returns the hierarchical path to the pins or ports loading this pin.

Related Information

Related attributes:	(net) loads on page 1001
	(pgpin) loads on page 995
	(port) drivers on page 1006
	(subport) drivers on page 1040

min_slew

`min_slew Tcl_list`

Read-only pin attribute. Returns a Tcl list containing the computed rise and fall minimum slew values, respectively, in picoseconds. This attribute is only computed if the minimum delay must be computed for the pin to honor the data-to-data timing constraints. You can use the `has_min_delay` pin attribute to check if the `min_slew` attribute has been computed.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

The following command retrieves the minimum slew values on pin D of instance `out_reg[1]`.

```
rc:/> get_att min_slew [find /des*/top -pin out_reg[1]/pins_in/D]
834.8 662.0
```

Related Information

Related command: [report timing](#)

Related attributes:

- (pin) [has_min_delay](#) on page 972
- (port) [has_min_delay](#) on page 1010
- (port) [min_slew](#) on page 1011
- (clock) [slew](#) on page 674
- (port) [slew](#) on page 1020

min_timing_arcs

`min_timing_arcs string`

Read-only [pin](#) attribute. Returns the minimum delay timing arcs to this pin. This attribute is only computed if the pin requires to have the minimum delay computed to honor the data-to-data timing constraints. You can use the [has_min_delay](#) pin attribute to check if the [min_timing_arcs](#) attribute has been computed.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default

Related Information

Related attributes:

- (pin) [has_min_delay](#) on page 972
- (pin) [timing_arcs](#) on page 984

net

`net string`

Read-only [pin](#) attribute. Returns the net connected to the pin.

Related Information

Related attributes:

- (pgpin) [net](#) on page 995
- (port) [net](#) on page 1012
- (subport) [net](#) on page 1041

pin_capacitance

```
pin_capacitance {no_value | float float}
```

Read-only pin attribute. Returns the rise and fall capacitances for the pin in femtofarads. Resolution is 1/10. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes:	(constant) pin_capacitance on page 958
	(port) pin_capacitance on page 1013
	(subport) pin_capacitance on page 1041

propagated_clocks

```
propagated_clocks pin_name
```

Read-only pin attribute. Returns a Tcl list of clock information that has propagated to the specified pin. Each element of the list contains information about a single clock that was propagated.

The clock information contained in the Tcl list can be easily converted into an associative array using the Tcl command `array get`. This provides a convenient method to query for information about propagated clocks in a design. The keys of the associative array are:

- `clock` — The clock object that has been propagated.
- `phase` — A string value indicating whether the clock has been inverted ("+" or "-").
- `clock_source_late_latency` — A Tcl list containing any source latency values that have been picked up from the `clock_source_late_latency` attribute on pins or ports in the clock network. This does not include latency values from the clock object itself.
- `clock_network_late_latency` — A Tcl list containing any network latency values that have been picked up from the `clock_network_late_latency` attribute on pins or ports in the clock network. This does not include latency values from the clock object itself.
- `clock_setup_uncertainty` — A Tcl list containing any uncertainty values that have been picked up from the `clock_setup_uncertainty` attribute on pins or ports in the clock network. This does not include uncertainty values from the clock object itself.

Example

The following Tcl code shows an example usage of this attribute:

```
proc propagated_clock_info {pin} {
    foreach clock_info [get_attribute propagated_clocks $pin] {
        array set info_array $clock_info
        puts "clock : [basename $info_array(clock)]"
        puts "phase : $info_array(phase)"
        puts "source_latency : $info_array(clock_source_late_latency)"
        puts "network_latency : $info_array(clock_network_late_latency)"
        puts "uncertainty : $info_array(clock_setup_uncertainty)"
        puts ""
    }
}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands:

[report clocks](#)

[report qor](#)

[report timing](#)

[write encounter](#)

[write sdc](#)

[write script](#)

Related commands:

[read sdc](#)

Related attributes:

[disabled_arcs](#) on page 560

(pin) [external_delays](#) on page 969

(port) [external_delays](#) on page 1008

(design) [latch_borrow](#) on page 547

(instance) [latch_borrow](#) on page 564

(design) [latch_max_borrow](#) on page 550

(instance) [latch_max_borrow](#) on page 567

(pin) [propagated_clocks](#) on page 975

(port) [propagated_clocks](#) on page 1014

(design) [slack](#) on page 944

(pin) [slack](#) on page 981

(port) [slack](#) on page 1018
(cost group) [slack](#) on page 1060
(pin) [timing_case_computed_value](#) on page 984
(port) [timing_case_computed_value](#) on page 1021
instance) [timing_case_disabled_arcs](#) on page 954
(pin) [timing_case_logic_value](#) on page 608
(port) [timing_case_logic_value](#) on page 659

propagated_clocks_by_mode

```
propagated_clocks_by_mode { {mode_name_1 [clock_info_1]...}  
    [{mode_name_2 [clock_info_2]...}]}...}
```

Read-only [pin](#) attribute. Returns a Tcl list of lists consisting of the mode, related clocks, and clock characteristics, such as latency and uncertainty.

The clock information contained in the Tcl list can be easily converted into an associative array using the Tcl command `array get`. This provides a convenient method to query for information about propagated clocks in a design. The keys of the associative array are:

- `clock` — The clock object that has been propagated.
- `phase` — A string value indicating whether the clock has been inverted ("+" or "-").
- `clock_source_late_latency` — A Tcl list containing any source latency values that have been picked up from the `clock_source_late_latency` attribute on pins or ports in the clock network. This does not include latency values from the clock object itself.
- `clock_network_late_latency` — A Tcl list containing any network latency values that have been picked up from the `clock_network_late_latency` attribute on pins or ports in the clock network. This does not include latency values from the clock object itself.
- `clock_setup_uncertainty` — A Tcl list containing any uncertainty values that have been picked up from the `clock_setup_uncertainty` attribute on pins or ports in the clock network. This does not include uncertainty values from the clock object itself.

Example

The following command returns the propagated clock information for the `out_reg[2]/CLK` pin:

```
rc:/> get_attribute propagated_clocks_by_mode out_reg[2]/CLK
{/designs/add/modes/b {{clock
/designs/add/modes/b/clock_domains/domain_1/ck phase +
clock_source_late_latency {no_value no_value no_value no_value}
clock_network_late_latency {no_value no_value no_value no_value}
clock_setup_uncertainty {no_value no_value}}} }{/designs/add/modes/a
{{clock /designs/add/modes/a/clock_domains/domain_1/ck phase -
clock_source_late_latency {no_value no_value no_value no_value}
clock_network_late_latency {no_value no_value no_value no_value}
clock_setup_uncertainty {no_value no_value}}}}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands:	report clocks report qor report timing write encounter write sdc write script
Related commands:	create mode read sdc
Related attributes:	disabled arcs by mode on page 561 (pin) external delays by mode on page 970 (port) external delays by mode on page 1008 (design) latch borrow by mode on page 548 (instance) latch borrow by mode on page 565 (design) latch max borrow by mode on page 551 (instance) latch max borrow by mode on page 568

(port) [propagated_clocks_by_mode](#) on page 1016
(design) [slack_by_mode](#) on page 944
(pin) [slack_by_mode](#) on page 981
(port) [slack_by_mode](#) on page 1019
(cost group) [slack_by_mode](#) on page 1060
(pin) [timing_case_computed_value_by_mode](#) on page 986
(port) [timing_case_computed_value_by_mode](#) on page 1022
(instance) [timing_case_disabled_arcs_by_mode](#) on page 955
(pin) [timing_case_logic_value_by_mode](#) on page 609
(port) [timing_case_logic_value_by_mode](#) on page 660

propagated_ideal_network

```
propagated_ideal_network {true | false }
```

Read-only [pin](#) attribute. Indicates whether the specified pin is ideal. Specifically, the attribute checks whether the specified pin is in the fanout of another pin with the [ideal_network](#) attribute set to `true`.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Propagation Rules

The propagation of the [ideal_network](#) attribute follows these rules:

- A pin is treated as ideal if it is either a:
 - Pin specified in the object list of the [ideal_network](#) attribute.
 - Driver pin and its cell is ideal.
 - Load pin attached to an ideal net.
- A net is treated as ideal if all its driving cells are ideal.
- A combinational cell is treated as ideal if all its input pins are ideal.

Note: A hierarchical pin can propagate the [ideal_network](#) attribute.

Attribute Reference for Encounter RTL Compiler

Analysis—Pin Attributes

Propagation stops at the pins where these conditions are not met. These pins are referred to as network boundary pins, and they are ideal pins.

Example

The following example indicates that the `foo` pin is not ideal:

```
rc:/> get_attribute propagated_ideal_network \
{/designs/example/instances_hier/inst2/pins_in/foo}
false
```

Related Information

Affected by these attributes: (pin) [ideal_network](#) on page 596
 (port) [ideal_network](#) on page 645

rf_slack

```
rf_slack rise fall
```

Read-only [pin](#) attribute. Returns the slack for the rising and falling edges in picoseconds. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

```
rc:/> get_att rf_slack [find /des*/top -pin out_reg[1]/pins_in/D]
1384.3 900.5
```

Related Information

Related attributes: ((port) [rf_slack](#) on page 1018
 (design) [slack](#) on page 944
 (instance) [slack](#) on page 953
 (pin) [slack](#) on page 981
 (port) [slack](#) on page 1018
 (cost group) [slack](#) on page 1060

slack

```
slack {no_value | float}
```

Read-only [pin](#) attribute. Returns the slack of the pin in picoseconds. The resolution is 1. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

```
rc:/> get_att slack [find /des*/top -pin out_reg[1]/pins_in/D]
900.5
```

Related Information

Related command: [report timing](#)

Related attributes:

- (pin) [rf slack](#) on page 980
- ((port) [rf slack](#) on page 1018)
- (design) [slack](#) on page 944
- (instance) [slack](#) on page 953
- (port) [slack](#) on page 1018
- (cost group) [slack](#) on page 1060

slack_by_mode

```
slack_by_mode {{mode_name_1 delay_value} [{mode_name_2 delay_value}]...}
```

Read-only [pin](#) attribute. Returns a Tcl list of lists with slack values in picoseconds for each timing mode for a pin.

Returns the worst negative slack for a pin even if the pin is not a timing start point or end point.

Examples

- The following example returns the worst negative slack for the Y pin for modes b and a:

```
rc:/>get_attribute slack_by_mode inv_inst/Y
{/designs/top/modes/b -166.2} {/designs/top/modes/a -201.1}
```

Attribute Reference for Encounter RTL Compiler

Analysis—Pin Attributes

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands:

[report clocks](#)

[report qor](#)

[report timing](#)

[write encounter](#)

[write sdc](#)

[write script](#)

Related commands:

[create mode](#)

[read sdc](#)

Related attributes:

[disabled arcs by mode](#) on page 561

(pin) [external delays by mode](#) on page 970

(port) [external delays by mode](#) on page 1008

(design) [latch borrow by mode](#) on page 548

(instance) [latch borrow by mode](#) on page 565

(design) [latch max borrow by mode](#) on page 551

(instance) [latch max borrow by mode](#) on page 568

(pin) [propagated clocks by mode](#) on page 977

(port) [propagated clocks by mode](#) on page 1016

(design) [slack by mode](#) on page 944

(port) [slack by mode](#) on page 1019

(cost group) [slack by mode](#) on page 1060

(pin) [timing case computed value by mode](#) on page 986

(port) [timing case computed value by mode](#) on page 1022

(instance) [timing case disabled arcs by mode](#) on page 955

(pin) [timing case logic value by mode](#) on page 609

(port) [timing case logic value by mode](#) on page 660

slew

```
slew {rise fall}
```

Read-only pin attribute. Returns the computed rise and fall slew values, respectively, in picoseconds. The values are returned as a Tcl list.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

```
rc:/> get_att slew [find /des*/top -pin out_reg[1]/pins_in/D]
834.8 662.0
```

Related Information

Related command: [report timing](#)

Related attributes: (clock) [slew](#) on page 674

(port) [slew](#) on page 1020

slew_by_mode

```
slew_by_mode {{mode_name_1 rise fall} [{mode_name_2 rise fall}]...}
```

Read-only pin attribute. Returns a Tcl list of lists. Each list contains the mode name followed by the computed rise and fall slew values for the pin for that mode, in picoseconds.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

```
rc:/> get_att slew_by_mode [find /des*/top -pin out_reg[1]/pins_in/D]
{/designs/top/modes/b 834.8 662.0}
```

Related Information

Related command: [report timing](#)

Related attributes: (port) [slew_by_mode](#) on page 1020

startpoint

```
startpoint {true | false}
```

Read-only pin attribute. Indicates if the pin is the startpoint of a timing path. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report timing](#)

Related attributes: (port) [startpoint](#) on page 1021

timing_arcs

```
timing_arcs string
```

Read-only pin attribute. Returns the timing arcs to this pin. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

```
rc:/designs/alu/instances_seq/aluout_reg_1/pins_in> get_attribute timing_arcs CD \
{CP {} removal {}} {CP {} recovery {}}
```

timing_case_computed_value

```
timing_case_computed_value {0 | 1 | no_value}
```

Read-only pin attribute. Indicates if the value of this pin was computed to have a constant logic value. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Examples

- If one input of an AND gate is set to logic value 0, the output is computed to have a logic value of 0. You can use this attribute to create a case analysis report.
- In the following example, the `timing_case_computed_value` indicates a computed logic value of 0 when the SDC `set_case_analysis` command was applied on the `A` pin of the `nand_inst` instance:

```
rc:/>get_attribute timing_case_computed_value nand_inst/A
0
```

Attribute Reference for Encounter RTL Compiler

Analysis—Pin Attributes

Related Information

[Combinational Feedback Loops in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands:	<u>report clocks</u> <u>report qor</u> <u>report timing</u> <u>write_encounter</u> <u>write_sdc</u> <u>write_script</u>
Related commands:	<u>read_sdc</u>
Related attributes:	(port) <u>timing_case_computed_value</u> on page 1021 (pin) <u>timing_case_computed_value_by_mode</u> on page 986 (port) <u>timing_case_computed_value_by_mode</u> on page 1022

timing_case_computed_value_by_mode

```
timing_case_computed_value_by_mode {{mode [0 | 1]} [{mode [0 | 1]}]...}
```

Read-only pin attribute. For each timing mode, indicates if the value of this pin was computed to have a constant logic value. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Examples

- The following example returns the logic value of pin RN for modes a and b:

```
rc:/> get_attribute timing_case_computed_value_by_mode [find my_flop -pin RN]
{/designs/attr_flop/modes/b 0} {/designs/attr_flop/modes/a 1}
```

- The following example sets the `timing_case_logic_value_by_mode` attribute to 0 in mode a at input I1 of instance p0009, which is a nand2. The `timing_case_computed_value_by_mode` attribute on output Z of the nand2 returns the propagated logic value.

```
rc:/> set_attribute timing_case_logic_value_by_mode {{a 0}} \
/designs/add/instances_comb/p0009A/I1
Setting attribute of pin 'I1': 'timing_case_logic_value_by_mode' =
{/designs/add/modes/a 0}
rc:/> get_attribute timing_case_computed_value_by_mode \
/designs/add/instances_comb/p0009A/Z
{/designs/add/modes/a 1}
```

- The following example sets the `timing_case_logic_value_by_mode` attribute at the input pin of inverter g3. The `timing_case_computed_value_by_mode` in output pin Y shows the computed value (which is the inverted value applied at the input).

```
rc:/designs/rct/instances_comb/g3> set_attribute
timing_case_logic_value_by_mode {{a 1} {b 1}} [find . -pin A]
Setting attribute of pin 'A': 'timing_case_logic_value_by_mode' =
{/designs/rct/modes/b 1} {/designs/rct/modes/a 1}
rc:/designs/rct/instances_comb/g3> get_attr
timing_case_computed_value_by_mode [find . -pin Y]
{/designs/rct/modes/b 0} {/designs/rct/modes/a 0}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands: [report clocks](#)
 [report qor](#)
 [report timing](#)

Attribute Reference for Encounter RTL Compiler

Analysis—Pin Attributes

[write encounter](#)

[write sdc](#)

[write script](#)

Related commands: [create mode](#)

[read sdc](#)

Related attributes: (instance) [disabled arcs by mode](#) on page 561

(pin) [external delays by mode](#) on page 970

(port) [external delays by mode](#) on page 1008

(design) [latch borrow by mode](#) on page 548

(instance) [latch borrow by mode](#) on page 565

(design) [latch max borrow by mode](#) on page 551

(instance) [latch max borrow by mode](#) on page 568

(pin) [propagated clocks by mode](#) on page 977

(port) [propagated clocks by mode](#) on page 1016

(design) [slack by mode](#) on page 944

(pin) [slack by mode](#) on page 981

(port) [slack by mode](#) on page 1019

(cost group) [slack by mode](#) on page 1060

(pin) [timing case computed value](#) on page 984

(port) [timing case computed value by mode](#) on page 1022

instance) [timing case disabled arcs by mode](#) on page 955

(instance) [timing case disabled arcs by mode](#) on page 955

(pin) [timing case logic value by mode](#) on page 609

(port) [timing case logic value by mode](#) on page 660

timing_info

`timing_info string`

Read-only `pin` attribute. Returns a Tcl list containing other Tcl lists about timing path information. If the pin is both a timing startpoint and an endpoint, then this attribute returns information from the endpoint. The outer Tcl list corresponds to a set of timing paths that share a common set of launching clock edge, capturing clock edge, and timing exceptions that are being applied. The inner Tcl lists correspond to name/value pairs that provide information about that particular set of timing paths. The supported name/value pairs are:

- **name: launch**
value: A Tcl list containing a clock object and a `R` or `F` character that indicates whether the launching clock edge is the rising or falling edge of the clock waveform. If the launch was not relative to a particular clock, for example at a register with no clock waveform, then `unclocked` is the value.
- **name: capture**
value: A Tcl list containing a clock object and a `R` or `F` character that indicates whether the capturing clock edge is the rising or falling edge of the clock waveform. If the capture was not relative to a particular clock, for example at a register with no clock waveform, then `unclocked` is the value.
- **name: cost_group**
value: The `cost_group` object that this particular set of timing paths is being applied to. Unconstrained paths are always applied to the default cost group.
- **name: exceptions**
value: A Tcl list of exception objects that are being applied to the timing paths. The list is empty if no timing exceptions are being applied.
- **name: mode**
value: The `mode` object that the set of paths belong to. This value is provided only in the presence of multi-mode constraints.
- **name: constraint**
value: The timing constraint value for this set of paths is in picoseconds. If the paths are unconstrained, then the value is listed as `no_value`.
- **name: slack**

Attribute Reference for Encounter RTL Compiler

Analysis—Pin Attributes

value: A Tcl list containing the worst timing slack for this set of paths for rising and falling transitions at the pin. The values are in picoseconds. If a particular transition is unconstrained, then the value is listed as `no_value`.

- **name: `input_delay`**

value: A Tcl list containing the input delays in picoseconds for the rising and falling worst-slack paths. The values are in picoseconds. If a particular input delay is not valid, then the value is listed as `no_value`.

- **name: `output_delay`**

value: A Tcl list containing the output delays in picoseconds for the rising and falling worst-slack paths. The values are in picoseconds. If a particular output path is not valid, then the value is listed as `no_value`.

For certain pins in the design, the timer has two sets of path information. These are pins that are both timing startpoints and also timing endpoints. The `startpoint` and `endpoint` attributes can be used to locate such pins, but these are typically clock inputs of sequential cells, D pins of latches, or pins at which timing paths have been broken using the `break_timing_paths` attribute. At these special pins the `timing_info` attribute will return information about the paths ending at that particular pin. To query about the paths that start at the particular pin instead, use the `timing_info_favor_startpoint` attribute. At pins and ports in the design that are not both timing startpoints and endpoints, the `timing_info` and the `timing_info_favor_startpoint` attributes return the same information.

Examples

- The following example returns timing information from the B pin for the n0002D instance:

```
rc:/> get_attribute timing_info n0002D/B
{launch {/designs/add/timing/clock_domains/domain_1/CK100 R}
capture {/designs/add/timing/clock_domains/domain_1/CK200 R}
cost_group /designs/add/timing/cost_groups/default
exceptions /designs/add/timing/exceptions/multi_cycles/mc constraint 200.0
slack {-261.7 -256.7} input_delay {10.0 10.0} output_delay {451.7 446.7}}
{launch {/designs/add/timing/clock_domains/domain_1/CK100 R}
capture {/designs/add/timing/clock_domains/domain_1/CK100 R}
cost_group /designs/add/timing/cost_groups/default
exceptions /designs/add/timing/exceptions/multi_cycles/mc
constraint 200.0 slack {-261.7 -256.7}
input_delay {10.0 10.0} output_delay {451.7 446.7}}
```

Attribute Reference for Encounter RTL Compiler

Analysis—Pin Attributes

- The following example shows the timing report:

```
rc:/> report timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
          : The design is 'add'.
          : Use 'report timing -lint' for more information.
=====
Generated by:           Version
Generated on:           Date
Module:                add
Technology library:    tutorial 1.0
Operating conditions:  typical_case (balanced_tree)
Wireload mode:         enclosed
=====

      Pin        Type     Fanout Load Slew Delay Arrival
                  (fF)   (ps)   (ps)   (ps)
-----
(clock CK100)  launch
(in_del_1)     ext delay
b[0]           in port   2 25.6   0   +0   10 R
n0002D/B
n0002D/Y       nand2    1 20.4   46  +123  133 F
groupyi/n0002D_Y
  n0001D78/A
  n0001D78/Y   invl     1 15.4   16  +85   218 R
groupyi/n0001D_A
n0001D/A
n0001D/Y       xor2     1 20.4   57  +144  362 R
out_reg_1/D    fflopd
out_reg_1/CK   setup
----- (clock CK200)  capture
                                         200 R
-----
Exception : 'multi_cycles/mc' launch shift 1, capture shift 1
Timing slack : -262ps (TIMING VIOLATION)
Start-point : b[0]
End-point   : out_reg_1/D
```

- The following example Tcl script formats the `timing_info` attribute to make it easier to view:

```
proc timing_info {point} {
    foreach path_set [get_attribute timing_info $point] {
        foreach {name value} $path_set {
            puts "$name : $value"
        }
        puts ""
    }

# an example call to 'timing_info'
timing_info my_reg/D
```

Related Information

Related command: [report timing](#)

Related attributes:	(port) timing_info on page 1024
	(pin) timing_info_favor_startpoint on page 991
	(port) timing_info_favor_startpoint on page 1026

timing_info_favor_startpoint

`timing_info_favor_startpoint` *Tcl_list*

Read-only pin attribute. Returns a Tcl list of information about timing paths. If the pin is both a timing startpoint and an endpoint, then this attribute returns information from the startpoint.

Related Information

Related command:	report_timing
Related attributes:	(port) timing_info_favor_startpoint on page 1026
	(pin) timing_info on page 988
	(port) timing_info on page 1024

Example

- The following example finds timing startpoints of paths that are disabled by the `path_disable` exception called `my_disable`:

```
# find all timing startpoints (actually launch points)
foreach point [filter launcher true [find / -pin -port *]] {
    set seen 0
    # loop over all path sets
    foreach path_set [get_attribute timing_info_favor_startpoint $point] {
        # collect name value pairs into an array for easier processing
        array set values $path_set

        # loop over all exceptions here
        foreach except $values(exceptions) {
            if {[string equal [basename $except] "my_disable"]} {
                # found "my_disable"
                puts "startpoint $point"
                set seen 1
                break
            }
        }
        if {$seen} {
            # already saw my_disable here
            break
        }
    }
}
```

unique_versions

`unique_versions string`

Read-only `pin` attribute. Returns a list of the unique uses of a pin from a collection of pins. An object can have multiple object versions in the design. Use this attribute to get all the objects it represents.

Example

The following command returns a list of the unique uses of a pin:

```
get_attribute unique_versions [find /des/* -pin pin_name]
```

Related Information

Related command: [report timing](#)

Related attributes: (constant) [unique_versions](#) on page 959

(instance) [unique_versions](#) on page 957

(net) [unique_versions](#) on page 1002

(port) [unique_versions](#) on page 1026

(subport) [unique_versions](#) on page 1042

wire_capacitance

`wire_capacitance {no_value | float}`

Read-only `pin` attribute. Returns the net capacitance for the pin in femtofarads. Resolution is 1/10. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes: (constant) [wire_capacitance](#) on page 959

(port) [wire_capacitance](#) on page 1027

(subport) [wire_capacitance](#) on page 1042

wire_length

```
wire_length {no_value | float}
```

Read-only pin attribute. Returns the wire length of the net connected to the pin.

Related Information

Related attributes:	(constant) wire_length on page 960
	(port) wire_length on page 1027
	(subport) wire_length on page 1043

wire_resistance

```
wire_resistance {no_value | float}
```

Read-only pin attribute. Returns the net resistance for the pin in kilohms. Resolution is 1/1000. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes:	(constant) wire_resistance on page 960
	(port) wire_resistance on page 1028
	(subport) wire_resistance on page 1043

wireload_model

```
wireload_model string
```

Read-only pin attribute. Retrieves the wire-load model of the parent module. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Pgpin Attributes

Contain information about power and ground instance pins. Pgpin objects are stored in `pgpins_in` and `pgpins_out` directories.

- To set a pgpin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pgpin instance/pin]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pgpin instance/pin]
```

direction

`direction {inout | in | out}`

Read-only pgpin attribute. Returns the direction of the pgpin.

Note: If the direction attribute of a pgpin is missing in the Liberty library, RTL Compiler sets the direction to `inout` by default.

Related Information

Related attributes:	(pin) direction on page 968
	(port) direction on page 1006
	(port bus) direction on page 1029
	(subport) direction on page 1040
	(subport bus) direction on page 1044

drivers

`drivers {constant | pin | port | subport}`

Read-only pgpin attribute. Returns the drivers for the pgpin.

Related Information

Related attributes:	(net) drivers on page 1000
	(pin) drivers on page 968
	(port) drivers on page 1006
	(subport) drivers on page 1040

libpin

`libpin string`

Read-only pgpin attribute. Returns the library pin associated with a pgpin of a mapped instance.

loads

`loads { pin | port | subport}`

Read-only pgpin attribute. Returns the hierarchical path to the pins or ports loading this pgpin.

Related Information

Related attributes:	(net) loads on page 1001
	(pin) loads on page 973
	(port) loads on page 1011
	(subport) drivers on page 1040

net

`net string`

Read-only pgpin attribute. Returns the net connected to the pgpin.

Related Information

Related attributes:	(pin) net on page 974
	(port) net on page 1012
	(subport) net on page 1041

unique_versions

`unique_versions string`

Read-only pgpin attribute. Returns a list of the unique uses of a pin from a collection of pins. An object can have multiple object versions in the design. Use this attribute to get all the objects it represents.

Example

The following command returns a list of the unique uses of a pin:

```
get_attribute unique_versions [find /des*/* -pgpin instance/pin_name]
```

Related Information

Related command: [report timing](#)

Related attributes: (constant) [unique_versions](#) on page 959

(instance) [unique_versions](#) on page 957

(net) [unique_versions](#) on page 1002

(pin) [unique_versions](#) on page 992

(port) [unique_versions](#) on page 1026

(subport) [unique_versions](#) on page 1042

Pin Bus Attributes

Contain information about the specified instance pin bus. A pin bus is a bussed connection point on an instance. Pin_bus objects are stored in `pin_busses_in` and `pin_busses_out` directories. These attributes are read-only attributes, so you cannot set their values.

- To get a `pin_bus` attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pin_bus instance/bus]
```

Note: You may need to specify more than one instance to serve as the search path that will uniquely identify the relevant pin bus.

bits

`bits string`

Read-only `pin_bus` attribute. Returns a list of individual bits that constitute the bus. The list includes the full pathnames of each individual pin object.

Related Information

Related attributes: (port bus) [bits](#) on page 1029
 (subport bus) [bits](#) on page 1044

Net Attributes

Contain information about a net in the specified design. A net can be a top-level net or can belong to a hierarchical instance.

- To set a net attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -net instance/net]
```

- To get a net attribute value, type

```
get_attribute attribute_name [find /des*/design -net instance/net]
```

Note: You may need to specify several instances to uniquely identify the net.

Note: These attributes are located at `/designs/design/nets`.

constant

```
constant {no_constant | constant_0 | constant_1}
```

Default: no_constant

Read-write net attribute. Specifies whether the net is driven by a constant.

driven_by_supply0

```
driven_by_supply0 {true | false}
```

Read-only net attribute. Indicates whether the specified net is driven by a supply0 net (ground net).

Note: This attribute is only supported in the structural flow.

Example

Assume the following HDL file was read in:

```
module test8(in1, outa, outb, out_vss);
    input in1;
    output outa, outb, out_vss;
    wire in1;
    wire outa, outb, out_vss;
    supply0 vss;
    supply1 vdd;
    assign out_vss = vss;
    buf i1(outa, vdd);
    buf i2(outb, vss);
endmodule
```

Attribute Reference for Encounter RTL Compiler

Analysis—Net Attributes

To check if net `out_vss` is driven by a `supply0` net, use the following command:

```
rc:/> get_attr driven_by_supply0 [find /des*/test8 -net out_vss]
true
```

To check if net `outb` is driven by a `supply0` net, use the following command:

```
rc:/> get_attr driven_by_supply0 [find /des*/test8 -net outb]
false
```

driven_by_supply1

```
driven_by_supply1 {true | false}
```

Read-only net attribute. Indicates whether the specified net is driven by a `supply1` net (power net).

Note: This attribute is only supported in the structural flow.

Example

Assume the following HDL file was read in:

```
module test8(in1, outa, outb, out_vss);
    input in1;
    output outa, outb, out_vss;
    wire in1;
    wire outa, outb, out_vss;
    supply0 vss;
    supply1 vdd;
    assign out_vdd = vdd;
    buf i1(outa, vdd);
    buf i2(outb, vss);
endmodule
```

To check if net `out_vdd` is driven by a `supply1` net, use the following command:

```
rc:/> get_attr driven_by_supply1 [find /des*/test8 -net out_vdd]
true
```

To check if net `outa` is driven by a `supply1` net, use the following command:

```
rc:/> get_attr driven_by_supply1 [find /des*/test8 -net outa]
false
```

drivers

drivers *string*

Read-only net attribute. Returns the hierarchical path to the pin driving this net.

Related Information

Related attributes:	(pgpin) drivers on page 994
	(pin) drivers on page 968
	(port) drivers on page 1006
	(subport) drivers on page 1040

ideal

ideal {true | false}

Read-only net attribute. Indicates whether the specified net is ideal.

Valid causes for a net being ideal are:

- An `ideal_driver` attribute is set on a driver of the net
- An `ideal_network` attribute is set on a driver in the fan in of the net
- A clock pin for a sequential instance is on the net
- A logic constant drives the net
- The net is connected to a pin whose `propagated_ideal_network` attribute is set to true.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

```
rc:/> get_attribute ideal /designs/EXAMPLE/nets/ck
true
```

Related Information

[Handling Ideal Nets in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

Affected by these attributes: (pin) [ideal_driver](#) on page 595
 (port) [ideal_driver](#) on page 645

is_net_part_of_bus

`is_net_part_of_bus {false | true}`

Read-only [net](#) attribute. Indicates whether the specified net is a single net or part of a bus.

Related Information

Affects this command: [change_names](#)

loads

`loads string`

Read-only [net](#) attribute. Returns the hierarchical path to the pins loading this net.

num_drivers

`num_drivers integer`

Read-only [net](#) attribute. Lists the number of drivers on the net.

num_loads

`num_loads integer`

Read-only [net](#) attribute. Lists the number of loads on the net.

type

```
type {wire | wand | wor | supply0 | supply1}
```

Default: wire

Read-write net attribute. Returns the type of the net.

unique_versions

```
unique_versions string
```

Read-only net attribute. Returns a list of the unique uses of a net from a collection of nets. An object can have multiple object versions in the design. Use this attribute to get all the objects it represents.

Example

The following command returns a list of the unique uses of a net from a collection of nets:

```
get_attribute unique_versions [find /des*/* -net net_name]
```

Related Information

Related command: [report timing](#)

Related attributes: (constant) [unique_versions](#) on page 959

(instance) [unique_versions](#) on page 957

(pin) [unique_versions](#) on page 992

(port) [unique_versions](#) on page 1026

(subport) [unique_versions](#) on page 1042

Port Attributes

Contain information about single bit connection points in the specified top-level design. Port objects are stored in `ports_in` and `ports_out` directories.

- To set a port attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -port name]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -port name]
```

Note: These attributes are located at `/designs/design/ports_in` and `/designs/design/ports_out`

bus

`bus string`

Read-only port attribute. Specifies the name of the bus that this port belongs to.

Related Information

Related attributes: (subport) bus on page 1039

capturer

`capturer {true | false}`

Read-only port attribute. Indicates if the pin's required time was derived through static analysis from a setup check (`false`) or user-defined output delay constraint (`true`). This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: report_timing

Related attribute (pin) capturer on page 963

causes_ideal_net

```
causes_ideal_net {true | false}
```

Read-only port attribute. Indicates whether the specified port is causing its net to become an ideal net.

RTL Compiler implicitly sets an ideal net if some port is

- A pin of a sequential element that is an asynchronous set or reset. You can check for this attribute by using the `ls -long -attribute` command on the pin or libcell.
- A pin of a sequential element that is a clock pin. You can check for this attribute by using the `ls -long -attribute` command on the pin or libcell.
- Any input pin of a sequential element that has no setup arc. That is, any sequential element that may be asynchronous.

Example

The following example queries whether the `in1[1]` input port is causing its net to become ideal:

```
rc:/> get_attribute causes_ideal_net {/designs/trance/ports_in/in1[0]}  
false
```

Related Information

Related attributes:

(pin) causes_ideal_net on page 963	(subport) causes_ideal_net on page 1039
time_recovery_arcs on page 535	

clock_sources_inverted

`clock_sources_inverted string`

Read-only `port` attribute. Returns a Tcl list of clock objects that were applied in the inverted sense to the specified port.

Related Information

Related attributes:	inverted_sources on page 673
	(pin) clock_sources_inverted on page 966
	(pin) clock_sources_non_inverted on page 967
	(port) clock_sources_non_inverted on page 1005
	(pin) propagated_clocks on page 975
	(port) propagated_clocks on page 1014
	(pin) propagated_clocks_by_mode on page 977
	(port) propagated_clocks_by_mode on page 1016

clock_sources_non_inverted

`clock_sources_non_inverted string`

Read-only `port` attribute. Returns a Tcl list of clock objects that were applied in the non-inverted sense to the specified port.

Related Information

Related attributes:	non_inverted_sources on page 673
	(pin) clock_sources_inverted on page 966
	(port) clock_sources_inverted on page 1005
	((pin) clock_sources_non_inverted on page 967)
	(pin) propagated_clocks on page 975
	(port) propagated_clocks on page 1014
	(pin) propagated_clocks_by_mode on page 977
	(port) propagated_clocks_by_mode on page 1016

connect_delay

```
connect_delay {{no_value no_value}| {float float}}
```

Read-only port attribute. Returns the rise and fall connect delay to this port in picoseconds. Resolution is 1. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report timing](#)

Related attribute: (pin) [connect_delay](#) on page 967

direction

```
direction {in | out | inout}
```

Read-only port attribute. Returns the direction of the port.

Related Information

Related attributes: (pin) [direction](#) on page 968

(port bus) [direction](#) on page 1029

(subport) [direction](#) on page 1040

(subport bus) [direction](#) on page 1044

drivers

```
drivers {constant | | pin | port | subport}
```

Read-only port attribute. Returns the drivers for the port.

Related Information

Related attributes: (net) [drivers](#) on page 1000

(pgpin) [drivers](#) on page 994

(pin) [drivers](#) on page 968

(subport) [drivers](#) on page 1040

endpoint

endpoint {true | false}

Read-only port attribute. Indicates if the port is the endpoint of a timing path. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report timing](#)

Related attribute: (pin) [endpoint](#) on page 968

exceptions

exceptions *string*

Read-only port attribute. Returns a list of all the timing exceptions that were applied to the specified port.

Related Information

Affected by these commands: [path_adjust](#)
[path_delay](#)
[path_disable](#)
[path_group](#)
[multi_cycle](#)

Related attributes: (clock) [exceptions](#) on page 1055
(clock_domain) [exceptions](#) on page 1058
(cost_group) [exceptions](#) on page 1059
(external_delay) [exceptions](#) on page 1070
(instance) [exceptions](#) on page 948
(pin) [exceptions](#) on page 969

external_delays

`external_delays string`

Read-only `port` attribute. Returns a list of the external delay objects associated with this port.

Related information

Set by this command: [external_delay](#)

Related Attribute [external_delays](#) on page 969

external_delays_by_mode

`external_delays_by_mode {mode external_delay} [{mode external_delay}]...`

Read-only `port` attribute. Returns a Tcl list of external delay objects by mode that are associated with a port.

Examples

After setting the SDC `set_input_delay`, `set_output_delay`, and `set_max_delay` commands or the RTL Compiler `external_delay` command, the `external_delays_by_mode` attribute returns a Tcl list of external delay objects associated with a port.

- The following example shows that an input delay was annotated on the `QN` port in mode `a` but not in mode `b`:

```
rc:/> get_attr external_delays_by_mode /designs/attr_flop/ports_out/QN
{/designs/attr_flop/modes/b \
/designs/attr_flop/modes/b/external_delays/ou_del_1} \
{/designs/attr_flop/modes/a \
/designs/attr_flop/modes/a/external_delays/ou_del_1}
```

- You can then use the `ls -al` command on the external delay for mode `a` to see the associated attributes, as shown in the following example:

```
rc:/> ls -al /designs/attr_flop/modes/b/external_delays/ou_del_1
/designs/attr_flop/modes/b/external_delays/ou_del_1 (external_delay)
All attributes:
clock = /designs/attr_flop/modes/b/clock_domains/domain_1/clock_b
clock_network_latency_included = false
clock_rise = true
clock_source_latency_included = false
delay = 0.0 0.0 0.0 0.0
```

Attribute Reference for Encounter RTL Compiler

Analysis—Port Attributes

```
exceptions =
external_delay_pins = /designs/attr_flop/ports_out/QN \
/designs/attr_flop/ports_out/Q
input_delay = false
level_sensitive = false
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands: [report clocks](#)

[report qor](#)

[report timing](#)

[write encounter](#)

[write sdc](#)

[write script](#)

Related commands: [create mode](#)

[read sdc](#)

Related attributes: [disabled arcs by mode](#) on page 561

(pin) [external delays by mode](#) on page 970

(design) [latch borrow by mode](#) on page 548

(instance) [latch borrow by mode](#) on page 565

(design) [latch max borrow by mode](#) on page 551

(instance) [latch max borrow by mode](#) on page 568

(pin) [latch max borrow by mode](#) on page 598

(pin) [propagated clocks by mode](#) on page 977

(port) [propagated clocks by mode](#) on page 1016

(design) [slack by mode](#) on page 944

(pin) [slack by mode](#) on page 981

(port) [slack by mode](#) on page 1019

(cost group) [slack by mode](#) on page 1060

(pin) [timing case computed value by mode](#) on page 986

Attribute Reference for Encounter RTL Compiler

Analysis—Port Attributes

(port) [timing case computed value by mode](#) on page 1022
instance) [timing case disabled arcs by mode](#) on page 955
(pin) [timing case logic value by mode](#) on page 609
(port) [timing case logic value by mode](#) on page 660

has_min_delay

```
has_min_delay {true | false}
```

Read-only [port](#) attribute. Indicates whether the minimum delays were computed for the port to honor the data-to-data checks timing constraints. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

The following command checks whether minimum delays were computed for port `IN1`. The returned value indicates that the minimum delays were computed.

```
rc:/> get_attribute has_min_delay [find /des*/top -port IN1]  
true
```

Related Information

Related command: [report timing](#)

Related attributes:

- (pin) [has_min_delay](#) on page 972
- (pin) [min_slew](#) on page 973
- (port) [min_slew](#) on page 1011
- (port) [min_port_delay](#) on page 1012

launcher

```
launcher {true | false}
```

Read-only [port](#) attribute. Indicates if the arrival time at this port was derived through static analysis from a clock signal arriving at an edge-triggered or level-sensitive point, or from a user-defined input delay. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report timing](#)

Related attribute: (pin) [launcher](#) on page 972

loads

```
loads { pin | port | subport }
```

Read-only [port](#) attribute. Returns the hierarchical path to the pins or ports loading this port.

Related Information

Related attributes: (net) [loads](#) on page 1001

(pin) [loads](#) on page 973

(pgpin) [loads](#) on page 1001

(subport) [loads](#) on page 1041

min_slew

```
min_slew Tcl_list
```

Read-only [port](#) attribute. Returns a Tcl list containing the computed rise and fall minimum slew values, respectively, in picoseconds. This attribute is only computed if the minimum delay must be computed for the port to honor the data-to-data timing constraints. You can use the [has_min_delay](#) port attribute to check if the [min_slew](#) attribute has been computed.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the [ls](#) command by default.

Example

The following command retrieves the minimum slew values on the IN1 port.

```
rc:/> get_att min_slew [find /des*/top -port IN1]
834.8 662.0
```

Related Information

Related command:	report timing
Related attributes:	(pin) has_min_delay on page 972
	(port) has_min_delay on page 1010
	(pin) min_slew on page 973
	(clock) slew on page 674
	(port) slew on page 1020

min_port_delay

`min_port_delay Tcl_list`

Read-only [port](#) attribute. Returns a Tcl list containing the computed rise and fall minimum delay values, respectively, in picoseconds. This attribute is only computed if the minimum delay must be computed for this port to honor the data-to-data timing constraints. You can use the [has_min_delay](#) port attribute to check if the [min_port_delay](#) attribute has been computed.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command:	report timing
Related attributes:	(pin) has_min_delay on page 972
	(port) has_min_delay on page 1010
	(port) connect_delay on page 1006

net

`net string`

Read-only [port](#) attribute. Returns the net connected to the port.

Attribute Reference for Encounter RTL Compiler

Analysis—Port Attributes

Related Information

Related attributes:

- (pin) [net](#) on page 974
- (pgpin) [net](#) on page 995
- (subport) [net](#) on page 1041

pin_capacitance

```
pin_capacitance {no_value | float float}
```

Read-only [port](#) attribute. Returns the rise and fall capacitances for the port in femtofarads. Resolution is 1/10. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes:

- (constant) [pin_capacitance](#) on page 958
- (pin) [pin_capacitance](#) on page 975
- (subport) [pin_capacitance](#) on page 1041

port_delay

```
port_delay {rise fall}
```

Read-only [port](#) attribute. Returns the rise and fall delay value in picoseconds that is attributed to this port in picoseconds. Resolution is 1. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report timing](#)
Related attribute: (port) [connect_delay](#) on page 1006

propagated_clocks

`propagated_clocks port_name`

Read-only `port` attribute. Returns a Tcl list of clock information that has propagated to the specified port. Each element of the list contains information about a single clock that was propagated.

The clock information contained in the Tcl list can be easily converted into an associative array using the Tcl command `array get`. This provides a convenient method to query for information about propagated clocks in a design. The keys of the associative array are:

- `clock` — The clock object that has been propagated.
- `phase` — A string value indicating whether the clock has been inverted ("+" or "-").
- `clock_source_late_latency` — A Tcl list containing any source latency values that have been picked up from the `clock_source_late_latency` attribute on pins or ports in the clock network. This does not include latency values from the clock object itself.
- `clock_network_late_latency` — A Tcl list containing any network latency values that have been picked up from the `clock_network_late_latency` attribute on pins or ports in the clock network. This does not include latency values from the clock object itself.
- `clock_setup_uncertainty` — A Tcl list containing any uncertainty values that have been picked up from the `clock_setup_uncertainty` attribute on pins or ports in the clock network. This does not include uncertainty values from the clock object itself.

Example

The following Tcl code shows an example usage of this attribute:

```
proc propagated_clock_info {port} {  
    foreach clock_info [get_attribute propagated_clocks $port] {  
        array set info_array $clock_info  
        puts "clock : [basename $info_array(clock)]"  
        puts "phase : $info_array(phase)"  
        puts "source_latency : $info_array(clock_source_late_latency)"  
        puts "network_latency : $info_array(clock_network_late_latency)"  
        puts "uncertainty : $info_array(clock_setup_uncertainty)"  
        puts ""  
    }  
}
```

Attribute Reference for Encounter RTL Compiler

Analysis—Port Attributes

Related Information

[Combinational Feedback Loops in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:	report clocks report qor report timing write_encounter write_sdc write_script
Related command:	read_sdc
Related attributes:	disabled_arcs on page 560 (pin) external_delays on page 969 (port) external_delays on page 1008 (design) latch_borrow on page 547 (instance) latch_borrow on page 564 (design) latch_max_borrow on page 550 (instance) latch_max_borrow on page 567 (pin) propagated_clocks on page 975 (port) propagated_clocks on page 1014 (design) slack on page 944 (pin) slack on page 981 (port) slack on page 1018 (cost group) slack on page 1060 (pin) timing_case_computed_value on page 984 (port) timing_case_computed_value on page 1021 instance) timing_case_disabled_arcs on page 954 (pin) timing_case_logic_value on page 608 (port) timing_case_logic_value on page 659

propagated_clocks_by_mode

```
propagated_clocks_by_mode {{mode_name_1 [clock_info_1]...}  
[{mode_name_2 [clock_info_2]...}]}...}
```

Read-only port attribute. Returns a Tcl list of lists consisting of the mode, related clocks, and clock characteristics, such as latency and uncertainty.

The clock information contained in the Tcl list can be easily converted into an associative array using the Tcl command `array get`. This provides a convenient method to query for information about propagated clocks in a design. The keys of the associative array are:

- `clock` — The clock object that has been propagated.
- `phase` — A string value indicating whether the clock has been inverted ("+" or "-").
- `clock_source_late_latency` — A Tcl list containing any source latency values that have been picked up from the `clock_source_late_latency` attribute on pins or ports in the clock network. This does not include latency values from the clock object itself.
- `clock_network_late_latency` — A Tcl list containing any network latency values that have been picked up from the `clock_network_late_latency` attribute on pins or ports in the clock network. This does not include latency values from the clock object itself.
- `clock_setup_uncertainty` — A Tcl list containing any uncertainty values that have been picked up from the `clock_setup_uncertainty` attribute on pins or ports in the clock network. This does not include uncertainty values from the clock object itself.

Example

- The following command returns clock information for the `clk` port:

```
rc:/> get_attribute propagated_clocks_by_mode [find / -port ck]  
{/designs/add/instances_seq/out_reg[1]/pins_in/CLK {/designs/add/modes/b  
{{clock /designs/add/modes/b/clock_domains/domain_1/ck phase +  
clock_source_late_latency {no_value no_value no_value no_value}  
clock_network_late_latency {no_value no_value no_value no_value}  
clock_setup_uncertainty {no_value no_value}}}} {/designs/add/modes/a  
{{clock /designs/add/modes/a/clock_domains/domain_1/ck phase  
clock_source_late_latency {no_value no_value no_value no_value}  
clock_network_late_latency {no_value no_value no_value no_value}  
clock_setup_uncertainty {no_value no_value}}}}}
```

Attribute Reference for Encounter RTL Compiler

Analysis—Port Attributes

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.](#)

Affects these commands:

[report clocks](#)

[report qor](#)

[report timing](#)

[write_encounter](#)

[write_sdc](#)

[write_script](#)

Related commands:

[create_mode](#)

[read_sdc](#)

Related attributes:

[disabled_arcs_by_mode](#) on page 561

(pin) [external_delays_by_mode](#) on page 970

(port) [external_delays_by_mode](#) on page 1008

(design) [latch_borrow_by_mode](#) on page 548

(instance) [latch_borrow_by_mode](#) on page 565

(design) [latch_max_borrow_by_mode](#) on page 551

(instance) [latch_max_borrow_by_mode](#) on page 568

(pin) [propagated_clocks_by_mode](#) on page 977

(design) [slack_by_mode](#) on page 944

(pin) [slack_by_mode](#) on page 981

(port) [slack_by_mode](#) on page 1019

(cost group) [slack_by_mode](#) on page 1060

(pin) [timing_case_computed_value_by_mode](#) on page 986

(port) [timing_case_computed_value_by_mode](#) on page 1022

(instance) [timing_case_disabled_arcs_by_mode](#) on page 955

(pin) [timing_case_logic_value_by_mode](#) on page 609

(port) [timing_case_logic_value_by_mode](#) on page 660

rf_slack

```
rf_slack rise fall
```

Read-only [port](#) attribute. Returns the slack for the rising and falling edges in picoseconds. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

```
rc:/designs/top> get_att rf_slack [find / -port out[4]]  
1344.8 1327.7
```

Related Information

Related attributes:	(pin) rf_slack on page 980 (design) slack on page 944 (instance) slack on page 953 (pin) slack on page 981 (port) slack on page 1018 (cost group) slack on page 1060
---------------------	---

slack

```
slack {no_value | float}
```

Read-only [port](#) attribute. Returns the slack of the port in picoseconds. The resolution is 1. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command:	report timing
Related attributes:	(pin) rf_slack on page 980 ((port) rf_slack on page 1018 (design) slack on page 944 (instance) slack on page 953 (pin) slack on page 981 (cost group) slack on page 1060

Attribute Reference for Encounter RTL Compiler

Analysis—Port Attributes

slack_by_mode

```
slack_by_mode {{mode_name_1 delay_value} [{mode_name_2 delay_value}]...}
```

Read-only [port](#) attribute. Returns a Tcl list of lists of slack values in picoseconds for each timing mode for a port.

Example

The following example shows the slack in mode a and mode b:

```
rc:/>get_attribute slack_by_mode /designs/top/ports_out/out
{/designs/top/modes/b -103.9} {/designs/top/modes/a -43.9}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands: [report clocks](#)

[report qor](#)

[report timing](#)

[write encounter](#)

[write sdc](#)

[write script](#)

Related commands: [create mode](#)

[read sdc](#)

Related attributes: [disabled arcs by mode](#) on page 561

(pin) [external delays by mode](#) on page 970

(port) [external delays by mode](#) on page 1008

(design) [latch borrow by mode](#) on page 548

(instance) [latch borrow by mode](#) on page 565

(design) [latch max borrow by mode](#) on page 551

(instance) [latch max borrow by mode](#) on page 568

(pin) [propagated clocks by mode](#) on page 977

(port) [propagated clocks by mode](#) on page 1016

(cost group) [slack by mode](#) on page 1060

Attribute Reference for Encounter RTL Compiler Analysis—Port Attributes

- (design) slack by mode on page 944
- (pin) slack by mode on page 981
- (pin) timing case computed value by mode on page 986
- (port) timing case computed value by mode on page 1022
- (instance) timing case disabled arcs by mode on page 955
- (pin) timing case logic value by mode on page 609
- (port) timing case logic value by mode on page 660

slew

slew {*rise fall*}

Read-only port attribute. Returns the computed rise and fall slew values, respectively, in picoseconds. The values are returned as a Tcl list.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes: (clock) [slew](#) on page 674
(pin) [slew](#) on page 983

slew_by_mode

```
slew_by_mode {{mode_name_1 rise fall}} [{{mode_name_2 rise fall}}]...}
```

Read-only port attribute. Returns a Tcl list of lists. Each list contains the mode name followed by the computed rise and fall slew values for the port for that mode, in picoseconds.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

```
rc:/> get_att slew_by_mode /designs/top/ports_out/out
{/designs/top/modes/b 834.8 662.0}
```

Related Information

Related command: [report timing](#)

Related attributes: (pin) [slew by mode](#) on page 983

startpoint

```
startpoint {true | false}
```

Read-only port attribute. Indicates if the pin is the startpoint of a timing path. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report timing](#)

Related attribute: (pin) [startpoint](#) on page 984

timing_case_computed_value

```
timing_case_computed_value {0 | 1 | no_value}
```

Read-only port attribute. Indicates if the value of this port was computed to have a constant logic value. You can use this attribute to create a case analysis report. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

- If one input of an AND gate is set to logic value 0, the output is computed to have a logic value of 0.
- In the following example, the `timing_case_computed_value` indicates a computed logic value of 0 when the SDC `set_case_analysis` command was applied on port `input2`:

```
rc:/>get_attribute timing_case_computed_value [find /-port input2]
0
```

Attribute Reference for Encounter RTL Compiler

Analysis—Port Attributes

Related Information

Affects these commands:	report clocks report qor report timing write encounter write sdc write script
Related commands:	read sdc
Related attributes:	(pin) timing_case_computed_value on page 984

timing_case_computed_value_by_mode

`timing_case_computed_value_by_mode {{mode [0 | 1]} [{mode [0 | 1]}]...}`

Read-only [port](#) attribute. Returns the computed logic constant value for a port for each timing mode.

Example

- The following example returns the logic value of the RN port for modes a and b:

```
rc:/> get_attribute timing_case_computed_value_by_mode [find my_flop -port RN]
{/designs/attr_flop/modes/b 0} {/designs/attr_flop/modes/a 1}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#).

Affects these commands:	report clocks report qor report timing write encounter write sdc write script
Related commands:	create mode

Attribute Reference for Encounter RTL Compiler

Analysis—Port Attributes

read_sdc

Related attributes:

- (instance) [disabled arcs by mode](#) on page 561
- (pin) [external delays by mode](#) on page 970
- (port) [external delays by mode](#) on page 1008
- (design) [latch borrow by mode](#) on page 548
- (instance) [latch borrow by mode](#) on page 565
- (design) [latch max borrow by mode](#) on page 551
- (instance) [latch max borrow by mode](#) on page 568
- (pin) [propagated clocks by mode](#) on page 977
- (port) [propagated clocks by mode](#) on page 1016
- (design) [slack by mode](#) on page 944
- (pin) [slack by mode](#) on page 981
- (port) [slack by mode](#) on page 1019
- (cost group) [slack by mode](#) on page 1060
- (pin) [timing case computed value](#) on page 984
- (pin) [timing case computed value by mode](#) on page 986
- instance) [timing case disabled arcs by mode](#) on page 955
- (instance) [timing case disabled arcs by mode](#) on page 955
- (pin) [timing case logic value by mode](#) on page 609
- (port) [timing case logic value by mode](#) on page 660

timing_info

`timing_info string`

Read-only `port` attribute. Returns a Tcl list containing other Tcl lists about timing path information. If the port is both a timing startpoint and an endpoint, then this attribute returns information from the endpoint. The outer Tcl list corresponds to a set of timing paths that share a common set of launching clock edge, capturing clock edge, and timing exceptions that are being applied. The inner Tcl lists correspond to name/value pairs that provide information about that particular set of timing paths. The supported name/value pairs are:

- `name: launch`

`value`: A Tcl list containing a clock object and a `R` or `F` character that indicates whether the launching clock edge is the rising or falling edge of the clock waveform. If the launch was not relative to a particular clock, for example at a register with no clock waveform, then `unclocked` is the value.

- `name: capture`

`value`: A Tcl list containing a clock object and a `R` or `F` character that indicates whether the capturing clock edge is the rising or falling edge of the clock waveform. If the capture was not relative to a particular clock, for example at a register with no clock waveform, then `unclocked` is the value.

- `name: cost_group`

`value`: The `cost_group` object that this particular set of timing paths is being applied to. Unconstrained paths are always applied to the default cost group.

- `name: exceptions`

`value`: A Tcl list of exception objects that are being applied to the timing paths. The list is empty if no timing exceptions are being applied.

- `name: mode`

`value`: The `mode` object that the set of paths belong to. This value is provided only in the presence of multi-mode constraints.

- `name: constraint`

`value`: The timing constraint value for this set of paths is in picoseconds. If the paths are unconstrained, then the value is listed as `no_value`.

Attribute Reference for Encounter RTL Compiler

Analysis—Port Attributes

- **name: slack**
value: A Tcl list containing the worst timing slack for this set of paths for rising and falling transitions at the pin or port. The values are in picoseconds. The value is listed as `no_value` if a particular transition is unconstrained.
- **name: input_delay**
value: A Tcl list containing the input delays in picoseconds for the rising and falling worst-slack paths. The values are in picoseconds. If a particular input delay is not valid, then the value is listed as `no_value`.
- **name: output_delay**
value: A Tcl list containing the output delays in picoseconds for the rising and falling worst-slack paths. The values are in picoseconds. If a particular output path is not valid, then the value is listed as `no_value`.

For certain ports in the design, the timer has two sets of path information. These are ports that are both timing startpoints and also timing endpoints. The `startpoint` and `endpoint` attributes can be used to locate such ports, but these are typically clock inputs of sequential cells, D pins of latches, or ports at which timing paths have been broken using the `break_timing_paths` attribute. At these special ports the `timing_info` attribute will return information about the paths ending at that particular port. To query about the paths that start at the particular port instead, use the `timing_info_favor_startpoint` attribute. For ports in the design that are not both timing startpoints and endpoints, the `timing_info` and the `timing_info_favor_startpoint` attributes return the same information.

Related Information

Related command:	report timing
Related attributes	(pin) timing_info on page 988
	(pin) timing_info_favor_startpoint on page 991

timing_info_favor_startpoint

`timing_info_favor_startpoint Tcl_list`

Read-only port attribute. Returns a Tcl list of information about timing paths. If the port is both a timing startpoint and an endpoint, then this attribute returns information from the startpoint.

Example

- The following example finds timing startpoints of paths that are disabled by the `path_disable` exception called `my_disable`:

```
# find all timing startpoints (actually launch points)
foreach point [filter launcher true [find / -pin -port *]] {
    set seen 0
    # loop over all path sets
    foreach path_set [get_attribute timing_info_favor_startpoint $point] {
        # collect name value pairs into an array for easier processing
        array set values $path_set
        # loop over all exceptions here
        foreach except $values(exceptions) {
            if {[string equal [basename $except] "my_disable"]} {
                # found "my_disable"
                puts "startpoint $point"
                set seen 1
                break
            }
        }
        if {$seen} {
            # already saw my_disable here
            break
        }
    }
}
```

Related Information

Related command: [report timing](#)

Related attributes (pin) [timing_info_favor_startpoint](#) on page 991

(pin) [timing_info](#) on page 988

(port) [timing_info](#) on page 1024

unique_versions

`unique_versions string`

Read-only port attribute. Returns a list of the unique uses of a port from a collection of ports. An object can have multiple object versions in the design. Use this attribute to get all the objects it represents.

Related Information

Related command:	report timing
Related attributes:	(constant) unique_versions on page 959
	(instance) unique_versions on page 957
	(net) unique_versions on page 1002
	(pin) unique_versions on page 992
	(subport) unique_versions on page 1042

wire_capacitance

`wire_capacitance {no_value | float}`

Read-only [port](#) attribute. Returns the net capacitance for the port in femtofarads. Resolution is 1/10. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes:	(constant) wire_capacitance on page 959
	(pin) wire_capacitance on page 992
	(subport) wire_capacitance on page 1042

wire_length

`wire_length {no_value | float}`

Read-only [port](#) attribute. Returns the wire length of the net connected to the port.

Related Information

Related attributes:	(constant) wire_length on page 960
	(pin) wire_length on page 993
	(subport) wire_length on page 1043

wire_resistance

```
wire_resistance {no_value | float}
```

Read-only port attribute. Returns the net resistance for the port in kilohms. Resolution is 1/1000. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes:

(constant) wire_resistance on page 960
(pin) wire_resistance on page 993
(subport) wire_resistance on page 1043

Port Bus Attributes

Contain information about the bussed input and output ports of a top-level design. These attributes are read-only attributes, so you cannot set their values.

- To get a `port_bus` attribute value, type

```
get_attribute attribute_name \
[find /des*/design -port_bus name]
```

Note: These attributes are located at `/designs/design/port_busses_out` and `/designs/design/port_busses_in`

bits

`bits string`

Read-only `port_bus` attribute. Returns a list of individual bits that constitute the bus. The list includes the full pathnames of each individual port object.

Related Information

Related attributes: (pin bus) [bits](#) on page 997
 (subport bus) [bits](#) on page 1044

direction

`direction {in | out | inout}`

Read-only `port_bus` attribute. Returns the direction of the port.

Related Information

Related attributes: (pin) [direction](#) on page 968
 (port) [direction](#) on page 1006
 (subport) [direction](#) on page 1040
 (subport bus) [direction](#) on page 1044

order

`order integer`

Read-only port bus attribute. Returns the order of the bus in the design port declaration list, the value of the leftmost being the first. The order value of the first bus is 0.

Related Information

Related attribute: (support bus) order on page 1045

Subdesign Attributes

Contain information about the subdesigns in the specified design. Subdesigns correspond to Verilog modules or VHDL entities instantiated in the top-level Verilog module or top-level VHDL entity.

- To set a subdesign attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subdesign name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subdesign name]
```

Note: These attributes are located at `/designs/design/subdesigns`

arch_name

`arch_name string`

Read-only `subdesign` attribute. Returns the name of the Verilog module or the VHDL architecture from which the subdesign is derived.

Related Information

Related command: [read_hdl](#)

Related attribute: [\(design\) arch_name](#) on page 931

area

`area float`

Read-only `subdesign` attribute. Computes the total area of the subdesign, including the net area. The area is specified in terms of the library units. This is a computed attribute. Computed attributes are potentially very time consuming to process and therefore turned off by default.

Related Information

Related command: [report area](#)

Related attributes: [\(libcell\) area](#) on page 188

(design) [area](#) on page 931

cell_area

`cell_area string`

Read-only [subdesign](#) attribute. Returns the cell area of the subdesign. The area is specified in terms of the library units. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report area](#)

Related attributes: [\(libcell\) area](#) on page 188

[\(design\) cell_area](#) on page 935
[\(instance\) cell_area](#) on page 947

Related attributes: [\(libcell\) area](#) on page 188

[\(design\) cell_area](#) on page 932

[\(instance\) cell_area](#) on page 947

cell_count

`cell_count integer`

Read-only [subdesign](#) attribute. Returns the cell count of the subdesign. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report area](#)

Related attributes: [\(design\) cell_count](#) on page 932

constant_0_nets

`constant_0_nets list_of_nets`

Read-only [subdesign](#) attribute. Returns a list of nets driven by constant 0.

Related Information

Related attributes: (design) [constant_0_nets](#) on page 933

constant_1_nets

`constant_1_nets list_of_nets`

Read-only [subdesign](#) attribute. Returns a list of nets driven by constant 1.

Related Information

Related attributes: (design) [constant_1_nets](#) on page 933

entity_name

`entity_name string`

Read-only [subdesign](#) attribute. Returns the name of the Verilog module or the VHDL entity from which the design is derived.

Related Information

Related command: [read_hdl](#)

Related attribute: (design) [entity_name](#) on page 933

hdl_elab_command_params

`hdl_elab_command_params {{parameter value}}...`

Read-only [subdesign](#) attribute. Returns a Tcl-list of the {parameter value} pairs that were specified as value for the -parameters option of the most recently executed elaborate command.

Example

```
rc:/> elaborate -parameters {width 4} mid
...
rc:/> get_attribute hdl_elab_command_params \
[find /des*/test -subdesign mid] {{width 4}}
```

Related Information

Affects this command: [elaborate](#)

Related attribute: (design) [hdl_elab_command_params](#) on page 934

hdl_parameters

`hdl_parameters {{parameter current_value cmdset default_value}...}`

Read-only [subdesign](#) attribute. Returns a Tcl list of Tcl lists. There can be as many Tcl lists as there are parameters and localparams in the specified subdesign. The Tcl list contains four elements:

- The parameter or localparam name
- The current parameter value
- A binary number

If the binary number is 1, it indicates that the parameter value was set through the `elaborate -parameters` command. For localparam, the `status` value will always be 0 because it cannot be overwritten.

- The default value of the parameter before it is overridden during instantiation or using the `elaborate` command.

Example

The following example RTL defines a local parameter `w` and a parameter `y`.

```
module topdes (z, i, a, b);
    localparam w = 8;
    parameter y = 8;
    defparam u1.W = w;
    output [w-1:0] z;
    input [w-1:0] i;
    output [y-1:0] b;
    input [y-1:0] a;
    subdes u1 (.Z(z), .I(i));
    subdes #(y) u2 (.Z(b), .I(a));
endmodule

module subdes (Z,I);
    parameter W = 8;
    parameter LS = 1;
    localparam LSB = 1;
    output [W-LSB:0] Z;
    input [W-LSB:0] I;
    assign Z = I + 1;
endmodule // subdes
```

Attribute Reference for Encounter RTL Compiler

Analysis—Subdesign Attributes

Using the following script the value of parameter `y` is changed from 8 to 4.

```
set_attr library tutorial.lbr
read_hdl -v2001 {subdes.v topdes.v}
elaborate -param {{y 4}} topdes
```

The following command requests the parameter information for subdesign `subdes_W4`.

```
rc:/> get_att hdl_parameters [find / -subdesign subdes_W4]
{W 4 0 8} {LS 1 0 1} {LSB 1 0 1}
```

- Parameter `W` of instance `u2` now has the value 4, because the first parameter `W` was overwritten by the current value of `y` which was set to 4 through the `elaborate -parameters` command.
- Parameter `LS` keeps the default value of 1.
- (Local) parameter `LSB` keeps the default value of 1.

Related Information

Affected by this command: [elaborate](#)

Related attribute: (design) [hdl_parameters](#) on page 937

hdl_pipeline_comp

`hdl_pipeline_comp {false | true}`

Default: `false`

Read-only [subdesign](#) attribute. Specifies whether the module represented by the particular subdesign object is a pipelined ChipWare component that needs to be retimed.

hdl_vdp_list

`hdl_vdp_list string`

Read-only [subdesign](#) attribute. Returns the list of RTL Compiler-supported datapath function primitives which have been instantiated in this subdesign.

Example

Consider the following RTL:

```
module top(in1, out1, out2, out3);
    input signed [7:0] in1;
    output [7:0] out1;
    parameter bits = 4;
    output [3:0] out2;
    output [3:0] out3;

    assign out1 = $round(in1, bits);
    sub s1(in1, out2, out3);
endmodule //top

module sub(a, b, c);
    input signed [7:0] a;
    output [3:0] b;
    output [3:0] c;

    assign b = $lead0(a);
    assign c = $lead1(a);
endmodule //sub
```

After reading in the RTL and elaborating the design, you can check the datapath extension functions instantiated in the design using the following command:

```
rc:/> get_attr hdl_vdp_list /des*/top/subdesigns/sub
lead0 lead1
```

Related Information

Affected by this command: [elaborate](#)

Related attribute: (design) [hdl_vdp_list](#) on page 938

instances

instances string

Read-only [subdesign](#) attribute. Lists the instances that refer to (instantiate) this subdesign.

language

language string

Read-only [subdesign](#) attribute. Specifies whether the subdesign is a Verilog module or VHDL entity.

Related Information

Related command: [read_hdl](#)

Related attribute: (design) [language](#) on page 946

library_name

library_name string

Read-only [subdesign](#) attribute. Returns the name of the Verilog or VHDL library in which the module or entity definition is stored. This name corresponds to the name specified with the -lib option of the `read_hdl` command. If this option was not specified, the name defaults to default.

Related Information

Related command: [read_hdl](#)

Related attribute: (design) [library_name](#) on page 947

logic_abstract

logic_abstract {false | true}

Read-only [subdesign](#) attribute. Specifies whether the subdesign is inferred as a logic abstract from an empty module, an entity without an architecture, or an entity whose architecture is empty in the input design description.

- If the attribute value is `true`, the subdesign is treated as an unresolved reference in the design.
- If the attribute value is `false`, the subdesign is considered a normal subdesign in the design hierarchy and can be optimized by the generic optimization engine.

Note: This attribute is supported in the RTL and structural flows.

Related Information

Affected by this command: [elaborate](#)

Related attribute: (design) [logic_abstract](#) on page 939

logical_hier

```
logical_hier {true | false}
```

Default: true

Read-write subdesign attribute. Indicates if the subdesign is a logical hierarchy object, that is, it corresponds to a user-defined entity (in VHDL) or user-defined module (in Verilog).

net_area

```
net_area string
```

Read-only subdesign attribute. Returns the net area of the subdesign. The area is specified in terms of the units specified in the library. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report area](#)

Related attribute: (design) [net_area](#) on page 942

physical_cell_area

```
physical_cell_area {no_value | float}
```

Read-only subdesign attribute. Computes the cell area of the subdesign using the LEF area values. The area is specified in terms of the units specified in the library.

Related Information

Related command: [report area](#)

Related attribute: (design) [physical_cell_area](#) on page 942

pin_count

pin_count integer

Read-only subdesign attribute. Returns the number of logical pins in the subdesign.

Related Information

Related attributes:	(design) pin_count on page 942
	(gcell) pin_count on page 436
	(instance) pin_count on page 952

wireload

wireload string

Read-only subdesign attribute. Returns the current wire-load model for the subdesign.

Related Information

Affected by this attribute:	force_wireload on page 543
Related attribute:	(design) wireload on page 946

Subport Attributes

Contain information about subports in the specified design. A subport is a single bit connection point within a hierarchical instance, that is a module or entity that has been instantiated.

- To set a subport attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subport hier_instance/name]
```

- To get a subport attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subport hier_instance/name]
```

Note: You may need to specify more than one (hierarchical) instance to serve as the search path that will uniquely identify the relevant subport.

bus

bus string

Read-only subport attribute. Returns the full path name of the bus object to which the subport belongs.

Related Information

Related attributes: (port) bus on page 1003

causes_ideal_net

causes_ideal_net {true | false}

Read-only subport attribute. Indicates whether the specified subport is causing its net to become an ideal net.

RTL Compiler implicitly sets an ideal net if some subport is

- A pin of a sequential element that is an asynchronous set or reset. You can check for this attribute by using the `ls -long -attribute` command on the pin or libcell.
- A pin of a sequential element that is a clock pin. You can check for this attribute by using the `ls -long -attribute` command on the pin or libcell.
- Any input pin of a sequential element that has no setup arc. That is, any sequential element that may be asynchronous.

Example

The following example queries whether the `in1[1]` input subport is causing its net to become ideal:

```
rc:/> get_attribute causes_ideal_net \
    {/designs/MOD69/instances_hier/inst1/supports_in/in1[1]}
false
```

Related Information

Related attributes	(pin) causes_ideal_net on page 963
	(port) causes_ideal_net on page 1004
	time_recovery_arcs on page 535

direction

```
direction {in | out| inout}
```

Read-only subport attribute. Returns the direction of the subport.

Related Information

Related attributes:	(pin) direction on page 968
	(port bus) direction on page 1029
	(port) direction on page 1006
	(subport bus) direction on page 1044

drivers

```
drivers {constant | pin | port | subport}
```

Read-only subport attribute. Returns the drivers for the subport.

Related Information

Related attributes:	(net) drivers on page 1000
	(pgpin) drivers on page 994
	(pin) drivers on page 968
	(port) drivers on page 1006

loads

```
loads { pin | port | subport}
```

Read-only subport attribute. Returns the hierarchical path to the pins or ports loading this subport.

Related Information

Related attributes:	(net) loads on page 1001 (pin) loads on page 973 (pgpin) loads on page 1001 (port) loads on page 1011
---------------------	--

net

```
net string
```

Read-only subport attribute. Returns the name of the net connected to the subport.

Related Information

Related attributes:	(pin) net on page 974 (pgpin) net on page 995 (port) net on page 1012
---------------------	---

pin_capacitance

```
pin_capacitance {no_value | float float}
```

Read-only subport attribute. Returns the rise and fall capacitances for the subport in femtofarads. Resolution is 1/10. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes:	(constant) pin_capacitance on page 958 (pin) pin_capacitance on page 975 (port) pin_capacitance on page 1013
---------------------	--

unique_versions

`unique_versions string`

Read-only subport attribute. Returns a list of the unique uses of a subport from a collection of subports. An object can have multiple object versions in the design. Use this attribute to get all the objects it represents.

Related Information

Related command: [report timing](#)

Related attributes: (constant) [unique_versions](#) on page 959

(instance) [unique_versions](#) on page 957

(net) [unique_versions](#) on page 1002

(pin) [unique_versions](#) on page 992

(port) [unique_versions](#) on page 1026

wire_capacitance

`wire_capacitance {no_value | float}`

Read-only subport attribute. Returns the net capacitance for the subport in femtofarads. Resolution is 1/10. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes: (constant) [wire_capacitance](#) on page 959

(pin) [wire_capacitance](#) on page 992

(port) [wire_capacitance](#) on page 1027

wire_length

wire_length {no_value | *float*}

Read-only subport attribute. Returns the wire length of the net connected to the subport.

Related Information

Related attributes:	(constant) wire_length on page 960
	(pin) wire_length on page 993
	(port) wire_length on page 1027

wire_resistance

wire_resistance {no_value | *float*}

Read-only subport attribute. Returns the net resistance for the subport in kilohms. Resolution is 1/1000. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related attributes:	(constant) wire_resistance on page 960
	(pin) wire_resistance on page 993
	(port) wire_resistance on page 1028

Support Bus Attributes

Contain information about support busses in the specified design. A support bus is a bussed connection point within a hierarchical instance, that is a module or entity that has been instantiated. These attributes are read-only attributes, so you cannot set their values.

- To get a `support_bus` attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subport_bus hier_instance/name]
```

Note: You may need to specify more than one (hierarchical) instance to serve as the search path that will uniquely identify the relevant support bus.

bits

bits *string*

Read-only subport bus attribute. Returns a list of individual bits that constitute the subport bus. This list includes the full pathnames of each individual subport object.

Related Information

Related attributes: (pin bus) [bits](#) on page 997
(port bus) [bits](#) on page 1029

direction

direction {in | out| inout}

Read-only `support_bus` attribute. Returns the direction of the support bus.

Related Information

Related attributes:	(pin) direction on page 968
	(port) direction on page 1006
	(port bus) direction on page 1029
	(subport) direction on page 1040

order

`order integer`

Read-only support bus attribute. Returns the order of the bus in the design port declaration list, the value of the leftmost bit being the first. The order value of the first bus is 0.

Related Information

Related attribute: (port bus) order on page 1030

Timing Bin Attributes

Contain information about timing bin objects in the specified design. These objects are created with the `create_timing_bin` command.

- To get a `timing_bin` attribute value, type

```
get_attribute attribute_name [find /des*/design -timing_bin name]
```

Note: These attributes are located at

```
/designs/design/analysis/timing_bin_name and  
/designs/design/analysis/timing_bin_name/sub_bins/sub_bin_name
```

is_sub_bin

```
is_sub_bin {false | true}
```

Read-only `timing_bin` attribute. Specifies whether this timing bin is a sub-bin, that is derived from a parent timing bin. This attribute is `true` for a sub-bin and `false` for a parent bin.

Related Information

Set by this command: [create_timing_bin](#)

Related attributes: [path_count](#) on page 1046
[root](#) on page 1047

path_count

```
path_count integer
```

Read-only `timing_bin` attribute. Returns the number of paths in the timing bin.

Example

```
rc:>/get_attribute path_count [find /des*/dtmf_recv_core -timing_bin  
m_clk_worst10]  
10
```

Related Information

Set by this command: [create_timing_bin](#)

Related attributes: [is_sub_bin](#) on page 1046
[root](#) on page 1047

root

`root string`

Read-only [timing_bin](#) attribute. Returns the name of the parent timing bin, which was specified using the `-parent` option of the `create_timing_bin` when the sub -bin was created.

Note: This attribute applies to sub bins only. For parent timing bins, this attribute had no value

Related Information

Set by this command: [create_timing_bin](#)

Related attributes: [is_sub_bin](#) on page 1046

[path_count](#) on page 1046

Timing Path Attributes

Contain information about the timing paths contained in either a parent timing bin, or sub bins located within a given parent. These timing paths are created with the `create_timing_bin` command.

- To get a `timing_path` attribute value, type

```
get_attribute attribute_name [find /des*/design -timing_path name]
```

For parent timing bins, these attributes are located at

```
/designs/design/analysis/parent_bin_name/paths/path and
```

For sub-bins, these attributes are located at

```
/designs/design/analysis/parent_bin_name/sub_bins/sub_bin_name/paths/path
```

Timing paths in a given sub-bin can only be derived from the parent bin within which the sub-bin is contained..

bin

`bin string`

Read-only `timing_path` attribute. Returns the name of the parent timing bin or sub-bin within which this path is contained.

Related Information

Set by this command: [create_timing_bin](#)

end_point

`end_point string`

Read-only `timing_path` attribute. Returns the end point of this timing path.

Related Information

Set by this command: [create_timing_bin](#)

exceptions

`exceptions string`

Read-only timing_path attribute. Returns a list of all timing exceptions associated with the specified timing path.

Example

The following command returns the list of exceptions associated with path `p_32` of parent bin `alpha`.

```
rc:/> get_attribute exceptions alpha/p_32
/designs/dtmf_recv_core/modes/func_mode/exceptions/path_delays/del_1
/designs/dtmf_recv_core/modes/func_mode/exceptions/path_groups/m_clk
```

Related Information

Set by this command: [create_timing_bin](#)

Affected by these commands: [multi_cycle](#)
[path_adjust](#)
[path_delay](#)
[path_disable](#)
[path_group](#)

mode

`mode string`

Read-only timing_path attribute. Returns the mode associated with the specified timing path.

Example

The following command returns the mode for path `p_32` of parent bin `alpha`.

```
rc:/> get_attribute mode alpha/p_32
/designs/dtmf_recv_core/modes/func_mode
```

Related Information

Set by this command: [create_timing_bin](#)

slack

`slack string`

Read-only [timing_path](#) attribute. Returns the slack of this timing path.

Related Information

Set by this command: [create_timing_bin](#)

startpoint

`startpoint string`

Read-only [timing_path](#) attribute. Returns the starting point of this timing path.

Related Information

Set by this command: [create_timing_bin](#)

Clock Attributes

Contain information about clock objects in the specified design.

- To get a `clock` attribute value, type

```
get_attribute attribute_name [find /des*/design -clock clock]
```

clock_sense_negative

```
clock_sense_negative {pin | port}...
```

Read-only `clock` attribute. Returns the pins that propagate the negative sense of this clock.

Example

Assume that the SDC file you read in contains the following command:

```
set_clock_sense -negative [get_pins {x1/Y}] -clock [get_clocks {CLK1}]
```

To verify on which pins the negative sense of clock CLK1 is propagated, specify the following command:

```
rc:/> get_attr clock_sense_negative [find / -clock CLK1]
/designs/top/instances_comb/x1/pins_out/Y
```

Related Information

Set by this command:

[dc::set_clock_sense -negative](#)

Related attributes:

(pin) [clock sense negative](#) on page 964

(clock) [clock sense positive](#) on page 1052

(pin) [clock sense positive](#) on page 965

(clock) [clock sense stop propagation](#) on page 1052

(pin) [clock sense stop propagation](#) on page 965

(pin) [propagated clocks](#) on page 975

clock_sense_positive

```
clock_sense_positive {pin | port}...
```

Read-only clock attribute. Returns the pins that propagate the positive sense of this clock.

Example

Assume that the SDC file you read in contains the following command:

```
set_clock_sense -positive [get_pins {x1/Y}] -clock [get_clocks {CLK1}]
```

To verify on which pins the positive sense of clock CLK1 is propagated, specify the following command:

```
rc:/> get_attr clock_sense_positive [find / -clock CLK1]  
/designs/top/instances_comb/x1/pins_out/Y
```

Related Information

Set by this command:

[dc::set_clock_sense -positive](#)

Related attributes:

(clock) [clock_sense_negative](#) on page 1051

(pin) [clock_sense_negative](#) on page 964

(pin) [clock_sense_positive](#) on page 965

(clock) [clock_sense_stop_propagation](#) on page 1052

(pin) [clock_sense_stop_propagation](#) on page 965

(pin) [propagated_clocks](#) on page 975

clock_sense_stop_propagation

```
clock_sense_stop_propagation pins
```

Read-only clock attribute. Returns the pins that stop propagation of the clock.

Example

Assume that the SDC file you read in contains the following command:

```
set_clock_sense -stop_propagation [get_pins {x1/Y}] -clock [get_clocks {CLK1}]
```

To verify on which pins clock CLK1 stops propagating, specify the following command:

```
rc:/> get_attr clock_sense_stop_propagation [find / -clock CLK1]  
/designs/top/instances_comb/x1/pins_out/Y
```

Related Information

Set by this command: [dc::set_clock_sense -stop_propagation](#)

Related attributes: (clock) [clock sense negative](#) on page 1051

(pin) [clock sense negative](#) on page 964

(clock) [clock sense positive](#) on page 1052

(pin) [clock sense positive](#) on page 965

(pin) [clock sense stop propagation](#) on page 965

(pin) [propagated clocks](#) on page 975

divide_fall

`divide_fall integer`

Read-only [clock](#) attribute. Returns the value specified using the `-divide_fall` option of the `define_clock` command.

Related Information

Set by this command: [define_clock](#)

Related attributes: (test_clock) [divide_fall](#) on page 1305

(test_signal) [divide_fall](#) on page 1314
[fall](#) on page 1056

divide_period

`divide_period integer`

Read-only [clock](#) attribute. Returns the value specified using the `-divide_period` option of the `define_clock` command.

Related Information

Set by this command: [define_clock](#)

Related attributes: (test_clock) [divide_period](#) on page 1305

(test_signal) [divide_period](#) on page 1315
[period](#) on page 1056

divide_rise

`divide_rise integer`

Read-only `clock` attribute. Returns the value specified using the `-divide_rise` option of the `define_clock` command.

Related Information

Set by this command:

[define_clock](#)

Related attributes:

(test_clock) [divide_rise](#) on page 1305

(test_signal) [divide_rise](#) on page 1315

[rise](#) on page 1056

divide_waveform

`divide_waveform integer_list`

Read-only `clock` attribute. Returns a list of integers that is used together with the values of the `waveform` attribute. The values of the `waveform` attribute are used as nominators, while the corresponding values of the `divide_waveform` attribute are used as denominators to determine edge changes during the clock period.

Example

In the example below a new clock is generated using the edges of a source clock. Since the generated clock refers to six edges, the difference between 2 edges in the generated clock is 1/5 the of the period of the generated clock. So edge 2 has nominator 1 and denominator 5. Since the last edge of any generated clock corresponds to the end of the period, it does not need to be included in the list because its nominator and denominator are always equal.

```
rc:/> dc::create_clock -period 30 -name clk ports_in/clk
rc:/> dc::create_generated_clock -name genclk -source ports_in/clk \
-edges {1 2 4 5 6} flop/CK
rc:/> get_attr divide_waveform [find /designs/top -clock genclk]
1 5 5 5
rc:/> get_attr waveform [find /designs/top -clock genclk]
0 1 3 4
```

Related Information

Set by this command:

[create_generated_clock](#)

Related attribute:

[waveform](#) on page 1057

exceptions

`exceptions string`

Read-only `clock` attribute. Returns a list of all the timing exceptions that were applied to the specified clock.

Example

The following example uses the `path_disable` command to unconstrain the path that is launched by the clock held in the `clock` variable. The `exceptions` attribute shows that there is only one timing exception associated with this same clock.

```
rc:/> path_disable -from $clock  
/designs/add/timing/exceptions/path_disables/dis_5  
rc:/> get_attribute exceptions $clock  
/designs/add/timing/exceptions/path_disables/dis_5
```

Related Information

Affected by these commands:

[path_adjust](#)
[path_delay](#)
[path_disable](#)
[path_group](#)
[multi_cycle](#)

Related attributes:

(clock) [exceptions](#) on page 1055
(clock_domain) [exceptions](#) on page 1058
(cost_group) [exceptions](#) on page 1059
(external_delay) [exceptions](#) on page 1070
(instance) [exceptions](#) on page 948
(pin) [exceptions](#) on page 969
(port) [exceptions](#) on page 1007

fall

fall integer

Read-only [clock](#) attribute. Returns the value specified using the **-fall** option of the **define_clock** command.

Related Information

Set by this command: [define_clock](#)

Related attributes: [divide_fall](#) on page 1053

(test_clock) [fall](#) on page 1306

(test_signal) [fall](#) on page 1315

period

period integer

Read-only [clock](#) attribute. Returns the value specified using the **-period** option of the **define_clock** command.

To derive the real clock period (specified in picoseconds), you can divide the value of the **period** attribute by the value of the **divide_period** attribute.

Related Information

Set by this command: [define_clock](#)

Related attributes: [divide_period](#) on page 1053

(test_clock) [period](#) on page 1307

(test_signal) [period](#) on page 1318

rise

rise integer

Read-only [clock](#) attribute. Returns the value specified using the **-rise** option of the **define_clock** command.

Related Information

Set by this command: [define_clock](#)

Related attributes: [divide_rise](#) on page 1054

(test_clock) [rise](#) on page 1307

(test_signal) [rise](#) on page 1320

waveform

`waveform integer_list`

Read-only [clock](#) attribute. Returns a list of integers that together with the values of the divide_waveform attribute. The values of the waveform attribute are used as nominators, while the corresponding values of the divide-waveform attribute are used as denominators to determine edge changes in the clock period.

Example

In the example below a new clock is generated using the edges of a source clock. Since the generated clock refers to six edges, the difference between 2 edges in the generated clock is 1/5 the of the period of the generated clock. So edge 2 has nominator 1 and denominator 5. Since the last edge of any generated clock corresponds to the end of the period, it does not need to be included in the list because its nominator and denominator are always equal.

```
rc:/> dc::create_clock -period 30 -name clk ports_in/clk
rc:/> dc::create_generated_clock -name genclk -source ports_in/clk \
-edges {1 2 4 5 6} flop/CK
rc:/> get_attr divide_waveform [find /designs/top -clock genclk]
1 5 5 5
rc:/> get_attr waveform [find /designs/top -clock genclk]
0 1 3 4
```

Related Information

Set by this command: [create_generated_clock](#)

Related attribute: [divide_waveform](#) on page 1054

Clock Domain Attributes

Contain information about a clock domain in the specified design. A clock domain refers to clocks that are grouped together because they are synchronous to each other, letting you perform timing analysis between these clocks.

- To get a `clock_domain` attribute value, type

```
get_attribute attribute_name [find /des*/design -clock_domain clock_domain]
```

Note: These attributes are located at `/designs/design/timing/clock_domains`

exceptions

exceptions *string*

Read-only `clock_domain` attribute. Returns a list of all the timing exceptions that were applied to the specified clock domain.

Related Information

Affected by these commands:	<u>path_adjust</u>
	<u>path_delay</u>
	<u>path_disable</u>
	<u>path_group</u>
	multi_cycle

Related attributes:	(clock) exceptions on page 1055
	(cost_group) exceptions on page 1059
	(external_delay) exceptions on page 1070
	(instance) exceptions on page 948
	(pin) exceptions on page 969
	(port) exceptions on page 1007

Cost Group Attributes

Contain information about a group of timing paths in the specified design that have a single timing optimization objective.

- To get a `cost_group` attribute value, type

```
get_attribute attribute_name [find /des*/design -cost_group group]
```

Note: These attributes are located at `/designs/design/timing/cost_groups`

exceptions

`exceptions string`

Read-only `cost_group` attribute. Returns a list of all the timing exceptions that were applied to the specified cost group.

Related Information

Affected by these commands: [path_adjust](#)
[path_delay](#)
[path_disable](#)
[path_group](#)
[multi_cycle](#)

Related attributes: [\(clock\) exceptions](#) on page 1055
[\(clock_domain\) exceptions](#) on page 1058
[\(external_delay\) exceptions](#) on page 1070
[\(instance\) exceptions](#) on page 948
[\(pin\) exceptions](#) on page 969
[\(port\) exceptions](#) on page 1007

slack

```
slack {no_value | float}
```

Read-only cost_group attribute. Returns the slack of the cost group in picoseconds. The resolution is 1. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Related command: [report timing](#)

Related attributes: (design) [slack](#) on page 944

(instance) [slack](#) on page 953

(pin) [slack](#) on page 981

(port) [slack](#) on page 1018

slack_by_mode

```
slack_by_mode {{mode_name_1 delay_value} [{mode_name_2 delay_value}]...}
```

Read-only cost_group attribute. Returns a Tcl list of lists in picoseconds the worst slack of the cost group for each timing mode.

Example

The following example returns the worst slack for the default cost group for modes a and b:

```
rc:/> get_attribute slack_by_mode [find /-cost_group default]
designs/top/modes/b -166.2{/designs/top/modes/a -43.9}
```

Related Information

[Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#) for detailed information.

Affects these commands: [report clocks](#)

[report qor](#)

[report timing](#)

[write encounter](#)

[write sdc](#)

Attribute Reference for Encounter RTL Compiler

Analysis—Cost Group Attributes

[write_script](#)

Related commands: [create_mode](#)

[read_sdc](#)

Related attributes: [disabled_arcs_by_mode](#) on page 561

(pin) [external_delays_by_mode](#) on page 970

(port) [external_delays_by_mode](#) on page 1008

(design) [latch_borrow_by_mode](#) on page 548

(instance) [latch_borrow_by_mode](#) on page 565

(design) [latch_max_borrow_by_mode](#) on page 551

(instance) [latch_max_borrow_by_mode](#) on page 568

(pin) [propagated_clocks_by_mode](#) on page 977

(port) [propagated_clocks_by_mode](#) on page 1016

(design) [slack_by_mode](#) on page 944

(pin) [slack_by_mode](#) on page 981

(port) [slack_by_mode](#) on page 1019

(pin) [timing_case_computed_value_by_mode](#) on page 986

(port) [timing_case_computed_value_by_mode](#) on page 1022

(instance) [timing_case_disabled_arcs_by_mode](#) on page 955

(pin) [timing_case_logic_value_by_mode](#) on page 609

(port) [timing_case_logic_value_by_mode](#) on page 660

Attribute Reference for Encounter RTL Compiler

Analysis—Cost Group Attributes

tns

`tns string`

Read-only cost_group attribute. Returns the sum (or total) of all worst negative slacks of all endpoints in the cost group in picoseconds. If the worst slack of a cost group is positive, a value of 0 will be reported. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Example

Assume a cost group has two endpoints, A and B, whose worst slack is -10 and +10 respectively. In this case, the `tns` value of the cost group returns -10, because the slack value of +10 is treated as zero slack.

Related Information

Related command: [report timing](#)

Related attribute: (design) [tns](#) on page 945

Exception Attributes

Contain information about the timing exceptions in the specified design. 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`.

- To get an exception attribute value, type

```
get_attribute attribute_name [find /des*/design/timing -exception name]
```

adjust_value

```
adjust_value {no_value | float}
```

Read-only exception attribute. Returns the delay constraint value of a `path_adjust` exception. The `no_value` value applies to all exceptions other than the `path_adjust` exception.

Related Information

Set by this command: [path_adjust](#)

cost_group

```
cost_group string
```

Read-only exception attribute. Returns the cost group to which a `path_group` exception belongs.

Related Information

Set by this command: [define_cost_group](#)

delay_value

```
delay_value {no_value | float}
```

Read-only exception attribute. Returns the delay constraint for a `path_delay` exception. The `no_value` value applies to all exceptions other than the `path_delay` exception.

Related Information

Set by this command: [path_delay](#)

domain

domain *clock_domain*

Read-only exception attribute. Returns the list of clock domains that this exception is restricted to.

Related Information

Affected by these commands: [path_adjust](#)
 [path_delay](#)
 [path_disable](#)
 [path_group](#)
 [multi_cycle](#)

exception_type

exception_type {multi_cycle | path_delay | path_disable | path_adjust | path_group}

Read-only exception attribute. Returns multi_cycle, path_delay, path_disable, path_adjust, or path_group, depending on the command used to create the exception.

Related Information

Constraint Commands in the *Command Reference for Encounter RTL Compiler*

Set by one of these commands: [path_adjust](#)
 [path_delay](#)
 [path_disable](#)
 [path_group](#)
 [multi_cycle](#)

from_points

`from_points string`

Read-only exception attribute. Returns the objects specified with the `-from` option of a timing exception command.

As the design is optimized, some of these points may have their names changed (for example, if a new hierarchy is introduced or if a hierarchy is flattened). This attribute is dynamically updated during such changes.

Related Information

Set by one of these commands: [path_adjust](#)
[path_delay](#)
[path_disable](#)
[path_group](#)
[multi_cycle](#)
[specify_paths](#)

lenient

`lenient {true | false}`

Read-only exception attribute. Indicates if the exception was created with the `-lenient` option.

Related Information

Set by one of these commands: [path_adjust](#)
[path_delay](#)
[path_disable](#)
[path_group](#)
[multi_cycle](#)
[specify_paths](#)

paths

`paths string`

Read-only exception attribute. Returns the Tcl command that specifies the paths to which the exception is being applied.

Related Information

Set by this command: [specify_paths](#)

precluded_path_adjusts

`precluded_path_adjusts string`

Read-only exception attribute. Returns a list of `path_adjust` objects that can be ignored if this `path_adjust` exception is satisfied. By default, `path_adjust` exceptions accumulate and their effects are added together. You do not need to set this attribute—it is normally only set by the `derive_environment` command.

Note: This attribute is only valid on `path_adjust` exceptions.

Related Information

Set by this command: [derive_environment](#)

priority

`priority integer`

Read-only exception attribute. Determines the priority of the timing exception in absence of a user-priority setting. If you read in native RTL Compiler constraints, RTL Compiler gives the highest priority to the exception that was last entered. If you read in SDC constraints, RTL Compiler sets the priority on the timing exceptions to match the behavior of PrimeTime.

Related Information

Affected by this attribute: [user_priority on page 677](#)

shift_capture

```
shift_capture {no_value | integer}
```

Read-only exception attribute. Returns the capture clock shift value of a `multi_cycle` exception. The `no_value` value applies to all exceptions other than the `multi_cycle` exception.

Related Information

Set by this command: [multi_cycle](#)

shift_launch

```
shift_launch {no_value | integer}
```

Read-only exception attribute. Returns the launch clock shift value of a `multi_cycle` exception. The `no_value` value applies to all exceptions other than the `multi_cycle` exception.

Related Information

Set by this command: [multi_cycle](#)

through_points

```
through_points string
```

Read-only exception attribute. Returns the objects using the `-through` option of a timing exception command. As the design is optimized, some of these points may have their names changed (for example, if a new hierarchy is introduced or if a hierarchy is flattened). This attribute is dynamically updated during such changes.

Related Information

Set by one of these commands: [path_adjust](#)
[path_delay](#)
[path_disable](#)
[path_group](#)
[multi_cycle](#)
[specify_paths](#)

to_points

`to_points string`

Read-only exception attribute. Returns the objects specified with the `-to` option of a timing exception command. As the design is optimized, some of these points may have their names changed (for example, if a new hierarchy is introduced or if a hierarchy is flattened). This attribute is dynamically updated during such changes.

Related Information

Set by one of these commands:

- [path_adjust](#)
- [path_delay](#)
- [path_disable](#)
- [path_group](#)
- [multi_cycle](#)
- [specify_paths](#)

External Delay Attributes

Contains information about the external delay constraints defined in the specified design. These attributes are read-only attributes, so you cannot set their values.

- To get an `external_delay` attribute value, type

```
get_attribute attribute_name [find /des*/design/timing -external_delay name]
```

clock

`clock string`

Read-only `external_delay` attribute. Returns the clock object for the `external_delay` constraint.

Related Information

Set by this command: [define_clock](#)

Affects this command: [external_delay](#)

clock_rise

`clock_rise {true | false}`

Read-only `external_delay` attribute. Returns `true` if the `external_delay` is relative to a rising clock edge and `false` if it is relative to a falling edge.

Related Information

Set by this command: [external_delay](#)

delay

`delay integer_list`

Read-only `external_delay` attribute. Returns the minimum and maximum rise and fall delay values of the `external_delay` constraint.

Note: RTL Compiler does not use the minimum values, but storing the minimum values allows RTL Compiler to write out the SDC constraints correctly.

Related Information

Set by this command: [external_delay](#)

exceptions

`exceptions string`

Read-only [external_delay](#) attribute. Returns a list of all the timing constraints that were applied to the specified external delay.

Related Information

Affected by these commands: [path_adjust](#)
[path_delay](#)
[path_disable](#)
[path_group](#)
[multi_cycle](#)

Related attributes: (clock) [exceptions](#) on page 1055
(clock_domain) [exceptions](#) on page 1058
(cost_group) [exceptions](#) on page 1059
(instance) [exceptions](#) on page 948
(pin) [exceptions](#) on page 969
(port) [exceptions](#) on page 1007

external_delay_pins

`external_delay_pins string`

Read-only [external_delay](#) attribute. Returns the Tcl list of pins and ports for the `external_delay` constraint.

Related Information

Set by this command: [external_delay](#)

Related attributes: [external_delays](#) on page 1008

input_delay

`input_delay {true | false}`

Read-only external_delay attribute. Returns `true` if the `external_delay` is an input delay and `false` if it is an output delay.

Related Information

Set by this command: [external_delay](#)

level_sensitive

`level_sensitive {true | false}`

Read-only external_delay attribute. Returns `true` if the external register is level-sensitive and `false` if the external register is edge-triggered.

Related Information

Set by this command: [external_delay](#)

Attribute Reference for Encounter RTL Compiler

Analysis—External Delay Attributes

Design For Test

Root Attributes

- [dft boundary cell module prefix](#) on page 1093
- [dft clock waveform divide fall](#) on page 1094
- [dft clock waveform divide period](#) on page 1094
- [dft clock waveform divide rise](#) on page 1095
- [dft clock waveform fall](#) on page 1095
- [dft clock waveform period](#) on page 1096
- [dft clock waveform rise](#) on page 1096
- [dft fence slow speed domains](#) on page 1097
- [dft generate et no testpoint file](#) on page 1097
- [dft identify internal test clocks](#) on page 1098
- [dft identify test signals](#) on page 1099
- [dft identify top level test clocks](#) on page 1099
- [dft identify xsource violations from timing models](#) on page 1099
- [dft include controllable pins in abstract model](#) on page 1100
- [dft include test signal outputs in abstract model](#) on page 1104
- [dft jtag module name](#) on page 1106
- [dft opcg block input to flop paths](#) on page 1107
- [dft opcg domain blocking](#) on page 1107
- [dft prefix](#) on page 1108
- [dft propagate test signals from hookup pins only](#) on page 1108

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [dft_report_empty_test_clocks](#) on page 1109
- [dft_rtl_insertion](#) on page 1109
- [dft_scan_style](#) on page 1110
- [dft_scanbit_waveform_analysis](#) on page 1110
- [dft_shift_register_identification_mode](#) on page 1111
- [dft_true_time_flow](#) on page 1112
- [et_license_options](#) on page 1113
- [pmbist_enable_multiple_views](#) on page 1114
- [unmap_scan_flops](#) on page 1114

Design Attributes

- [boundary_type](#) on page 1116
- [dft_boundary_scan_exists](#) on page 1117
- [dft_clock_edge_for_head_of_scan_chains](#) on page 1117
- [dft_clock_edge_for_tail_of_scan_chains](#) on page 1118
- [dft_connect_scan_data_pins_during_mapping](#) on page 1119
- [dft_connect_shift_enable_during_mapping](#) on page 1120
- [dft_dont_scan](#) on page 1121
- [dft_jtag_macro_exists](#) on page 1121
- [dft_lockup_element_type](#) on page 1122
- [dft_lockup_element_type_for_tail_of_scan_chains](#) on page 1123
- [dft_max_length_of_scan_chains](#) on page 1124
- [dft_min_number_of_scan_chains](#) on page 1124
- [dft_mix_clock_edges_in_scan_chains](#) on page 1125
- [dft_scan_map_mode](#) on page 1126
- [dft_scan_output_preference](#) on page 1127
- [dft_tap_tck_period](#) on page 1128

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [iocell_enable](#) on page 1129
- [iocell_input](#) on page 1129
- [iocell_output](#) on page 1130
- [mbist_enable_shared_library_domain_set](#) on page 1130

Instance Attributes

- [dft_abstract_dont_scan](#) on page 1132
- [dft_custom_se](#) on page 1132
- [dft_dont_scan](#) on page 1133
- [dft_exempt_from_system_clock_check](#) on page 1134
- [dft_force_blackbox_for_atpg](#) on page 1134
- [dft_is_blackbox_for_atpg](#) on page 1134
- [dft_mapped](#) on page 1135
- [dft_part_of_segment](#) on page 1135
- [dft_scan_chain](#) on page 1135
- [dft_status](#) on page 1136
- [dft_test_clock](#) on page 1137
- [dft_test_clock_edge](#) on page 1137
- [dft_test_clock_source](#) on page 1138
- [dftViolation](#) on page 1138
- [mbist_instruction_set](#) on page 1139
- [pmbist_instruction_set](#) on page 1140

Pin Attributes

- [dft_constant_value](#) on page 1141
- [dft_controllable](#) on page 1142
- [dft_driven_by_clock](#) on page 1142

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [dft_opcg_domain_clock_pin](#) on page 1143
- [dft_opcg_domain_fanout_pin](#) on page 1143
- [dft_opcg_domain_launch_clock](#) on page 1144
- [dft_opcg_domain_se_input_pin](#) on page 1145
- [dft_opcg_domain_unfenced_capture](#) on page 1145
- [user_differential_negative_pin](#) on page 1146
- [user_from_core_data](#) on page 1146
- [user_from_core_enable](#) on page 1147
- [user_test_receiver_acmode](#) on page 1148
- [user_test_receiver_data_output](#) on page 1148
- [user_test_receiver_init_clock](#) on page 1148
- [user_test_receiver_init_data](#) on page 1148
- [user_to_core_data](#) on page 1149
- [user_to_core_enable](#) on page 1149
- [wrapper_control](#) on page 1150
- [wrapper_segment](#) on page 1150

Net Attributes

- [dft_constant_value](#) on page 1152

Subdesign Attributes

- [dft_dont_scan](#) on page 1153

Support Attributes

- [dft_driven_by_clock](#) on page 1155
- [dft_opcg_domain_clock_pin](#) on page 1155
- [dft_opcg_domain_fanout_pin](#) on page 1156
- [dft_opcg_domain_launch_clock](#) on page 1156

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [dft_opcg_domain_se_input_pin](#) on page 1157
- [dft_opcg_domain_unfenced_capture](#) on page 1157
- [wrapper_control](#) on page 1166
- [wrapper_segment](#) on page 1159

Port Attributes

- [dft_driven_by_clock](#) on page 1160
- [dft_enable hookup_pin](#) on page 1160
- [dft_enable hookup_polarity](#) on page 1161
- [dft_opcg_asserted_domain](#) on page 1162
- [dft_opcg_domain_clock_pin](#) on page 1162
- [dft_opcg_domain_fanout_pin](#) on page 1163
- [dft_opcg_domain_launch_clock](#) on page 1163
- [dft_opcg_domain_se_input_pin](#) on page 1164
- [dft_opcg_domain_unfenced_capture](#) on page 1165
- [dft_sdi_output hookup_pin](#) on page 1166
- [dft_sdo_input hookup_pin](#) on page 1166
- [wrapper_control](#) on page 1166
- [wrapper_segment](#) on page 1167

Boundary-Scan Segment Attributes

- [acdcsel_11496](#) on page 1168
- [acpclk_11496](#) on page 1168
- [acpsen_11496](#) on page 1169
- [acptrenbl_11496](#) on page 1169
- [acpulse_11496](#) on page 1170
- [bsdl](#) on page 1170

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [capturedr](#) on page 1187
- [clockdr](#) on page 1187
- [differential_pairs](#) on page 1172
- [highz](#) on page 1189
- [instance](#) on page 1189
- [modea](#) on page 1173
- [modeb](#) on page 1174
- [modec](#) on page 1174
- [shiftdr](#) on page 1192
- [tdi](#) on page 1192
- [tdo](#) on page 1193
- [updatedr](#) on page 1194

JTAG Instruction Attributes

- [capture](#) on page 1177
- [length](#) on page 1177
- [opcode](#) on page 1178
- [private](#) on page 1178
- [register](#) on page 1178
- [register_capturedr](#) on page 1179
- [register_clockdr](#) on page 1179
- [register_decode](#) on page 1180
- [register_reset](#) on page 1180
- [register_reset_polarity](#)
- [register_runitidle](#) on page 1181
- [register_shiftdr](#) on page 1181
- [register_shiftdr_polarity](#) on page 1181

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [register_tck](#) on page 1182
- [register_tdi](#) on page 1182
- [register_tdo](#) on page 1182
- [register_updatedr](#) on page 1183
- [tap_decode](#) on page 1183
- [tap_tdi](#) on page 1184
- [tap_tdo](#) on page 1184

JTAG Instruction Register Attributes

- [capture](#) on page 1185
- [length](#) on page 1185

JTAG Macro Attributes

- [boundary_tdo](#) on page 1186
- [bsr_clockdr](#) on page 1186
- [bsr_shiftdr](#) on page 1186
- [bsr_updatedr](#) on page 1187
- [capturedr](#) on page 1187
- [clockdr](#) on page 1187
- [dot6_acdcsel](#) on page 1188
- [dot6_acemode](#) on page 1188
- [dot6_acpulse](#) on page 1188
- [dot6_preset_clock](#) on page 1188
- [dot6_trcell_enable](#) on page 1189
- [exitdr](#) on page 1189
- [highz](#) on page 1189
- [instance](#) on page 1189

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [mode_a](#) on page 1190
- [mode_b](#) on page 1190
- [mode_c](#) on page 1190
- [por](#) on page 1191
- [reset](#) on page 1191
- [runidle](#) on page 1191
- [shiftdr](#) on page 1192
- [tck](#) on page 1192
- [tdi](#) on page 1192
- [tdo](#) on page 1193
- [tdo_enable](#) on page 1193
- [tms](#) on page 1193
- [trst](#) on page 1194
- [updatedr](#) on page 1194
- [user_defined_macro](#) on page 1194

JTAG Port Attributes

- [aio_pin](#) on page 1195
- [bcell_location](#) on page 1195
- [bcell_required](#) on page 1196
- [bcell_segment](#) on page 1196
- [bcell_type](#) on page 1197
- [bdy_enable](#) on page 1197
- [bdy_in](#) on page 1198
- [bdy_out](#) on page 1198
- [bsr_dummy_after](#) on page 1198
- [bsr_dummy_before](#) on page 1199

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [cell](#) on page 1200
- [comp_enable](#) on page 1201
- [custom_bcell](#) on page 1201
- [differential](#) on page 1201
- [index](#) on page 1202
- [pin](#) on page 1202
- [pinmap](#) on page 1202
- [sys_enable](#) on page 1203
- [sys_use](#) on page 1203
- [test_use](#) on page 1204
- [tr_bdy_in](#) on page 1204
- [tr_cell](#) on page 1204
- [trcell_acmode](#) on page 1205
- [trcell_clock](#) on page 1205
- [trcell_enable](#) on page 1205
- [type](#) on page 1206

TAP Port Attributes

- [dft hookup_pin](#) on page 1207
- [dft hookup_polarity](#) on page 1207
- [pin](#) on page 1207
- [type](#) on page 1208

DFT Configuration Mode Attributes

- [current_mode](#) on page 1209
- [decoded_pin](#) on page 1209
- [jtag_instruction](#) on page 1210

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [mode_enable_high](#) on page 1210
- [mode_enable_low](#) on page 1211
- [type](#) on page 1212
- [usage](#) on page 1212
- [user_defined](#) on page 1213

Memory Data Bit Structure Attributes

- [column_order](#) on page 1214
- [partial_row_order](#) on page 1214
- [row_order](#) on page 1215

Memory Libcell Attributes

- [address_limit](#) on page 1216
- [data_order](#) on page 1216
- [memory_libcell](#) on page 1217
- [read_delay](#) on page 1217
- [wrapper](#) on page 1217

Memory Libpin Action Attributes

- [value](#) on page 1219

Memory Libpin Alias Attributes

- [base_port_name](#) on page 1220

Memory Spare Column Attributes

- [address_bits](#) on page 1221
- [banks](#) on page 1221
- [data_bits](#) on page 1222

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [enable](#) on page 1222
- [srclk](#) on page 1222
- [srsi](#) on page 1223
- [srso](#) on page 1223

Memory Spare Column Map Address Attributes

- [address_logical_value](#) on page 1224
- [address_port](#) on page 1224
- [address_port_value](#) on page 1225

Memory Spare Column Map Data Attributes

- [data_logical_value](#) on page 1226
- [data_port](#) on page 1226
- [data_port_value](#) on page 1227

Memory Spare Row Attributes

- [address_bits](#) on page 1228
- [banks](#) on page 1228
- [data_bits](#) on page 1229
- [enable](#) on page 1229
- [srclk](#) on page 1229
- [srsi](#) on page 1230
- [srso](#) on page 1230

Memory Spare Row Map Address Attributes

- [address_logical_value](#) on page 1231
- [address_port](#) on page 1231
- [address_port_value](#) on page 1232

Attribute Reference for Encounter RTL Compiler

Design For Test—List

Write Mask Bit Attributes

- [masked_bits](#) on page 1233

Direct Access Function Attributes

- [active](#) on page 1234
- [clocked_by_mtclk](#) on page 1234
- [source](#) on page 1235

Programmable Direct Access Function Attributes

- [active](#) on page 1236
- [source](#) on page 1236

MBIST Clock Attributes

- [dft hookup_pin](#) on page 1237
- [dft hookup_polarity](#) on page 1237
- [hookup_period](#) on page 1238
- [internal](#) on page 1238
- [is_jtag_tck](#) on page 1238
- [period](#) on page 1239
- [sources](#) on page 1239

Domain Macro Parameters Attributes

- [counter_length](#) on page 1240
- [max_num_pulses](#) on page 1240
- [target_period](#) on page 1241
- [trigger_delay](#) on page 1241

OPCG Domain Attributes

- [divide_by](#) on page 1242
- [domain_macro_parameter](#) on page 1242
- [location](#) on page 1242
- [min_domain_period](#) on page 1243
- [opcg_trigger](#) on page 1243
- [osc_source](#) on page 1243

OPCG Mode Attributes

- [jtag_controlled](#) on page 1245
- [mode_init](#) on page 1245

Osc Source Reference Attributes

- [osc_source_period](#) on page 1251
- [ref_clk_period](#) on page 1251

OPCG Trigger Attributes

- [active](#) on page 1246
- [inside_inst](#) on page 1246
- [osc_source](#) on page 1247
- [pin](#) on page 1247

Osc Source Attributes

- [max_input_period](#) on page 1248
- [max_output_period](#) on page 1248
- [min_input_period](#) on page 1249
- [min_output_period](#) on page 1249
- [pin](#) on page 1250

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [ref_clock_pin](#) on page 1250

Actual Scan Chain Attributes

- [analyzed](#) on page 1252
- [compressed](#) on page 1252
- [connected_shift_enable](#) on page 1253
- [dft hookup pin_sdi](#) on page 1253
- [dft hookup pin_sdo](#) on page 1254
- [domain](#) on page 1254
- [edge](#) on page 1254
- [elements](#) on page 1255
- [has_opcg_segments](#) on page 1255
- [head_lockup](#) on page 1256
- [mode_name](#) on page 1256
- [other_clocks](#) on page 1256
- [power_domain](#) on page 1257
- [reg_count](#) on page 1257
- [scan_clock_a](#) on page 1257
- [scan_clock_b](#) on page 1258
- [scan_in](#) on page 1258
- [scan_out](#) on page 1259
- [shared_output](#) on page 1260
- [shared_select](#) on page 1260
- [shift_enable](#) on page 1260
- [terminal_lockup](#) on page 1261

Actual Scan Segment Attributes

- [active](#) on page 1262
- [clock](#) on page 1263
- [clock_edge](#) on page 1263
- [connected_scan_clock_a](#) on page 1264
- [connected_scan_clock_b](#) on page 1264
- [connected_shift_enable](#) on page 1264
- [core_wrapper](#) on page 1265
- [dft_tail_test_clock](#) on page 1265
- [dft_tail_test_clock_edge](#) on page 1266
- [dft_test_clock](#) on page 1266
- [dft_test_clock_edge](#) on page 1267
- [elements](#) on page 1268
- [instance](#) on page 1268
- [other_clocks](#) on page 1268
- [reg_count](#) on page 1270
- [reorderable](#) on page 1270
- [scan_clock_a](#) on page 1271
- [scan_clock_b](#) on page 1271
- [scan_in](#) on page 1271
- [scan_out](#) on page 1272
- [shift_enable](#) on page 1273
- [skew_safe](#) on page 1273
- [tail_clock](#) on page 1274
- [tail_clock_edge](#) on page 1274
- [type](#) on page 1275

Attribute Reference for Encounter RTL Compiler

Design For Test—List

Violations Attributes

- [description](#) on page 1276
- [endpoints](#) on page 1278
- [file_name](#) on page 1279
- [fixed](#) on page 1279
- [id](#) on page 1280
- [line_number](#) on page 1281
- [reg_count](#) on page 1281
- [registers](#) on page 1282
- [root_node](#) on page 1283
- [segments](#) on page 1283
- [tristate_net_drivers](#) on page 1284
- [tristate_net_load](#) on page 1284
- [type](#) on page 1284

Scan Chain Attributes

- [body](#) on page 1285
- [complete](#) on page 1285
- [dft hookup pin sdi](#) on page 1286
- [dft hookup pin sdo](#) on page 1286
- [domain](#) on page 1286
- [edge](#) on page 1287
- [head](#) on page 1287
- [max_length](#) on page 1287
- [scan_clock_a](#) on page 1288
- [scan_clock_b](#) on page 1288
- [scan_in](#) on page 1289

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [scan_out](#) on page 1289
- [shared_output](#) on page 1290
- [shared_select](#) on page 1290
- [shift_enable](#) on page 1290
- [tail](#) on page 1291
- [terminal_lockup](#) on page 1291

Scan Segment Attributes

- [active](#) on page 1292
- [clock](#) on page 1292
- [clock_edge](#) on page 1293
- [connected_scan_clock_a](#) on page 1293
- [connected_scan_clock_b](#) on page 1294
- [connected_shift_enable](#) on page 1294
- [core_wrapper](#) on page 1294
- [core_wrapper_ports](#) on page 1295
- [core_wrapper_type](#) on page 1295
- [core_wrapper_usage](#) on page 1296
- [dft_dont_scan](#) on page 1296
- [dft_status](#) on page 1297
- [dft_tail_test_clock](#) on page 1298
- [dft_tail_test_clock_edge](#) on page 1298
- [dft_test_clock](#) on page 1299
- [dft_test_clock_edge](#) on page 1299
- [dftViolation](#) on page 1300
- [elements](#) on page 1300
- [instance](#) on page 1301

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [other_clocks](#) on page 1301
- [power_domain](#) on page 1301
- [reg_count](#) on page 1302
- [reorderable](#) on page 1303
- [scan_clock_a](#) on page 1303
- [scan_clock_b](#) on page 1303
- [scan_in](#) on page 1304
- [scan_out](#) on page 1304
- [shift_enable](#) on page 1305
- [skew_safe](#) on page 1305
- [tail_clock](#) on page 1305
- [tail_clock_edge](#) on page 1306
- [test_modes](#) on page 1306
- [type](#) on page 1307
- [user_defined_segment](#) on page 1308

Test Bus Port Attributes

- [dft hookup_pin](#) on page 1309
- [dft hookup_polarity](#) on page 1309
- [function](#) on page 1310
- [index](#) on page 1310
- [pin](#) on page 1310

Test Clock Attributes

- [at_speed](#) on page 1311
- [atpg_use](#) on page 1312
- [blocking_se](#) on page 1312

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [controllable](#) on page 1313
- [dft hookup pin](#) on page 1313
- [dft hookup polarity](#) on page 1313
- [dft mask clk](#) on page 1314
- [dft misr clock](#) on page 1314
- [divide fall](#) on page 1315
- [divide period](#) on page 1315
- [divide rise](#) on page 1315
- [fall](#) on page 1316
- [off state](#) on page 1316
- [period](#) on page 1317
- [rise](#) on page 1317
- [root source pins](#) on page 1317
- [root source polarity](#) on page 1318
- [sources](#) on page 1318
- [user defined signal](#) on page 1319

Test Signal Attributes

- [active](#) on page 1320
- [atpg use](#) on page 1321
- [dedicated pin](#) on page 1322
- [default shift enable](#) on page 1322
- [dft compression signal](#) on page 1323
- [dft hookup pin](#) on page 1323
- [dft hookup polarity](#) on page 1324
- [divide fall](#) on page 1324
- [divide period](#) on page 1325

Attribute Reference for Encounter RTL Compiler

Design For Test—List

- [divide_rise](#) on page 1325
- [fall](#) on page 1325
- [has_fanout](#) on page 1326
- [ideal](#) on page 1326
- [lec_value](#) on page 1327
- [master_signal](#) on page 1327
- [period](#) on page 1328
- [pin](#) on page 1328
- [pmbist_use](#) on page 1329
- [rise](#) on page 1330
- [scan_shift](#) on page 1330
- [type](#) on page 1331
- [user_defined_signal](#) on page 1331

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

dft_apply_sdc_constraints

```
dft_apply_sdc_constraints {false | true}
```

Default: false

Read-write `root` attribute. When set to `true`, SDC constraints for DFT constructs are applied during boundary-scan insertion, scan connection and xor compression without masking. The SDC constraints are applied when the relevant commands to insert those DFT constructs are run. The SDC constraints can be written out using the `write_sdc` command.

Related Information

[Generating SDC Constraints in DFT in Design for Test in Encounter RTL Compiler](#)

Affects these commands:

[compress_scan_chains](#)
[connect_scan_chains](#)
[insert_dft_boundary_scan](#)
[insert_dft_compression_logic](#)
[insert_dft_jtag_macro](#)

dft_boundary_cell_module_prefix

```
dft_boundary_cell_module_prefix string
```

Default: " "

Read-write `root` attribute. Specifies the prefix for the module names of boundary-scan cells added during the insertion of the boundary-scan logic.

Related Information

[Inserting Boundary-Scan Logic in Design for Test in Encounter RTL Compiler](#)

Affects this command: [insert_dft_boundary_scan](#)

dft_clock_waveform_divide_fall

`dft_clock_waveform_divide_fall integer`

Default: 100

Read-write root attribute. Used with the `dft_clock_waveform_fall` attribute to specify 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 `dft_clock_waveform_fall` by `dft_clock_waveform_divide_fall`.

The attribute value is applied by the `define_dft test_clock` command in lieu of specifying the command option `divide_fall` and is used by all auto-generated test-clocks defined having run the `check_dft_rules` command.

Related Information

[Defining Test Clock Waveforms in Design for Test in Encounter RTL Compiler](#)

Affects this command: [define_dft test_clock](#)

dft_clock_waveform_divide_period

`dft_clock_waveform_divide_period integer`

Default: 1

Read-write root attribute. Used with the `dft_clock_waveform_divide` attribute to specify the clock period interval. The clock period is specified in picoseconds and is derived by dividing `dft_clock_waveform_period` by `dft_clock_waveform_divide_period`. The attribute value is applied by the `define_dft test_clock` command in lieu of specifying the command option `divide_period` and is used by all auto-generated test-clocks defined having run the `check_dft_rules` command.

Related Information

[Defining Test Clock Waveforms in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [define_dft test_clock](#)

dft_clock_waveform_divide_rise

`dft_clock_waveform_divide_rise integer`

Default: 100

Read-write root attribute. Used with the `dft_clock_waveform_rise` attribute to specify 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 `dft_clock_waveform_rise` by `dft_clock_waveform_divide_rise`. The attribute value is applied by the `define_dft test_clock` command in lieu of specifying the command option `-divide_rise` and is used by all auto-generated test-clocks defined having run the `check_dft_rules` command.

Related Information

[Defining Test Clock Waveforms in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [define_dft test_clock](#)

dft_clock_waveform_fall

`dft_clock_waveform_fall integer`

Default: 90

Read-write root attribute. Used with the `dft_clock_waveform_divide_fall` to specify 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 `dft_clock_waveform_fall` by `dft_clock_waveform_divide_fall`. The attribute value is applied by the `define_dft test_clock` command in lieu of specifying the command option `-fall` and is used by all auto-generated test-clocks defined having run the `check_dft_rules` command.

Related Information

[Defining Test Clock Waveforms in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [define_dft test_clock](#)

dft_clock_waveform_period

`dft_clock_waveform_period integer`

Default: 50000

Read-write root attribute. Used with the `dft_clock_waveform_divide_period` attribute to specify the clock period interval. The clock period is specified in picoseconds and is derived by dividing `dft_clock_waveform_period` by `dft_clock_waveform_divide_period`. The attribute value is applied by the `define_dft test_clock` command in lieu of specifying the command option `-period` and is used by all auto-generated test-clocks defined having run the `check_dft_rules` command.

Related Information

[Defining Test Clock Waveforms in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [define_dft test_clock](#)

dft_clock_waveform_rise

`dft_clock_waveform_rise integer`

Default: 50

Read-write root attribute. Used with the `dft_clock_waveform_divide_rise` attribute to specify 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 `dft_clock_waveform_rise` by `dft_clock_waveform_divide_rise`. The attribute value is applied by the `define_dft test_clock` command in lieu of specifying the command option `-rise` and is used by all auto-generated test-clocks defined having run the `check_dft_rules` command.

Attribute Reference for Encounter RTL Compiler

Design For Test—Root Attributes

Related Information

[Defining Test Clock Waveforms in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [define_dft test_clock](#)

dft_fence_slow_speed_domains

`dft_fence_slow_speed_domains {false | true}`

Default: false

Read-write [root](#) attribute. Controls addition of blocking logic for slow clock domains.

Related Information

Affects this command: [connect_scan_chains](#)

dft_generate_et_no_testpoint_file

`dft_generate_et_no_testpoint_file {false | true}`

Default: false

Read-write [root](#) attribute. Controls the automatic generation of the `design.noTpfile` by the `write_et_dfa` and `write_et_rrfa` commands. This file contains a list of subdesigns, instances, nets, and pins that have been constrained, thus preventing test point insertion on them.

This file is passed to Encounter Test via the `notpfile` keyword of the `analyze_random_resistance` and `analyze_deterministic_faults` commands.

Related Information

Affects these commands: [insert_dft_rrfa_test_points](#)
[write_et_dfa](#)
[write_et_rrfa](#)

dft_identify_internal_test_clocks

```
dft_identify_internal_test_clocks {false | true | no_cgic_hier}
```

Default: false

Read-write root attribute. Indicates whether the DFT rule checker must identify the output pins of multi-input combinational gates and the clock output pins of the clock-gating instances in the clock path as separate test clocks in the same DFT clock domain as its root-level test clock.

This attribute can have the following values:

false	Prevents that output pins of multi-input combinational gates and the clock output pins of the clock-gating instances in the clock path are identified as separate test clocks.
no_cgic_hier	Prevents that the clock output pins of the clock-gating instances are identified as separate test clocks. However, output pins of multi-input combinational gates in the clock path are identified.
true	Requests to identify both the output pins of multi-input combinational gates and the clock output pins of the clock-gating instances in the clock path as separate test clocks.

The internal test clocks and their associated combinational logic gates are listed in the report generated by the `report dft_setup` command.

Note: The internal test clocks are only identified for those multi-input combinational cells that are mapped to a technology component.

Related Information

[Identifying Output Pins of Multi-Input Combinational Gates as Separate Test Clocks in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [check_dft_rules](#)

dft_identify_test_signals

`dft_identify_test_signals {true | false}`

Default: true

Read-write root attribute. Indicates whether the DFT rule checker can automatically assign a test mode signal for each top-level pin that is traceable from the async set or reset pin of a flip-flop. To prevent auto identification, set this attribute to `false`.

Related Information

Affects this command: [check_dft_rules](#)

dft_identify_top_level_test_clocks

`dft_identify_top_level_test_clocks {true | false}`

Default: true

Read-write root attribute. Indicates whether the DFT rule checker can automatically assign a *test clock* for each top-level clock pin that is traceable from the clock pin of a flip-flop, and a corresponding *test-clock domain*. To prevent auto identification, set this attribute to `false`.

Related Information

Affects this command: [check_dft_rules](#)

dft_identify_xsource_violations_from_timing_models

`dft_identify_xsource_violations_from_timing_models {false | true}`

Default: false

Read-write root attribute. Indicates whether the DFT rule checker can automatically identify x-source violations from output pins of timing models. To enable automatic identification, set this attribute to `true`.

Related Information

Affects this command: [check_dft_rules -advanced](#)

dft_include_controllable_pins_in_abstract_model

`dft_include_controllable_pins_in_abstract_model {allmodes | none | test_setup}`

Default: allmodes

Read-write root attribute. Controls how design output pins which are controllable from design input pins under tied-constant propagation setup and test setup propagation are written to the native or CTL abstract models. This attribute can have the following values:

allmodes	Defines all design-level output signals which can be traced back to a design-level input signal for both tied-constant propagation setup run separately from test setup propagation as DFT controllable by that input signal. If the output signal cannot be traced back to an input signal and if the output pin is a constant, the test signal is defined on it.
	Note: Test setup propagation is also referred to as <code>test_setup</code> .
none	Prevents writing out any additional information for design-level output signals which can be traced back to a design-level input signal for all propagation analysis modes.
test_setup	Defines all design-level output signals which can be traced back to a design-level input signal under test setup propagation as DFT controllable by that input signal. If the output signal cannot be traced back to an input signal because the output pin is a constant under <code>test_setup</code> , a test signal is defined for the output pin.

Note: The

`dft_include_test_signal_outputs_in_abstract_model` root attribute (default `true`) controls the writing of output signals to the abstract models for outputs signals

- whose values are constant under `test_setup` propagation
- assigned to tied constant values

Note: Test setup propagation refers to the propagation of tied-constants, test-mode and shift-enable signals.

Examples

In the following example, the tool can trace back from the `teout` pin to the `tein` pin when tied-constant propagation setup is run separately from test setup propagation. The example shows how the output of the native abstract model changes with the setting of the attribute.

For the following examples,

- Consider this netlist snippet:

```
module test (....)
  ...
  assign out = in;
  assign constantOutOne = 1'b1;
  assign constantOutZero = 1'b0;
  assign teout = tein;
  ...
endmodule
```

- Assume input signal `tein` is defined as an active high test signal:

```
define_dft test_mode -active high tein
```

- Assume the `dft_include_test_signal_outputs_in_abstract_model` attribute is set to `true`.

The following commands show how the output of the native and CTL abstract models change with the settings of the attribute:

```
rc-suspend:/> set_attr dft_include_controllable_pins_in_abstract_model allmodes /
Setting attribute of root '/': 'dft_include_controllable_pins_in_abstract_model' =allmodes
rc-suspend:/> write_dft_abstract_model
scan style is muxed_scan
#Writing out dft_controllable
proc ::dft_controllable_proc { inst } {
    set_attribute dft_controllable "$inst/pins_in/in non_inverting" $inst/pins_out/out
    set_attribute dft_controllable "$inst/pins_in/tein non_inverting" $inst/pins_out/teout
    define_dft test_mode -active low $inst/pins_out/constantOutZero
    define_dft test_mode -active high $inst/pins_out/constantOutOne }

set_dft_functions_from_proc -subdesign sub -function dft_controllable_proc

# writing abstract model for 1 scan chain

define_dft abstract_segment -module sub \
    -name sub_AutoChain_1 \
    -sdi DFT_sdi_1 -sdo DFT_sdo_1 \
    -shift_enable_port shift -active high \
    -test_mode_port tein -test_mode_active high \
    -test_mode_port reset -test_mode_active low \
    -clock_port clk -rise \
    -tail_clock_port clk -tail_edge_rise \
    -length 4

rc-suspend:/> write_dft_abstract_model -ctl
scan style is muxed_scan
Writing out abstraction information in CTL format
...
Environment "sub" {
    CTLMode Internal_scan {
```

Attribute Reference for Encounter RTL Compiler

Design For Test—Root Attributes

```
TestMode InternalTest;
DomainReferences {
    SignalGroups Internal_scan ;
    ScanStructures Internal_scan ;
    Procedures Internal_scan ;
    MacroDefs Internal_scan ;
}
...
...
"out" {
    IsConnected Out {
        Signal "in";
    }
}
"teout" {
    IsConnected Out {
        Signal "tein";
    }
}
"constantOutZero" {
    DataType Constant {
        ActiveState ExpectLow;
    }
}
"constantOutOne" {
    DataType Constant {
        ActiveState ExpectHigh;
    }
}
}
}
}

rc-suspend:/> set_attr dft_include_controllable_pins_in_abstract_model none /
Setting attribute of root '/': 'dft_include_controllable_pins_in_abstract_model' = none
rc-suspend:/> write_dft_abstract_model
scan style is muxed_scan

# writing abstract model for 1 scan chain

define_dft abstract_segment -module sub \
-name sub_AutoChain_1 \
-sdi DFT_sdi_1 -sdo DFT_sdo_1 \
-shift_enable_port shift -active high \
-test_mode_port tein -test_mode_active high \
-test_mode_port reset -test_mode_active low \
-clock_port clk -rise \
-tail_clock_port clk -tail_edge_rise \
-length 4

rc-suspend:/> set_attr dft_include_controllable_pins_in_abstract_model test_setup /
Setting attribute of root '/': 'dft_include_controllable_pins_in_abstract_model' =
test_setup
rc-suspend:/> write_dft_abstract_model
scan style is muxed_scan
#Writing out dft_controllable
proc ::dft_controllable_proc { inst } {
    define_dft test_mode -active high $inst/pins_out/teout
    define_dft test_mode -active low $inst/pins_out/constantOutZero
    define_dft test_mode -active high $inst/pins_out/constantOutOne
    set_attribute dft_controllable "$inst/pins_in/in non_inverting" $inst/pins_out/out
}

set_dft_functions_from_proc -subdesign sub -function dft_controllable_proc

# writing abstract model for 1 scan chain
```

Attribute Reference for Encounter RTL Compiler

Design For Test—Root Attributes

```
define_dft abstract_segment -module sub \
-name sub_AutoChain_1 \
-sdi DFT_sdi_1 -sdo DFT_sdo_1 \
-shift_enable_port shift -active high \
-test_mode_port tein -test_mode_active high \
-test_mode_port reset -test_mode_active low \
-clock_port clk -rise \
-tail_clock_port clk -tail_edge_rise \
-length 4

rc-suspend:/> write_dft_abstract_model -ctl
scan style is muxed_scan
Writing out abstraction information in CTL format

...
Environment "sub" {
    CTLMode Internal_scan {
        TestMode InternalTest;
        DomainReferences {
            SignalGroups Internal_scan ;
            ScanStructures Internal_scan ;
            Procedures Internal_scan ;
            MacroDefs Internal_scan ;
        }
    }
...
...
    "out" {
        IsConnected Out {
            Signal "in";
        }
    }
    "teout" {
        DataType Constant {
            ActiveState ExpectHigh;
        }
    }
    "constantOutZero" {
        DataType Constant {
            ActiveState ExpectLow;
        }
    }
    "constantOutOne" {
        DataType Constant {
            ActiveState ExpectHigh;
        }
    }
}
}
```

Related Information

[Creating a Scan Abstract Model in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [write_dft_abstract_model](#)

Affected by this attribute: [dft_include test signal outputs in abstract_model](#) on page 1104

dft_include_test_signal_outputs_in_abstract_model

```
dft_include_test_signal_outputs_in_abstract_model {true | false}
```

Default: true

Read-write root attribute. Enables writing output signals whose values are constant in test setup, and output signals assigned to tied constant values, as test mode signals in the native abstract model and as constants in the CTL abstract models.

Examples

For the following examples,

- Consider this netlist snippet:

```
module test (....)
...
    assign out = in;
    assign constantOutOne = 1'b1;
    assign constantOutZero = 1'b0;
    assign teout = tein;
...
endmodule
```

- Assume input signal `tein` is defined as an active high test signal:

```
define_dft test_mode -active high tein
```

- Assume the `dft_include_controllable_pins_in_abstract_model` attribute is set to `allmodes`.

The following commands show how the output of the native and CTL abstract models change with the settings of the attribute:

```
rc-suspend:/> set_attr dft_include_test_signal_outputs_in_abstract_model true /
Setting attribute of root '/': 'dft_include_test_signal_outputs_in_abstract_model' = true
rc-suspend:/> write_dft_abstract_model
scan style is muxed_scan
#Writing out dft_controllable
proc ::dft_controllable_proc { inst } {
    set_attribute dft_controllable "$inst/pins_in/in non_inverting" $inst/pins_out/out
    set_attribute dft_controllable "$inst/pins_in/tein non_inverting" $inst/pins_out/teout
    define_dft test_mode -active low $inst/pins_out/constantOutZero
    define_dft test_mode -active high $inst/pins_out/constantOutOne
}

set_dft_functions_from_proc -subdesign sub -function dft_controllable_proc
# writing abstract model for 1 scan chain

define_dft abstract_segment -module sub \
    -name sub_AutoChain_1 \
    -sdi DFT_sdi_1 -sdo DFT_sdo_1 \
    -shift_enable_port shift -active high \
    -test_mode_port tein -test_mode_active high \
    -test_mode_port reset -test_mode_active low \
    -clock_port clk -rise \
```

Attribute Reference for Encounter RTL Compiler

Design For Test—Root Attributes

```
-tail_clock_port clk -tail_edge_rise \
-length 4

rc-suspend:/> write_dft_abstract_model -ctl
scan style is muxed_scan
Writing out abstraction information in CTL format
...
Environment "sub" {
    CTLMode Internal_scan {
        TestMode InternalTest;
        DomainReferences {
            SignalGroups Internal_scan ;
            ScanStructures Internal_scan ;
            Procedures Internal_scan ;
            MacroDefs Internal_scan ;
        }
    }
...
...
    "out" {
        IsConnected Out {
            Signal "in";
        }
    }
    "teout" {
        IsConnected Out {
            Signal "tein";
        }
    }
    "constantOutZero" {
        DataType Constant {
            ActiveState ExpectLow;
        }
    }
    "constantOutOne" {
        DataType Constant {
            ActiveState ExpectHigh;
        }
    }
}
}

rc-suspend:/> set_attr dft_include_test_signal_outputs_in_abstract_model false /
Setting attribute of root '/': 'dft_include_test_signal_outputs_in_abstract_model' = false

rc-suspend:/> write_dft_abstract_model
scan style is muxed_scan
#Writing out dft_controllable
proc ::dft_controllable_proc { inst } {
    set_attribute dft_controllable "$inst/pins_in/in non_inverting" $inst/pins_out/out
    set_attribute dft_controllable "$inst/pins_in/tein non_inverting" $inst/pins_out/teout
}
set_dft_functions_from_proc -subdesign sub -function dft_controllable_proc
# writing abstract model for 1 scan chain

define_dft abstract_segment -module sub \
    -name sub_AutoChain_1 \
    -sdi DFT_sdi_1 -sdo DFT_sdo_1 \
    -shift_enable_port shift -active high \
    -test_mode_port tein -test_mode_active high \
    -test_mode_port reset -test_mode_active low \
    -clock_port clk -rise \
    -tail_clock_port clk -tail_edge_rise \
    -length 4
```

Attribute Reference for Encounter RTL Compiler

Design For Test—Root Attributes

```
rc-suspend:/> write_dft_abstract_model -ctl
scan style is muxed_scan
Writing out abstraction information in CTL format
...
Environment "sub" {
    CTLMode Internal_scan {
        TestMode InternalTest;
        DomainReferences {
            SignalGroups Internal_scan ;
            ScanStructures Internal_scan ;
            Procedures Internal_scan ;
            MacroDefs Internal_scan ;
        }
    ...
    ...
    "out" {
        IsConnected Out {
            Signal "in";
        }
    }
    "teout" {
        IsConnected Out {
            Signal "tein";
        }
    }
}
}
```

Related Information

[Creating a Scan Abstract Model in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [write dft abstract model](#)

Affects this attribute: [dft include controllable pins in abstract model on page 1100](#)

dft_jtag_module_name

`dft_jtag_module_name string`

Default: JTAG_MACRO

Read-write root attribute. Specifies the module (or subdesign) name of the JTAG macro.

Related Information

[Inserting Boundary-Scan Logic in Design for Test in Encounter RTL Compiler](#)

Affects these commands: [insert dft boundary scan](#)

[insert dft jtag macro](#)

dft_opcg_block_input_to_flop_paths

`dft_opcg_block_input_to_flop_paths {false | true}`

Default: false

Read-write root attribute. Controls whether to block input ports to the flop paths.

When you perform synthesis on a block in a bottom-up test synthesis flow, and you apply test clocks to the block that are generated by OPCG domain macros located outside the block, you must set this attribute to true. In this case, the input data is typically coming from a different clock domain. When this attribute is enabled, the tool propagates unknown clocks from the primary inputs that are not defined as test clocks.

When you process the top level in test synthesis flow with OPCG logic insertion, the input data typically comes from the tester, which is static. In that case, blocking is not required on the sink of an input port to the flop path.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Affects this command: [connect_scan_chains](#)

Affected by this attribute: [dft_opcg_asserted_domain](#) on page 1162

dft_opcg_domain_blocking

`dft_opcg_domain_blocking {false | true}`

Default: false

Read-write root attribute. Enables OPCG domain blocking.

Related Information

Affects this command: [insert_dft_opcg](#)

Related command: [define_dft_opcg_domain](#)

dft_prefix

`dft_prefix string`

Default: DFT_

Read-write `root` attribute. Specifies the prefix for instances of added control logic, and the base names for any scan-data input, scan-data output, shift-enable, `scan_clock_a` and `scan_clock_b` ports or ports for compression test signals created during test synthesis.

The base names are `prefixSDI_`, `prefixSDO_`, `prefixSEN_`, `prefixsclk_a_` or `prefixsclk_b_`, `prefixcompression_enable`, `prefixspreader`, `prefixmask_load`, `prefixmask_enable`, `prefixmask_sdi`, `prefixmask_sdo` (depending on the pin type).

Related Information

[Controlling Naming of Scan Chains in Design for Test in Encounter RTL Compiler](#).

[Controlling Naming of Scan Data Ports in Design for Test in Encounter RTL Compiler](#).

Affects these commands:

[compress_scan_chains](#)
[connect_scan_chains](#)
[define_dft_scan_chain](#)

dft_propagate_test_signals_from hookup_pins_only

`dft_propagate_test_signals_from hookup_pins_only {false | true}`

Default: false

Read-write `root` attribute. Controls constant propagation from the pin specified by a test signal's `pin` attribute. Set this attribute to `true` to disable constant propagation from the pin specified by the `pin test_signal` attribute.

Related Information

Affects this command:

[check_dft_rules](#)

Related constraints:

[define_dft_shift_enable](#)
[define_dft_test_mode](#)

dft_report_empty_test_clocks

```
dft_report_empty_test_clocks {false | true}
```

Default: false

Read-write root attribute. Determines whether the DFT rule checker and the `report dft_setup` command should report information on test clocks and their edges that do not drive any registers. By default, this information is not reported.

Related Information

Affects these commands: [check_dft_rules](#)
[report dft setup](#)

dft_rtl_insertion

```
dft_rtl_insertion { false | true }
```

Default: false

Read-write root attribute. Enables a DFT RTL insertion flow. When set to `true`, the user-supplied RTL files will be updated with the RTL constructs for the inserted JTAG macro and MBIST structures using the `write_dft_rtl_model` command.

Related Information

[Inserting Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler](#)

Affects these commands [insert_dft_jtag_macro](#)
[insert_dft_mbist](#)
[write_dft_rtl_model](#)

dft_scan_style

```
dft_scan_style {muxed_scan | clocked_lssd_scan}
```

Default: muxed_scan

Read-write root attribute. Specifies the scan style for all designs read in. This attribute can have the following values:

clocked_lssd_scan	Selects the clocked LSSD scan style
muxed_scan	Selects the muxed scan style

Set the appropriate scan style before running `check_dft_rules` since the DFT rules vary with different scan styles. If no scan style is set, the tool assumes `muxed_scan` style.

Related Information

[Setting Up for Test in Design for Test in Encounter RTL Compiler](#).

Affects these commands:	<u>check_dft_rules</u> <u>connect_scan_chains</u> <u>report_dft_registers</u> <u>synthesize</u>
-------------------------	--

dft_scanbit_waveform_analysis

```
dft_scanbit_waveform_analysis {false | true}
```

Default: false

Read-write root attribute. When set to `true`, additional analysis of the test clock waveforms is performed to determine if a non-scan flop in an analyzed preserved segment or analyzed scan chain is a scan-bit or a lockup flop.

Note: To determine whether a non-scan flop is a scan-bit or lockup flop, the registers of the scan chain segment must pass the DFT rule checks.

Attribute Reference for Encounter RTL Compiler

Design For Test—Root Attributes

Related Information

[Analyzing Chains in a Scan-Connected Netlist in Design for Test in Encounter RTL Compiler](#)

Affects these commands: [define_dft_preserved_segment -analyze](#)
 [define_dft_scan_chain -analyze](#)

dft_shift_register_identification_mode

`dft_shift_register_identification_mode {test_setup_shiftenable | test_setup
| logical_only | testmode_only | testmode_shiftenable}`

Default: test_setup_shiftenable

Read-write root attribute. Controls propagation of constants and test signals when identifying shift registers. This attribute can have the following values:

logical_only	Does not perform any constant value propagation.
test_setup	Propagates constant values and test signal values.
test_setup_shiftenable	Propagates logical constants, test-mode and shift-enable signal values.
testmode_only	Propagates test-mode signal values only.
testmode_shiftenable	Propagates test-mode and shift-enable signal values.

Related Information

[Automatically Identifying Shift Registers in Design for Test in Encounter RTL Compiler.](#)

Affects these commands: [identify_shift_register_scan_segments](#)
 [synthesize -auto_identify_shift_register,](#)

dft_true_time_flow

`dft_true_time_flow {false | true}`

Default: false

Read-write root attribute. Controls generation of the Encounter Test true time use model script which allows access to all flows supported by the true time script.

Related Information

[Writing the Scripts and Setup Files to Perform ATPG in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [write_et_atpg](#)

dft_wait_for_license

`dft_wait_for_license {false | true}`

Default: false

Read-write root attribute. Specifies that DFT commands which require an Encounter Test license, wait for a license to become available. By default, the command will fail if the required license is not available. For more information on the exact product requirements for the DFT commands, refer to [Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.](#)

Note: The time that a command can wait for a license is controlled by the value of the -wait option of the `rc` command.

et_license_options

`et_license_options string`

Read-write root attribute. Specifies a string of options you want to pass to the `et` command when an advanced DFT feature in RTL Complier invokes the Encounter Test software.

Examples of options you might want to pass to the `et` command are:

-architect	Use applicable Encounter Test Architect licenses first.
-nolicpromote	Disable automatic license promotion. Instead, wait for the first applicable license to free up before proceeding.
-showlic	Causes the application(s) invoked to print out what license is being checked out.
-truetime	Use applicable Encounter Test True Time licenses first.

For a complete list of the options, refer to *Command Line Reference* (of the Encounter Test documentation).

Example

The following command requests to use Encounter Test 64 bit with the license promotion feature disabled:

```
set_attribute et_license_options "-64 -nolicpromote" /
```

Related Information

[Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler](#).

Affects these commands:

[analyze scan compressibility](#)
[analyze testability](#)
[insert dft rrfa test points](#)
[report test power -atpg option](#)

pmbist_enable_multiple_views

```
pmbist_enable_multiple_views {false | true}
```

Default: false

Read-write root attribute. Controls whether more than one view of a memory can be specified as a target for PMBIST.

When set to true, a *logical wrapper* and either a *memory wrapper* or a memory module can be the two different views of a given memory that must be included in a common target group within the configuration file for PMBIST insertion to allow both views to be tested by PMBIST.

Related Information

[Inserting Programmable Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler](#)

Affects this command: [insert_dft pmbist](#)

unmap_scan_flops

```
unmap_scan_flops {false | not_mapped_for_dft | true}
```

Default: false

Read-write root attribute. Controls the selective unmapping and remapping of scan flip-flops during synthesis. (During synthesis optimization, a mapped instance can be unmapped to generic logic and then be remapped to a cell from one of the technology libraries.) This attribute can have the following values:

false	Prevents unmapping of scan flip-flops. However, during optimization a scan register instance can still be replaced by a different library cell to meet synthesis constraints. Using this setting, you can retain the scan chain connectivity while still allowing optimization of the scan registers.
not_mapped_for_dft	Allows unmapping and remapping of scan flip-flops that were not mapped for DFT.
true	Allows unmapping of all scan flip-flops. This implies that during optimization the flip-flops can lose their scan connectivity. Therefore, you must run the <code>connect_scan_chains</code> command to recreate any scan chains in the design.

Attribute Reference for Encounter RTL Compiler

Design For Test—Root Attributes

Related Information

[Controlling Unmapping of Scan Flip-Flops in Design for Test in Encounter RTL Compiler.](#)

Affects these commands: [synthesize](#)

Design Attributes

Contain information about the specified design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design
```

or

```
set_attribute attribute_name attribute_value [find /des* -design name]
```

- To get a design attribute value, type

```
get_attribute attribute_name design
```

boundary_type

```
boundary_type {IEEE_11491 | IEEE_11496}
```

Default: none

Read-write design attribute. Specifies either an IEEE 1149.1 or IEEE 1149.6 IEEE boundary-scan architecture.

This attribute can have the following values:

IEEE_11491	Specifies an IEEE 1149.1 JTAG_Macro and boundary-scan architecture
IEEE_11496	Specifies an IEEE 1149.6 JTAG_Macro and boundary-scan architecture.

Related Information

[Inserting Boundary-Scan Logic in Design for Test in Encounter RTL Compiler](#)

Affects these commands:

[insert_dft_boundary_scan](#)

[insert_dft_jtag_macro](#)

dft_boundary_scan_exists

```
dft_boundary_scan_exists {false | true}
```

Default: false

Read-write design attribute. Indicates whether boundary-scan logic was inserted into the design.

Related Information

[Inserting a JTAG Macro in Design for Test in Encounter RTL Compiler](#)

[Inserting Boundary-Scan Logic in Design for Test in Encounter RTL Compiler](#)

Set by this command: [insert dft boundary scan](#)

Affects this command: [insert dft boundary scan](#)

dft_clock_edge_for_head_of_scan_chains

```
dft_clock_edge_for_head_of_scan_chains {" " | leading}
```

Default: " "

Read-write design attribute. Specifies the triggering edge of the first element of the actual scan chains to build.

If the triggering edge of the first element of an actual scan chain does not match the value specified for this attribute, the tool inserts a lockup flop at the beginning of the actual scan chain. This lockup flop will be driven by the same clock that drives the first element in the chain and will be triggered by the clock edge specified with this attribute.

For RTZ clock waveforms, leading edge refers to the rising edge.

For RT1 clock waveforms, leading edge refers to the falling edge.

This attribute can have the following values:

" " Does not insert a lockup flop at the head of any actual scan chain.

leading Inserts a leading edge-triggered lockup flop at the head of all actual scan chains for which the first element is not leading edge-triggered.

Related Information

[Connecting the Scan Chains in Design for Test in Encounter RTL Compiler.](#)

Affects these commands:	<u>connect scan chains</u> <u>write dft abstract model</u>
Related attributes:	<u>dft clock edge for tail of scan chains</u> on page 1118 <u>dft lockup element type for tail of scan chains</u> on page 1123

dft_clock_edge_for_tail_of_scan_chains

`dft_clock_edge_for_tail_of_scan_chains {" " | leading | trailing}`

Default: " "

Read-write [design](#) attribute. Specifies the triggering edge of the last element of the actual scan chains to be built.

If the triggering edge of the last element of an actual scan chain does not match the value specified for this attribute, the tool inserts a lockup element (flop or latch) at the end of the actual scan chain. This lockup element will be driven by the same clock that drives the last element in the chain and will be triggered by the clock edge specified with this attribute.

For RTZ clock waveforms, `leading` refers to the rising edge and `trailing` to the falling edge.

For RT1 clock waveforms, `leading` refers to the falling edge and `trailing` to the rising edge.

This attribute can have the following values:

" "	Does not insert a lockup element at the end of an actual scan chain, unless the chain is built from a user-defined chain that is defined with the <code>-terminal_lockup</code> option.
<code>leading</code>	Inserts a leading edge-triggered lockup element at the end of all actual scan chains for which the last element is not leading edge-triggered.
<code>trailing</code>	Inserts a trailing edge-triggered lockup element at the end of all actual scan chains for which the last element is not trailing edge-triggered.

Related Information

[Connecting the Scan Chains in Design for Test in Encounter RTL Compiler.](#)

Affects these commands:	<u>connect_scan_chains</u> <u>write_dft_abstract_model</u>
Related attributes:	<u>dft_clock_edge_for_head_of_scan_chains</u> on page 1117 <u>dft_lockup_element_type_for_tail_of_scan_chains</u> on page 1123

dft_connect_scan_data_pins_during_mapping

`dft_connect_scan_data_pins_during_mapping {loopback | floating | ground}`

Default: loopback

Read-write [design](#) attribute. Starting from RTL, this attribute controls the connections of the scan-data pins during an initial synthesis run (`synthesize -to_mapped`), when mapping generic flops to their scan-equivalent flops.

Starting from a gate-level netlist, this attribute controls the connections of the scan-data pins during an incremental synthesis run (`synthesize -incremental`), when remapping non-scan flops to their scan-equivalent flops.

This attribute can have the following values:

<code>floating</code>	Leaves the scan-data input pins of the scan flops unconnected.
<code>ground</code>	Connects the scan-data input pins of the scan flops to logic 0. If available, a logic 0 net or equivalent library cell is used. Otherwise a constant ('1'b0) is used.
<code>loopback</code>	Connects a scan-data output pin of the scan flops to its own scan-data input pin, emulating the loading effect on the scan-data output without connecting the chain.

The recommended use model is to use `loopback` mode. This approach emulates a single standard load seen by the timing engine during synthesis. This load emulates the additional resistance and capacitance on the network, when the scan chains are subsequently connected in the circuit using the `connect_scan_chains` command.

Attribute Reference for Encounter RTL Compiler

Design For Test—Design Attributes

Note: Connection of the shift-enable pins is controlled by the `dft_connect_shift_enable_during_mapping` attribute.

Related Information

[Controlling Connection of Scan Data and Shift-Enable Pins in Design for Test in Encounter RTL Compiler](#).

Affects this command: [synthesize](#)

Related attribute: [dft scan map mode](#) on page 1126

dft_connect_shift_enable_during_mapping

`dft_connect_shift_enable_during_mapping {tie_off | floating}`

Default: `tie_off`

Read-write [design](#) attribute. Controls the connection of the following pins during an initial synthesis run (`synthesize -to_mapped`), when mapping generic flops to their scan-equivalent flops:

- Shift-enable pins of the scan flip-flops (when using the muxed scan style)
- Scan clock pins of the scan flip-flops (when using the clocked LSSD scan style)

This attribute can have the following values:

`floating` Shift-enable (scan clock) pins are left unconnected.

`tie_off` Shift-enable (scan clock) pins are connected to their off state (tied-to a low for active high enables, tied-to a high for active low enables).

Related Information

[Controlling Connection of Scan Data and Shift-Enable Pins in Design for Test in Encounter RTL Compiler](#).

Affects this command: [synthesize](#)

Related attribute: [dft scan map mode](#) on page 1126

dft_dont_scan

```
dft_dont_scan {inherited | true | false}
```

Default: inherited

Read-write design attribute. Controls scan replacement—for the purposes of test—of flip-flops in the specified design. This attribute can have the following values:

false Allows scan replacement of the design.

inherited Allows scan replacement of the design.

Note: At the design level, the `inherited` value has the same meaning as `false`.

true Prevents scan replacement of the design.

You must set this attribute prior to running the `check_dft_rules` command.

Related Information

[Marking Objects not to be Mapped to Scan Flip-Flops in Design for Test in Encounter RTL Compiler.](#)

Affects these commands:

[check_dft_rules](#)

[connect_scan_chains](#)

[synthesize](#)

Related attributes:

(instance) [dft_dont_scan](#) on page 1133

(scan_segment) [dft_dont_scan](#) on page 1296

(subdesign) [dft_dont_scan](#) on page 1153

dft_jtag_macro_exists

```
dft_jtag_macro_exists {true | false}
```

Default: false

Read-write design attribute. Indicates whether an existing JTAG_Macro has been acknowledged as the master TAP controller in the design, or if a new JTAG_Macro has been inserted into the design.

Related Information

[Inserting Boundary-Scan Logic in Design for Test in Encounter RTL Compiler](#)

[Inserting a JTAG Macro in Design for Test in Encounter RTL Compiler](#)

Affects these commands: [insert_dft_boundary_scan](#)
[insert_dft_jtag_macro](#)

dft_lockup_element_type

`dft_lockup_element_type {preferred_level_sensitive | preferred_edge_sensitive
| edge_sensitive | level_sensitive}`

Default: preferred_level_sensitive

Read-write [design](#) attribute. Controls the type of component to be used as lockup element when combining scan flops triggered by different clocks, or different edges of a test clock in the same chain. This attribute can have the following values:

edge_sensitive	Requests to insert an edge-sensitive component (D-flop). Note: This could cause skew problems.
level_sensitive	Requests to insert a level-sensitive component (latch). Note: This could cause skew problems.
preferred_edge_sensitive	Requests to insert lockup flops where possible, but latches when required to avoid skew concerns between the different test clocks.
preferred_level_sensitive	Requests to insert lockup latches where possible, but flops when required to avoid skew concerns between the different test clocks.

Note: This attribute applies only to the muxed scan style.

Related Information

[Controlling the Type of Lockup Elements in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [connect_scan_chains](#)
Related attribute: [dft_mix_clock_edges_in_scan_chains](#) on page 1125

dft_lockup_element_type_for_tail_of_scan_chains

`dft_lockup_element_type_for_tail_of_scan_chains {edge_sensitive | level_sensitive}`

Default: edge_sensitive

Read-write [design](#) attribute. Specifies the lockup element type to be added at the end of actual scan chains when the `dft_clock_edge_for_tail_of_scan_chains` attribute is set to a non null value. This attribute has no effect when the `dft_clock_edge_for_tail_of_scan_chains` is not set.

For actual scan chains derived from user-defined scan chains defined with the `-terminal_lockup` option, the lockup element type specified with the `-terminal_lockup` option will override the lockup element type specified with this attribute.

This attribute can have the following values:

<code>edge_sensitive</code>	Inserts a lockup flop at the end of actual scan chain during scan chain connection.
<code>level_sensitive</code>	Inserts a lockup latch at the end of actual scan chain during scan chain connection.

Related Information

[Connecting the Scan Chains in Design for Test in Encounter RTL Compiler.](#)

Affects these commands: [connect_scan_chains](#)
[write_dft_abstract_model](#)
Related attributes: [dft_clock_edge_for_head_of_scan_chains](#) on page 1117
[dft_clock_edge_for_tail_of_scan_chains](#) on page 1118

dft_max_length_of_scan_chains

`dft_max_length_of_scan_chains integer`

Default: no_value

Read-write `design` attribute. Specifies the maximum length of any scan chain. If necessary, the test synthesis engine creates additional scan chains to keep each scan chain at or below the required maximum length. By default, there is no limit to the maximum length of a scan chain, unless you specified the maximum scan chain length explicitly using the `define dft scan chain` constraint. The maximum length for a specific scan chain takes precedence over the design-specific maximum scan chain length.

Related Information

[Setting Maximum Length of Scan Chains in Design for Test in Encounter RTL Compiler](#).

Affects this command:

[connect_scan_chains](#)

dft_min_number_of_scan_chains

`dft_min_number_of_scan_chains integer`

Default: no_value

Read-write `design` attribute. Specifies the minimum number of scan chains to be created.

Related Information

[Setting Minimum Number of Scan Chains in Design for Test in Encounter RTL Compiler](#).

Affects this command:

[connect_scan_chains](#)

dft_mix_clock_edges_in_scan_chains

`dft_mix_clock_edges_in_scan_chains {false | true}`

Default: false

Read-write [design](#) attribute. Controls 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 can specify DFT domains and their assumed test clock waveforms during test mode using the `define_dft test_clock` constraint. In the absence of defining the DFT domains explicitly, the assignment of scan flops to their DFT domains is performed by running the `check_dft_rules` command. By default, all rising and all falling edge-triggered scan flops driven by the same logical clock source belong to the same DFT domain. A separate DFT domain is created for each uniquely identified logical clock source.

The test synthesis engine evaluates the test clock waveforms when deciding how to order the scan flops along the scan chain (rise-to-fall, fall-to-rise). It also evaluates where to include a data lockup element when combining scan chain segments triggered by the different active edges of the test clocks belonging to the same DFT domain. The type of lockup element inserted into the scan chain is defined using the `dft_lockup_element_type` attribute.

Set this attribute to `true` if you want a single scan chain mixing edges from the same test clock onto the same scan chain.

Set this attribute to `false` if you want a single scan chain per active clock edge per DFT domain.

Note: This attribute applies only to the muxed scan style.

Related Information

[Mixing Edges of Scan Flip-Flops in Same Scan Chain in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [connect_scan_chains](#)

Related attribute: [dft_lockup_element_type](#) on page 1122

dft_scan_map_mode

```
dft_scan_map_mode {tdrc_pass | force_all | preserve}
```

Default: tdrc_pass

Read-write design attribute. Controls the mapping of the flip-flops to their scan-equivalent flip-flops.

Note: Non-scan flip-flops marked with either a `dft_dont_scan` attribute or a `preserve` attribute, or non-scan flip-flops instantiated in blocks marked with either a `preserve` or a `dft_dont_scan` attribute, are not affected by the setting of the `dft_scan_map_mode` attribute—these flip-flops will not be mapped to scan flip-flops.

`force_all`

Controls mapping of all (non attributed) non-scan flip-flops to their scan-equivalent flip-flops. Even though these flip-flops are now mapped to scan flip-flops, only those flip-flops that pass the DFT rule checks, are connected into a scan chain during scan connection. You should only use this setting if you plan to fix the violations.

Note: Since a subsequent synthesis run will not unmap a scan flop which fails the DFT rule checks to its non-scan equivalent flop, this approach can result in a Quality of Silicon (QoS) degradation over the `tdrc_pass` approach.

`preserve`

Preserves the flip-flop type (non-scan and scan) of previously mapped flip-flops.

During synthesis scan mapping is bypassed. To prevent inferred flip-flops from being mapped to scan flip-flops for functional use, set the `use_scan_seqs_for_non_dft` root attribute to `false` before you run initial synthesis on the design.

`tdrc_pass`

Allows only flip-flops that pass the DFT rule checks to be mapped during synthesis. Use this value when

- Synthesizing from RTL.
To maximize fault coverage, fix any outstanding DFT violations using the `fix_dftViolations` command, prior to mapping (`synthesize -to_mapped`) .
- Performing an incremental optimization (`synthesize -incremental`) on a mapped netlist with the intent to replace any flip-flop which passes the DFT rule checks with scan flops.

Note: You need to run the `check_dft_rules` command to determine the DFT status of the flip-flops.

Related Information

[Controlling Mapping to Scan Flip-Flops in Design for Test in Encounter RTL Compiler](#).

Affected by this command: [check_dft_rules](#)

Affects these commands: [replace_scan](#)
[synthesize](#)

Related attributes: [dft_connect_scan_data_pins_during_mapping](#) on page 1119
[dft_connect_shift_enable_during_mapping](#) on page 1120

dft_scan_output_preference

`dft_scan_output_preference {auto | non_inverted | inverted}`

Default: auto

Read-write [design](#) attribute. Controls which scan flip-flop output pin to use for the scan-data path connection.

auto	Lets the tool choose the output pin based on the pin load or impact on the timing.
inverted	Selects the output pin with the <code>test_scan_out_inverted</code> attribute or QB pin.
non_inverted	Selects the output pin with the <code>test_scan_out</code> attribute or Q pin.

Depending on the scan flip-flop selected by the technology mapper during synthesis, a scan flop can potentially have three output pins:

- Q: a functional non-inverted output pin
- QB: a functional inverted output pin
- SO: a dedicated scan-data output pin

Attribute Reference for Encounter RTL Compiler

Design For Test—Design Attributes

Whether a scan flip-flop output pin can be used for scan-data purposes is controlled through a test attribute defined for each scan flip-flop in the technology library. The test attributes defined in Liberty format are:

`test_scan_out` Specifies an output scan pin.

`test_scan_out_inverted` Specifies an output scan pin having inverted polarity.

Related Information

[Controlling Scan Flip-Flop Output Connection in Design for Test in Encounter RTL Compiler.](#)

[Scan Cell Requirements in the Library Guide for Encounter RTL Compiler.](#)

Affects this command: [synthesize](#)

dft_tap_tck_period

`dft_tap_tck_period integer`

Default: 50000

Read-write `design` attribute. Specifies the period of the Test Clock (TCK) on the TAP Controller in picoseconds.

You can set this attribute directly, or the attribute value can be set through the `-tck_period` option of the `define_dft_tap_port` command.

The `write_bsdl` command will write out the TCK frequency in Hz in the BSSDL file. The default frequency is 20 MHz.

Related Information

[Inserting Boundary-Scan Logic in Design for Test in Encounter RTL Compiler](#)

Set by this command: [define_dft_tap_port](#)

Affects this command: [write_bsdl](#)

iocell_enable

`iocell_enable string`

Default: no_value

Read-write design attribute. Specifies the functional output enable pin of an iocell. Use this attribute to manually identify the I/O cell enable pin on I/O cells that cannot be automatically identified from the cell description in the library. You can specify a single pin name or a list of pin names to match the enable pin on the I/O cell. If a list of pin names is specified, the I/O cell will be examined for each pin name in the order that the pin name appears in the list until a match is found. To specify the search criteria for the pin, the I/O cell name may be specified along with the pin name in the `iocell_enable` string. If the enable pin on the I/O pad cell is an active low enable pin, it can be specified using the ! character before the pin name.

Related Information

[Overriding Inferred I/O Pad Cell Behavior in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [insert_dft_boundary_scan](#)

iocell_input

`iocell_input string`

Default: no_value

Read-write design attribute. Specifies the functional input pin of an iocell. Use this attribute to manually identify the I/O cell input pin on I/O cells that cannot be automatically identified from the cell description in the library. You can specify a single pin name or a list of pin names to match the input pin on the I/O cell. If a list of pin names is specified, the I/O cell will be examined for each pin name in the order that the pin name appears in the list until a match is found. To specify the search criteria for the pin, the I/O cell name may be specified along with the pin name in the `iocell_input` string.

Related Information

[Overriding Inferred I/O Pad Cell Behavior in Design for Test in Encounter RTL Compiler.](#)

Affects this command: [insert_dft_boundary_scan](#)

iocell_output

`iocell_output string`

Default: no_value

Read-write `design` attribute. Specifies the functional output pin of an iocell. Use this attribute to manually identify the I/O cell output pin on I/O cells that cannot be automatically identified from the cell description in the library. You can specify a single pin name or a list of pin names to match the output pin on the I/O cell. If a list of pin names is specified, the I/O cell will be examined for each pin name in the order that the pin name appears in the list until a match is found. To specify the search criteria for the pin, the I/O cell name may be specified along with the pin name in the `iocell_output` string.

Related Information

[Overriding Inferred I/O Pad Cell Behavior in Design for Test in Encounter RTL Compiler](#).

Affects this command: [insert_dft_boundary_scan](#)

mbist_enable_shared_library_domain_set

`mbist_enable_shared_library_domain_set {""|list_of_lists|all}`

Default: no_value

Read-write `design` attribute. Controls whether memories that have the same module name but that belong to different library domains can share the same engine. By default, memories present in different library domains cannot share an MBIST or PMBIST engine.

The value for this attribute is a list. Each element in this list is a list of library domains. The memories present in this list of library domains can share an MBIST or PMBIST engine.

Examples

- The following command prevents sharing between the memories in the timing and power library domains.

```
set_attribute mbist_enable_shared_library_domain_set \
  "[list [list timing] [list power]]" "design"
```

- The following command allows sharing between the memories in the timing and power library domains.

```
set_attribute mbist_enable_shared_library_domain_set \
  "[list [list timing power]]" "design"
```

Attribute Reference for Encounter RTL Compiler

Design For Test—Design Attributes

- The following commands both prevent sharing of MBIST or PMBIST engines among memories in all library domains.

```
set_attribute mbist_enable_shared_library_domain_set "" design  
set_attribute mbist_enable_shared_library_domain_set "[list [list timing]]" \  
design
```

- The following command allows sharing of MBIST or PMBIST engines among memories in all library domains.

```
set_attribute mbist_enable_shared_library_domain_set all design
```

Related Information

[Inserting Programmable Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler](#)

Affects these commands:

[insert_dft mbist](#)

[insert_dft pmbist](#)

Instance Attributes

Contain information about a combinational, sequential, or hierarchical instance in the specified design. Instance objects are found in the `instances_comb`, `instances_seq`, and `instances_hier` directories.

- To set an instance attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -instance name]
```

- To get an instance attribute value, type

```
get_attribute attribute_name [find /des*/design -instance name]
```

dft_abstract_dont_scan

```
dft_abstract_dont_scan {true | false}
```

Read-only `instance` attribute. Indicates whether an abstract segment was defined across the boundaries of this instance. If an abstract segment was defined on a subdesign or instance for which a netlist was also read in, none of the flip-flops in this instance will be considered for scan replacement and will be excluded from scan chain connection if they were not part of the abstract segment.

Related Information

[Defining Abstract Segments in Design for Test in Encounter RTL Compiler.](#)

Affects these commands:

[check_dft_rules](#)

[replace_scan](#)

[synthesize](#)

dft_custom_se

```
dft_custom_se test_signal
```

Read-write `instance` attribute. Indicates which shift-enable signal to connect to the scan flop during scan connection.

The shift-enable signal must have been previously defined with a `define_dft_shift_enable` command.

Note: This attribute applies only to sequential instances of type flop.

Attribute Reference for Encounter RTL Compiler

Design For Test—Instance Attributes

Related Information

Affects these commands: [connect_scan_chains](#)

Related constraint: [define_dft_shift_enable](#)

dft_dont_scan

`dft_dont_scan {inherited | true | false}`

Default: inherited

Read-write [instance](#) attribute. Controls scan replacement of the flip-flop instance for the purposes of test. This attribute can have the following values:

false Allows scan replacement of the flip-flop instance.

inherited Indicates that the instance does inherit the `dft_dont_scan` status from its parent module or hierarchical instance.

true Prevents scan replacement of the flip-flop and excludes the flip-flop from any scan chain.

You must set this attribute prior to running the `check_dft_rules` command.

Related Information

[Marking Objects not to be Mapped to Scan Flip-Flops in Design for Test in Encounter RTL Compiler.](#)

Affects these commands: [check_dft_rules](#)

[synthesize](#)

[connect_scan_chains](#)

[Insert_dft_jtag_macro](#)

Related attributes: (design) [dft_dont_scan](#) on page 1121

(scan_segment) [dft_dont_scan](#) on page 1296

(subdesign) [dft_dont_scan](#) on page 1153

dft_exempt_from_system_clock_check

```
dft_exempt_from_system_clock_check {false | true}
```

Default: false

Read-write instance attribute. Exempts the flop from the system clock violation report when reporting the clock domain information. Setting this attribute removes the flop from the report, but does not otherwise affect how the flop is processed.

Related Information

Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler

Affects this command: report dft_clock_domain_info

dft_force_blackbox_for_atpg

```
dft_force_blackbox_for_atpg {none | false | true}
```

Default: none

Read-write instance attribute. Controls whether the specified instance is to be interpreted as a blackbox and a possible x-source generator by the DFT rule checker.

Related Information

Affects this command: check_dft_rules -advanced

dft_is_blackbox_for_atpg

```
dft_is_blackbox_for_atpg {false | true}
```

Default: false

Read-only instance attribute. Indicates whether the specified instance is interpreted as a blackbox and a possible x-source generator by the DFT rule checker command.

Related Information

Affects this command: check_dft_rules -advanced

Related attribute: dft_force_blackbox_for_atpg

dft_mapped

```
dft_mapped {true | false}
```

Read-only instance attribute. Indicates whether the scan flip-flop instance is mapped for DFT purposes or used for functional purposes.

A (muxed) scan flip-flop is considered mapped for DFT if its shift-enable pin is

- Tied off
- Floating
- Connected to a shift-enable signal defined with a `define_dft shift_enable` constraint.

Note: For scan flip-flops other than the muxed scan flip-flops, this attribute is always `true`, because these scan flip-flops are usually too complex to be used for functional purposes.

Related Information

Affected by these commands: [define_dft shift_enable](#)
[replace_scan](#)
[synthesize](#)

dft_part_of_segment

```
dft_part_of_segment {abstract | fixed | floating | preserve | shift_register}
```

Read-only instance attribute. Returns the type of scan segment a flip-flop belongs to.

Note: This attribute has no value for instances that are not flip-flops, and for flip-flops that are not part of a scan segment.

dft_scan_chain

```
dft_scan_chain string
```

Read-only attribute. Returns the path to the actual scan chain that the instance belongs to.

Note: This attribute has no value for instances that are not flip-flops.

Attribute Reference for Encounter RTL Compiler

Design For Test—Instance Attributes

Example

```
rc:/designs/test/instances_seq> get_att dft_scan_chain out1_reg[112]/
/designs/test/dft/report/actual_scan_chains/AutoChain_3
```

Related Information

Set by this command:

[connect_scan_chains](#)

dft_status

```
dft_status {Passes DFT rules | Fails DFT rules | Abstract Segment Dont scan
| Dont scan | Misc. non scan}
```

Read-only instance attribute. Returns the DFT rule checker (scan) status of the flip-flop.

Abstract Segment Dont scan

Indicates that the instance must not be mapped to a scan. The instance can be an instance of a libcell for which an abstract segment was defined, or can be a flip-flop that belongs to a hierarchical instance for which an abstract segment was defined. In either case, the scan chain information for the instance is assumed from the abstract definition.

Dont scan

Indicates that the flip-flop must not be mapped to a scan flip-flop.

Fails DFT rules

Indicates that the flip-flop failed the DFT rules.

Misc. non scan

Indicates that the instance is a non-scan element. This applies for example to lockup elements, or clock-gating elements.

Passes DFT rules

Indicates that the flip-flop passed the DFT rules.

Note: This attribute has no value if the DFT rule checker has not yet been run, or for instances that are not flip-flops.

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by these commands:

[check_dft_rules](#)

[fix_dft_violations](#)

Related attribute:

(scan_segment) [dft_status](#) on page 1297

dft_test_clock

`dft_test_clock string`

Read-only instance attribute. Returns the path to the `test_clock` object that was created by the `check_dft_rules` command when it identified the test clock for the flip-flop.

Note: For instances that are not flip-flops, this attribute has no value.

Example

```
rc:/designs/core> get_att dft_test_clock [find / -inst out2_reg]  
/designs/core/dft/test_clock_domains/Equiv/test_clock
```

Related Information

[Defining an Equivalent Test Clock for Different Top-level Clock Pins in Design for Test in Encounter RTL Compiler](#)

Set by this command: [check_dft_rules](#)

Related attributes: (actual_scan_segment) [dft_test_clock](#) on page 1266
(scan_segment) [dft_test_clock](#) on page 1299

dft_test_clock_edge

`dft_test_clock_edge {rise | fall}`

Read-only instance attribute. Returns the active edge of the actual source of the test clock when the test clock was associated with multiple clock sources that were defined equivalent in test mode.

Note: For instances that are not flip-flops, this attribute has no value.

Related Information

Set by this command: [check_dft_rules](#)

Related attributes: (actual_scan_segment) [dft_test_clock_edge](#) on page 1267
(scan_segment) [dft_test_clock_edge](#) on page 1299

dft_test_clock_source

```
dft_test_clock_source {pin|port|bus}
```

Read-only instance attribute. Returns for the flip-flop the actual source of the test clock when the test clock was associated with multiple clock sources that were defined equivalent in test mode.

Note: For instances that are not flip-flops, this attribute has no value.

Example

```
rc:/designs/core> get_att dft_test_clock_source [find / -inst out2_reg]  
/designs/core/ports_in/clk2
```

Related Information

[Defining an Equivalent Test Clock for Different Top-level Clock Pins in Design for Test in Encounter RTL Compiler](#)

Set by this command: [define_dft test_clock](#)

dftViolation

```
dftViolation {clock | async set | async reset} #(violation_Id_number)
```

Read-only instance attribute. Returns the type of violation (clock or asynch) for the flip-flop together with the violation ID number given by the `check_dft_rules` command.

Note: For instances that are not flip-flops, this attribute has no value.

Example

```
rc:/> get_att dftViolation /des*/*/*seq/*1  
clock #(0 ) async set #(1 )
```

Related Information

Set by this command: [check_dft_rules](#)

Relate attribute: (scan_segment) [dft violation](#) on page 1300

mbist_instruction_set

`mbist_instruction_set string`

Default: none

Read-write instance attribute. Defines the user-specific MBIST instructions to be used for a block instance in which MBIST logic has been inserted.

string has the following format:

`"tdr_option instruction [tdr_option instruction]..."`

tdr_option corresponds to the name of an `insert_dft mbist` command option that specifies the user-defined instruction name for a test data register

instruction is the name of the user-specific instruction.

Note: Use this attribute to specify which MBIST TDR set must be used for design blocks with MBIST logic inserted, when more than one MBIST TDR set is defined. This is only needed in a bottom-up MBIST flow when a block with MBIST logic is instantiated multiple times and you want to assign separate MBIST TDR sets to these instances. Otherwise all instances of the block will be connected into a single MBIST TDR set or will be assigned to the MBIST TDR set defined during MBIST insertion.

Example

The following command defines three user-specific MBIST instructions for instance `BLOCK_A`. The instructions are defined for the `MBIST`, `MBISTDIAG`, and `BYPASS` test data registers, respectively.

```
set_attr mbist_instruction_set "-run_mbist RUN_MBIST_0 \
                                -diagnose_mbist DIAG_MBIST_0 \
                                -continue_mbist CONT_MBIST_0" BLOCK_A
```

Related Information

[Inserting Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler](#)

Affects this command: [insert_dft mbist](#)

pmbist_instruction_set

`pmbist_instruction_set string`

Default: none

Read-write instance attribute. Defines the user-specific PMBIST instructions to be used for a block instance in which PMBIST logic has been inserted.

string has the following format:

`"instruction_option instruction [instruction_option instruction]..."`

instruction_option corresponds to the name of an `insert_dft pmbist` command option that specifies the user-defined instruction name and a corresponding test data register

instruction is the name of the user-specific instruction.

Note: Use this attribute to specify which PMBIST instruction set must be used for design blocks with PMBIST logic inserted, when more than one PMBIST instruction set is defined. This is only needed in a bottom-up PMBIST flow when a block with PMBIST logic is instantiated multiple times and you want to assign separate MBIST instruction sets to these instances. Otherwise all instances of the block will be connected into a single MBIST instruction set at the current processing level or will be assigned to the PMBIST instruction set defined during PMBIST insertion.

Example

The following command defines four user-specific PMBIST instructions for instance `BLOCK_A`. The instructions are defined for the `MBISTTPN`, `MBISTAMR`, `MBISTSCH`, and `MBISTCHK` test data registers, respectively.

```
set_attr pmbist_instruction_set "-testplan_instruction MBISTTPN_0 \
                                  -constraint_instruction MBISTAMR_0 \
                                  -schedule_instruction MBISTSCH_0 \
                                  -check_instruction MBISTCHK_0" BLOCK_A
```

Related Information

[Inserting Programmable Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler](#)

Affects this command: [insert_dft pmbist](#)

Pin Attributes

Contain information about the specified instance pin. A pin is a single bit connection point on an instance. Pin objects are stored in `pins_in` and `pins_out` directories.

- To set a pin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pin instance/pin]
```

- To get a pin attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pin instance/pin]
```

Note: You may need to specify more than one instance to serve as the search path that will uniquely identify the relevant pin.

dft_constant_value

```
dft_constant_value {logic_0|logic_1|logic_z|no_value}
```

Read-only pin attribute. Indicates whether the value of the pin was propagated from a test signal or a logic constant. A `logic_z` value indicates that the pin is being driven by a tristate buffer whose control signal is not active. A `no_value` value indicates that the pin is not in the path of a test signal or logic constant. The specified test signal value is propagated by running the check_dft_rules command.

Example

The following example shows the returned value for pin `i_core/pins_in/SE1`. This signal is driven by the top-level `shift_enable` signal, `sel`. The value of the attribute is shown before and after the `shift_enable` signal was defined.

```
rc:/> get_attribute dft_constant_value i_core/pins_in/SE1
no_value
rc:/> define_dft shift_enable -name sel -active high sel
...
/designs/top/dft/test_signals/sel
rc:/> check_dft_rules
    Checking DFT rules for 'top' module under 'muxed_scan' style
...
rc:/> get_attribute dft_constant_value i_core/pins_in/SE1
logic_1
```

Related Information

Affected by these constraints: [define dft shift_enable](#)
[define dft test mode](#)

Related attribute: (net) [dft_constant_value](#) on page 1152

dft_controllable

`dft_controllable string`

Read-write [pin](#) attribute. Specifies the logical connectivity across the pins of a blackbox module, from an input port to an output port. Otherwise, when your design has blackbox modules on the test control path, such as the path to the clock, or asynchronous set/reset pins, the `check_dft_rules` command cannot detect whether a direct path exists from a primary input to the flip-flop's clock pin, set pins, or reset pins of the module; and it reports a DFT violation.

This attribute is set on the output pin of a direct path from an input port to an output port of a blackbox. The attribute value has the following format:

`"input_pin_name {non_inverting|inverting}"`

The value specifies the name of the input pin followed by an indication of whether the path to the output pin is inverted or not.

You must define this attribute prior to running the `check_dft_rules` command.

Related Information

[Marking Objects as DFT-Controllable](#) in *Design for Test in Encounter RTL Compiler*.

Affects this command: [check_dft_rules](#)

dft_driven_by_clock

`dft_driven_by_clock {false | true}`

Read-only [pin](#) attribute. Indicates whether this pin is driven by a clock.

Example

```
rc:/designs/vga_csm_pb_DWIDTH24_AWIDTH9> get_attr dft_driven_by_clock \
==> [find / -pin g471/A]
false
```

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Affected by this command: [connect scan chains](#)

Related attributes: (port) [dft driven by clock](#) on page 1160

(subport) [dft driven by clock](#) on page 1155

dft_opcg_domain_clock_pin

`dft_opcg_domain_clock_pin {false | true}`

Default: false

Read-write [pin](#) attribute. This attribute is set in the scan abstract and specifies whether this input pin is connected to an OPCG test clock.

Note: This attribute applies only to input pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Set by this command: [write dft abstract model](#)

Related attributes: (port) [dft_opcg_domain_clock_pin](#) on page 1162

(subport) [dft_opcg_domain_clock_pin](#) on page 1155

dft_opcg_domain_fanout_pin

`dft_opcg_domain_fanout_pin pin_list`

Read-write [pin](#) attribute. This attribute is set in the scan abstract and returns a list of input pins or ports that are directly feeding the output pin.

In this case, the clock domains propagated from this output port will be the same as the clock domains from the corresponding input ports.

Note: This attribute applies only to output pins or ports.

Attribute Reference for Encounter RTL Compiler Design For Test—Pin Attributes

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

Block-Level Domain-Blocking Flow in *Design for Test in Encounter RTL Compiler*

Set by this command:	<u>write dft abstract model</u>
Related attributes:	(port) <u>dft opcg domain fanout pin</u> on page 1163 (subport) <u>dft opcg domain fanout pin</u> on page 1156

dft_opcg_domain_launch_clock

`dft_opcg_domain_launch_clock pin_list`

Read-write `pin` attribute. This attribute is set in the scan abstract and returns a list of clock pins that correspond to the clock domain that launch the data on this output pin. The information is used to propagate the clock domain from this output port of the abstract model.

If the data is launched by an internal clock domain that has no corresponding output port, an empty string will be included. In this case, UNKNOWN will be propagated from this output port of the abstract model.

Note: This attribute applies only to output pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

Block-Level Domain-Blocking Flow in *Design for Test in Encounter RTL Compiler*

dft_opcg_domain_se_input_pin

`dft_opcg_domain_se_input_pin se_pin`

Read-write pin attribute. This attribute is set in the scan abstract and specifies the input pin of the blocking shift enable signal that corresponds to this clock pin.

Note: This attribute applies only to clock input pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Set by this command: [write_dft_abstract_model](#)

Related attributes: (port) [dft_opcg_domain_se_input_pin](#) on page 1164

 (subport) [dft_opcg_domain_se_input_pin](#) on page 1157

dft_opcg_domain_unfenced_capture

`dft_opcg_domain_unfenced_capture {clock_pin_list | NONE | INTERNAL | ""}`

Read-write pin attribute. This attribute is set in the scan abstract and indicates whether additional domain blocking is required at this input pin of the abstract model.

The attribute can have the following values:

<code>clock_pin_list</code>	Specifies the list of clock pins which capture the data without fencing. When the pin is driven by a clock domain other than the ones listed, additional fencing logic is required.
<code>INTERNAL</code>	Indicates that the data is captured by an internal clock domain that has no corresponding output port. In this case, additional domain blocking will be required.
<code>NONE</code>	Indicates that the data is not captured by any flop. It implies that all endpoints of this pin are fenced.
<code>""</code>	An empty string will be treated like <code>INTERNAL</code> .

Note: This attribute applies only to input pins or ports.

Attribute Reference for Encounter RTL Compiler

Design For Test—Pin Attributes

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow](#) in *Design for Test in Encounter RTL Compiler*

Set by this command: [write dft abstract model](#)

Related attributes: (port) [dft opcg domain unfenced capture](#) on page 1165

(subport) [dft opcg domain unfenced capture](#) on page 1157

user_differential_negative_pin

`user_differential_negative_pin string`

Read-write [pin](#) attribute. Specifies the negative-leg pad pin that is associated with the pad cell instance positive-leg pad pin.

`negative-leg-pad-pin-name pad-cell-instance-positive-leg-pad-pin-name`

Example

```
set_attr user_differential_negative_pin PN PAD_CLKDR_P/pins_in/P
```

Related Information

[Identifying Differential I/O Ports](#) in *Design for Test in Encounter RTL Compiler*

Affects this command: [insert dft boundary scan](#)

user_from_core_data

`user_from_core_data string`

Read-write [pin](#) attribute. Specify the name of the corresponding from-core data pin of a pad pin on an I/O pad instance. It is possible to specify different from-core data pins for I/O pad cells that have more than one pad pin by setting this attribute on each of the I/O cells pad pins. This can be useful in the case of multi-pad instances such as SERDES blocks.

`from_core_data_pin_name pad_cell_instance_pad_pin_name`

Example

```
set_attr user_from_core_data fromCore1 i_inpad0/pins_in/pad1
```

Note: An inversion on the data path is specified using the ! character in front of the from_core pin name.

Related Information

[Specifying the Function of I/O Pad Pins Using Attributes and Support for I/O Cells with Multiple PAD Pins in Design for Test in Encounter RTL Compiler](#)

Affects this command: [insert_dft_boundary_scan](#)

user_from_core_enable

```
user_from_core_enable string
```

Read-write [pin](#) attribute. Specify the name of the corresponding from-core enable pin of a pad pin on an I/O pad instance. It is possible to specify different from-core enable pins for I/O pad cells that have more than one pad pin by setting this attribute on each of the I/O cells pad pins. This can be useful in the case of multi-pad instances such as SERDES blocks.

Note: An active low output enable is specified using the ! character in front of the output_enable pin name.

Example

```
set_attr user_from_core_enable !en1 i_inpad0/pins_in/pad1
```

Related Information

[Specifying the Function of I/O Pad Pins Using Attributes and Support for I/O Cells with Multiple PAD Pins in Design for Test in Encounter RTL Compiler](#)

Affects this command: [insert_dft_boundary_scan](#)

user_test_receiver_acmode

`user_test_receiver_acmode string`

Read-write pin attribute. Specifies the name of the test receiver pin for AC and DC mode control.

Affects this command: [insert dft boundary scan](#)

user_test_receiver_data_output

`user_test_receiver_data_output string`

Read-write pin attribute. Specifies the name of the test receiver input from the boundary cell.

Affects this command: [insert dft boundary scan](#)

user_test_receiver_init_clock

`user_test_receiver_init_clock string`

Read-write pin attribute. Specifies the name of the test receiver clock to latch data from the boundary cell.

Affects this command: [insert dft boundary scan](#)

user_test_receiver_init_data

`user_test_receiver_init_data string`

Read-write pin attribute. Specifies the name of the test receiver input from the boundary cell.

Affects this command: [insert dft boundary scan](#)

user_to_core_data

`user_to_core_data string`

Read-write pin attribute. Specify the name of the corresponding to-core data pin of a pad pin on an I/O pad instance. It is possible to specify different to-core data pins for I/O pad cells that have more than one pad pin by setting this attribute on each of the I/O cells pad pins. This can be useful in the case of multi-pad instances such as SERDES blocks.

`to_core_data_pin_name pad_cell_instance_pad_pin_name`

Note: An inversion on the data path is specified using the ! character in front of the `to_core` pin name.

Example

`set_attr user_to_core_data toCore1 i_inpad0/pins_in/pad1`

Related Information

[Support for I/O Cells with Multiple PAD Pins in Design for Test in Encounter RTL Compiler](#)

Affects this command: [insert_dft_boundary_scan](#)

user_to_core_enable

`user_to_core_enable string`

Default: none

Read-write pin attribute. Specify the name of the corresponding to-core enable pin of a pad pin on an I/O pad instance. It is possible to specify different to-core enable pins for I/O pad cells that have more than one pad pin by setting this attribute on each of the I/O cells pad pins. This can be useful in the case of multi-pad instances such as SERDES blocks.

Example

`set_attr user_to_core_enable !en1 i_inpad0/pins_in/pad1`

An active low input enable is specified using the ! character in front of the `to_core` enable pin name.

Related Information

Affects this command: [insert_dft_boundary_scan](#)

wrapper_control

```
wrapper_control {false | true }
```

Default: false

Read-write `pin` attribute. Specifies that a wrapper control signal is applied to the pin.

Related Information

Related attributes: [\(port\) wrapper_control](#) on page 1166
[\(subport\) wrapper_control](#) on page 1166

wrapper_segment

wrapper_segment *string*

Read-write `pin` attribute. Lists the wrapper segments associated with the pin.

Related Information

Set by this command: [insert dft wrapper cell](#)

Related attributes: [\(port\) wrapper_segment](#) on page 1167
[\(support\) wrapper_segment](#) on page 1159

Net Attributes

Contain information about a net in the specified design. A net can be a top-level net or can belong to a hierarchical instance.

- To set a net attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -net instance/net]
```

- To get a net attribute value, type

```
get_attribute attribute_name [find /des*/design -net instance/net]
```

Note: You may need to specify several instances to uniquely identify the net.

dft_clock_domain_info

```
dft_clock_domain_info {UNKNOWN | pin_list}
```

Read-only net attribute. Returns the clock domain information that is propagated during the scan connection step when the dft_opcg_domain_blocking root attribute is set to true.

The value can be UNKNOWN if you set the dft_opcg_block_input_to_flop_paths root attribute to true, allowing propagation of unknown clock information.

If you set the dft_opcg_asserted_domain attribute on an input port, the connected net will have this clock information. Same is true for a net connected to a test clock pin.

Example

- The clock domain propagated on net `dat1_[2]` corresponds to an unknown clock.

```
rc:/designs/myblock/nets> get_att dft_clock_domain_info dat1_i[2]
UNKNOWN
```

- The clock information on clock net `clk_i` corresponds to the driving clock pin `clk_i`.

```
rc:/designs/myblock> get_att dft_clock_domain_info /designs/myblk/nets/clk_i
/designs/myblk/ports_in/clk_i
```

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Affected by this command:

[connect_scan_chains](#)

Affect by these attributes:

[dft_opcg_domain_blocking](#)

[dft_opcg_block_input_to_flop_paths](#) on page 1107

[dft_opcg_asserted_domain](#) on page 1162

dft_constant_value

```
dft_constant_value {logic_0|logic_1|logic_z|no_value}
```

Read-only net attribute. Indicates whether the value of the net was propagated from a test signal or a logic constant. A `logic_z` value indicates that the net is being driven by a tristate buffer whose control signal is not active. A `no_value` value indicates that the net is not in the path of a test signal or logic constant. The specified test signal value is propagated by running the `check_dft_rules` command.

Example

The following example shows the returned value for net `n_19`, which is connected to the output of an OR gate. One of the inputs to this gate is a test mode signal. The value of the attribute is shown before and after the test mode signal was defined.

```
rc:/> get_attribute dft_constant_value n_19
no_value
rc:/> define_dft test_mode -name TM -active high tm
...
/designs/top/dft/test_signals/TM
rc:/> check_dft_rules
  Checking DFT rules for 'top' module under 'muxed_scan' style
...
rc:/> get_attribute dft_constant_value n_19
logic_1
```

Related Information

Affected by these constraints: [define dft shift_enable](#)

[define dft test mode](#)

Related attribute: (pin) [dft_constant_value](#) on page 1141

Subdesign Attributes

Contain information about the subdesigns in the specified design. Subdesigns correspond to Verilog modules or VHDL entities instantiated in the top-level Verilog module or top-level VHDL entity.

- To set a subdesign attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subdesign name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subdesign name]
```

dft_dont_scan

```
dft_dont_scan {inherited | true | false}
```

Default: inherited

Read-write subdesign attribute. Controls scan replacement—for the purposes of test—of the flip-flops in the subdesign. This attribute can have the following values:

false	Allows scan replacement of the flip-flops in the subdesign.
inherited	Indicates that the flip-flops in the subdesign inherit the dft_dont_scan status from the parent module or hierarchical instance.
true	Prevents scan replacement of the flip-flops in the subdesign and excludes the flip-flops from any scan chain.

You must set this attribute prior to running the `check_dft_rules` command.

Attribute Reference for Encounter RTL Compiler

Design For Test—Subdesign Attributes

Related Information

[Marking Objects not to be Mapped to Scan Flip-Flops in Design for Test in Encounter RTL Compiler.](#)

Affects these commands:	<u>check_dft_rules</u> <u>connect_scan_chains</u> <u>synthesize</u>
Related attributes	(design) <u>dft_dont_scan</u> on page 1121 (instance) <u>dft_dont_scan</u> on page 1133 (scan_segment) <u>dft_dont_scan</u> on page 1296

Subport Attributes

Contain information about the subports in the specified design.

- To set a subport attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subport name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subport name]
```

dft_driven_by_clock

```
dft_driven_by_clock {false | true}
```

Read-only subport attribute. Indicates whether this subport is driven by a clock.

Example

```
rc:/designs/> get_attr dft_driven_by_clock [find /designs/*/clut_mem -subport clk]
true
rc:/designs/> get_attr dft_driven_by_clock [find /des*/*/clut_mem -subport di[6]]
false
```

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Affected by this command: [connect scan chains](#)

Related attributes: (pin) [dft driven by clock](#) on page 1142

(port) [dft driven by clock](#) on page 1160

dft_opcg_domain_clock_pin

```
dft_opcg_domain_clock_pin {false | true}
```

Default: false

Read-write subport attribute. This attribute is set in the scan abstract and specifies whether this input port is connected to an OPCG test clock.

Note: This attribute applies only to input pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Set by this command: [write_dft_abstract_model](#)

Related attributes: (pin) [dft_opcg_domain_clock_pin](#) on page 1143

(port) [dft_opcg_domain_fanout_pin](#) on page 1163

dft_opcg_domain_fanout_pin

`dft_opcg_domain_fanout_pin pin_list`

Read-write subport attribute. This attribute is set in the scan abstract and returns a list of input pins or ports that are directly feeding the output port.

In this case, the clock domain propagated from this output port should be the same as the clock domain from the corresponding input ports.

Note: This attribute applies only to output pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Set by this command: [write_dft_abstract_model](#)

Related attributes: (pin) [dft_opcg_domain_fanout_pin](#) on page 1143

(port) [dft_opcg_domain_fanout_pin](#) on page 1163

dft_opcg_domain_launch_clock

`dft_opcg_domain_launch_clock pin_list`

Read-write subport attribute. This attribute is set in the scan abstract and returns a list of clock pins that correspond to the clock domain that launch the data on this output port. The information is used to propagate the clock domain from this output port of the abstract model. If the data is launched by an internal clock domain that has no corresponding output port, an empty string will be included. In this case, UNKNOWN will be propagated from this output port of the abstract model.

Note: This attribute applies only to output pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Set by this command: [write_dft_abstract_model](#)

Related attributes: (pin) [dft_opcg_domain_launch_clock](#) on page 1144
(port) [dft_opcg_domain_launch_clock](#) on page 1163

dft_opcg_domain_se_input_pin

`dft_opcg_domain_se_input_pin se_pin`

Read-write subport attribute. This attribute is set in the scan abstract and specifies the input pin of the blocking shift enable signal that corresponds to this clock port.

Note: This attribute applies only to clock input pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Set by this command: [write_dft_abstract_model](#)

Related attributes: (pin) [dft_opcg_domain_se_input_pin](#) on page 1145
(port) [dft_opcg_domain_se_input_pin](#) on page 1164

dft_opcg_domain_unfenced_capture

`dft_opcg_domain_unfenced_capture {clock_pin_list | NONE | INTERNAL | ""}`

Read-write subport attribute. This attribute is set in the scan abstract and indicates whether additional domain blocking is required at this input port of the abstract model.

Attribute Reference for Encounter RTL Compiler

Design For Test—Subport Attributes

The attribute can have the following values:

<i>clock_pin_list</i>	Specifies the list of clock pins which capture the data without fencing. When the pin is driven by a clock domain other than the ones listed, additional fencing logic is required.
INTERNAL	Indicates that the data is captured by an internal clock domain that has no corresponding output port. In this case, additional domain blocking will be required.
NONE	Indicates that the data is not captured by any flop. It implies that all endpoints of this pin are fenced.
" "	An empty string will be treated like INTERNAL.

Note: This attribute applies only to input pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Set by this command: [write dft abstract model](#)

Related attributes: (pin) [dft_opcg_domain_unfenced_capture](#) on page 1145

(port) [dft_opcg_domain_unfenced_capture](#) on page 1165

wrapper_control

wrapper_control {false | true }

Default: false

Read-write [subport](#) attribute. Specifies that a wrapper control signal is applied to the subport.

Related Information

Related attributes: (port) [wrapper_control](#) on page 1166

(pin) [wrapper_control](#) on page 1150

wrapper_segment

`wrapper_segment string`

Read-write subport attribute. Lists the wrapper segments associated with the subport.

Related Information

Set by this command: [insert dft wrapper cell](#)

Related attributes: [\(port\) wrapper_segment](#) on page 1167

[\(pin\) wrapper_segment](#) on page 1150

Port Attributes

Contain information about the subports in the specified design.

- To set a port attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -port name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name \
[find /des*/design -port name]
```

dft_driven_by_clock

```
dft_driven_by_clock {false | true}
```

Read-only port attribute. Indicates whether this port is driven by a clock.

Example

```
rc:/designs/myblock/> get_attr dft_driven_by_clock [find / -port dat1_i[13]]
false
rc:/designs/myblock/> get_attr dft_driven_by_clock [find / -port clk_i]
true
```

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Affected by this command:

[connect scan chains](#)

Related attributes:

(pin) [dft_driven_by_clock](#) on page 1142

(subport) [dft_driven_by_clock](#) on page 1155

dft_enable hookup_pin

```
dft_enable hookup_pin pin
```

Read-write port attribute. Specifies the enable hookup pin for a bidirectional port with a bidirectional pad attached to it.

When a bidirectional pad is attached to a bidirectional port, the pad must be configured in the proper direction during test. Usually this is done by inserting a test point at the `from_core_enable` or `to_core_enable` pin of the pad.

The test point is inserted using the `configure_pad_dft` command, or using the `-configure_pad` option when defining the scan chains, and the shift-enable and test-mode test signals. You can use this attribute to specify the hookup pin to be used when the test point is inserted.

Usually during bidirectional compression, the `mistr_read` signal is connected to the `from_core_enable` pin of the pad cell connected to the bidirectional scan data input port. You can use this attribute to specify the hookup pin to which the `mistr_read` signal must be connected.

Related Information

Related commands:

[configure_pad_dft](#)
[define_dft_scan_chain](#)
[define_dft_shift_enable](#)
[define_dft_test_mode](#)

dft_enable_hookup_polarity

`dft_enable_hookup_polarity {low | high}`

Read-write port attribute. Indicates the value to drive the `dft_enable_hookup_pin` to configure the pad in output mode.

In the case of bidirectional compression, if the polarity is `low`, the `mistr_read` signal is inverted before connecting it to the `dft_enable_hookup_pin`.

In the case of pad configuration for test, if the polarity is `low`, the test point required to activate or block the `from_core_enable` pin will be inverted when the test point is inserted.

Related Information

Related commands:

[configure_pad_dft](#)
[define_dft_scan_chain](#)
[define_dft_shift_enable](#)
[define_dft_test_mode](#)

dft_opcg_asserted_domain

`dft_opcg_asserted_domain clock_port`

Read-write port attribute. Specifies the clock domain to be propagated from this primary inputs. Specify the input pin to which the clock pin is applied.

When the `dft_opcg_block_input_to_flop_paths` attribute is set to `true`, an unknown clock domain is propagated from each input port. This unknown domain will require fencing regardless of the clock domain of the capturing flop.

However, if every instantiation of the block will drive data from a particular clock domain into this port, you can declare this clock domain on the input port. Then, when the data is captured by a flop in the same domain, unnecessary fencing is avoided.

Example

```
rc:/designs> set_attr dft_opcg_asserted_domain [find /designs/* -port clk_i] \
==> [find /designs/* -port dat1_i[13]]
Setting attribute of port 'dat1_i[13]': 'dft_opcg_asserted_domain' = /designs/
myblock/ports_in/clk_i
```

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Affects this command: [connect_scan_chains](#)

Affected by this attribute: [dft_opcg_block_input_to_flop_paths](#) on page 1107

dft_opcg_domain_clock_pin

`dft_opcg_domain_clock_pin {false | true}`

Default: `false`

Read-write port attribute. This attribute is set in the scan abstract and specifies whether this input port is connected to an OPCG test clock.

Note: This attribute applies only to input pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

- Set by this command: [write_dft_abstract_model](#)
Related attributes: (pin) [dft_opcg_domain_clock_pin](#) on page 1143
(subport) [dft_opcg_domain_clock_pin](#) on page 1155

dft_opcg_domain_fanout_pin

`dft_opcg_domain_fanout_pin pin_list`

Read-write [port](#) attribute. This attribute is set in the scan abstract and returns a list of input pins or ports that are directly feeding the output port.

In this case, the clock domain propagated from this output port should be the same as the clock domain from the corresponding input ports.

Note: This attribute applies only to output pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

- Set by this command: [write_dft_abstract_model](#)
Related attributes: (pin) [dft_opcg_domain_fanout_pin](#) on page 1143
(subport) [dft_opcg_domain_fanout_pin](#) on page 1156

dft_opcg_domain_launch_clock

`dft_opcg_domain_launch_clock pin_list`

Read-write [port](#) attribute. This attribute is set in the scan abstract and returns a list of clock pins that correspond to the clock domain that launch the data on this output port. The information is used to propagate the clock domain from this output port of the abstract model.

Attribute Reference for Encounter RTL Compiler

Design For Test—Port Attributes

If the data is launched by an internal clock domain that has no corresponding output port, an empty string will be included. In this case, UNKNOWN will be propagated from this output port of the abstract model.

Note: This attribute applies only to output pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Set by this command: [write_dft_abstract_model](#)

Related attributes: (pin) [dft_opcg_domain_launch_clock](#) on page 1144

(subport) [dft_opcg_domain_launch_clock](#) on page 1156

dft_opcg_domain_se_input_pin

`dft_opcg_domain_se_input_pin se_pin`

Read-write port attribute. This attribute is set in the scan abstract and specifies the input pin of the blocking shift enable signal that corresponds to this clock port.

Note: This attribute applies only to clock input pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Set by this command: [write_dft_abstract_model](#)

Related attributes: (pin) [dft_opcg_domain_se_input_pin](#) on page 1145

(subport) [dft_opcg_domain_se_input_pin](#) on page 1157

dft_opcg_domain_unfenced_capture

```
dft_opcg_domain_unfenced_capture {clock_pin_list | NONE | INTERNAL | ""}
```

Read-write port attribute. This attribute is set in the scan abstract and indicates whether additional domain blocking is required at this input port of the abstract model.

The attribute can have the following values:

<i>clock_pin_list</i>	Specifies the list of clock pins which capture the data without fencing. When the pin is driven by a clock domain other than the ones listed, additional fencing logic is required.
INTERNAL	Indicates that the data is captured by an internal clock domain that has no corresponding output port. In this case, additional domain blocking will be required.
NONE	Indicates that the data is not captured by any flop. It implies that all endpoints of this pin are fenced.
""	An empty string will be treated like INTERNAL.

Note: This attribute applies only to input pins or ports.

Note: Even though this is a read-write attribute, this attribute is set by the tool and you should not change its value.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Set by this command: [write_dft_abstract_model](#)

Related attributes: (pin) [dft_opcg_domain_unfenced_capture](#) on page 1145

(subport) [dft_opcg_domain_unfenced_capture](#) on page 1157

dft_sdi_output_hookup_pin

dft_sdi_output_hookup_pin

Read-write port attribute. Specifies the output hookup pin of the bidirectional scan data input port when it functions as an output port while reading out the MISR signature during compression.

Note: This attribute applies to a bidirectional port that serves as a scan-in of a compressed scan chain.

Related Information

Affected by this command: [compress scan_chain](#)

dft_sdo_input_hookup_pin

dft_sdi_input_hookup_pin

Read-write port attribute. Specifies the input hookup pin of the bidirectional scan data output port when it functions as an input port for the scan unloading operation during compression.

Note: This attribute applies only to a bidirectional port that serves as a scan-out of a compressed scan chain.

Related Information

Affected by this command: [compress scan_chain](#)

wrapper_control

wrapper_control {false | true }

Default: false

Read-write port attribute. Specifies that a wrapper control signal is applied to the port.

Related Information

Related attributes: [\(pin\) wrapper_control](#) on page 1150

[\(subport\) wrapper_control](#) on page 1158

wrapper_segment

wrapper_segment string

Read-write port attribute. Lists the wrapper segments associated with the port.

Related Information

Set by this command:

[insert_dft_wrapper_cell](#)

Related attributes:

[\(pin\) wrapper_segment](#) on page 1150

(subport) [wrapper_segment](#) on page 1159

Boundary-Scan Segment Attributes

Contain information about the boundary-scan segments in the specified design. These attributes are read-write attributes.

- To set an `boundary_scan_segment` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -boundary_scan_segment segment]
```
- To get a `boundary_scan_segment` attribute value, type

```
get_attribute attribute_name \
[find /des*/design -boundary_scan_segment segment]
```

Note: These attributes are located at

```
/designs/design/dft/boundary_scan/boundary_scan_segments
```

acdcsel_11496

```
acdcsel_11496 {pin | port}
```

Read-write `boundary_scan_segment` attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_ACDCSEL pin on the IEEE 1149.6 JTAG_MACRO.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)
[write_bsdl](#)

acpclk_11496

```
acpclk_11496 {pin | port}
```

Read-write `boundary_scan_segment` attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_ACPSCLK pin on the IEEE 1149.6 JTAG_MACRO.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)

[write_bsdl](#)

acpsen_11496

acpsen_11496 {pin | port}

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_ACPSEN pin on the IEEE 1149.6 JTAG_MACRO.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)

[write_bsdl](#)

acptrenbl_11496

acptrenbl_11496 {pin | port}

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_ACTRENBL pin on the IEEE 1149.6 JTAG_MACRO.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)

[write_bsdl](#)

acpulse_11496

acpulse_11496 {*pin* | *port*}

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_ACPULSE pin on the IEEE 1149.6 JTAG_MACRO.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler](#).

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)

[write_bsdl](#)

bsdl

bsdl *string*

Read-write boundary_scan_segment attribute. Stores the bsdl abstract information that describes the function and position of the boundary cells embedded within a boundary-scan segment and their associated ports.

Note: The attribute value corresponds to the information derived from the file specified for the -bsdl_file option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler](#).

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)

[write_bsdl](#)

capturedr

`capturedr {pin | port}`

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_CAPTUREDR pin on the JTAG_MACRO.

Note: The attribute value corresponds to the value specified for the -capturedr option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define dft boundary scan segment](#)

Affects these commands: [insert dft boundary scan](#)
 [write bsdl](#)

clockdr

`clockdr {pin | port}`

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_BOUNDARY_CLOCKDR pin on the JTAG_MACRO.

The attribute value corresponds to the value specified for the -clockdr option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define dft boundary scan segment](#)

Affects these commands: [insert dft boundary scan](#)
 [write bsdl](#)

differential_pairs

`differential_pairs string`

Read-write boundary_scan_segment attribute. Specifies the full list of differential pin pairs on the boundary-scan segment.

Note: The attribute value corresponds to the value(s) specified for the -differential_pair option(s) when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)
[write_bsdl](#)

highz

`highz {pin | port}`

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_INSTRUCTION_DECODE_CTRL_HIGHZ pin on the JTAG_MACRO.

Note: The attribute value corresponds to the value specified for the -highz option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)
[write_bsdl](#)

instance

`instance string`

Read-write boundary_scan_segment attribute. Specifies the name of the hierarchical instance associated to the boundary-scan segment.

Note: The attribute value corresponds to the value specified for the `-instance` option when the boundary-scan segment was defined. Alternatively, if a boundary-scan segment has been defined on either a libcell or a subdesign, boundary-scan segment objects are created for each instantiation and this attribute set for each boundary-scan segment with the appropriate instance value.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler](#).

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)
[write_bsdl](#)

modea

`modea {pin | port}`

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_INSTRUCTION_DECODE_MODE_A pin on the JTAG_MACRO.

Note: The attribute value corresponds to the value specified for the `-mode_a` option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler](#).

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)
[write_bsdl](#)

modeb

```
modeb {pin | port}
```

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_INSTRUCTION_DECODE_MODE_B pin on the JTAG_MACRO.

Note: The attribute value corresponds to the value specified for the -mode_b option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler](#).

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)
[write_bsdl](#)

modec

```
modec {pin | port}
```

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_INSTRUCTION_DECODE_MODE_C pin on the JTAG_MACRO.

Note: The attribute value corresponds to the value specified for the -mode_c option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler](#).

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)
[write_bsdl](#)

shiftdr

shiftdr {pin | port}

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_BOUNDARY_SHIFTDR pin on the JTAG_MACRO.

Note: The attribute value corresponds to the value specified for the -shiftdr option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler](#).

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)

[write_bsdl](#)

tdi

tdi {pin | port}

Read-write boundary_scan_segment attribute. Specifies the name of the TDI pin on the boundary-scan segment that must be connected in the boundary-scan register.

Note: The attribute value corresponds to the value specified for the -tdi option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler](#).

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)

[write_bsdl](#)

tdo

`tdo {pin | port}`

Read-write boundary_scan_segment attribute. Specifies the name of the TDO pin on the boundary-scan segment that must be connected in the boundary-scan register.

Note: The attribute value corresponds to the value specified for the `-tdo` option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)
[write_bsdl](#)

updatedr

`updatedr {pin | port}`

Read-write boundary_scan_segment attribute. Specifies the name of the pin on the boundary-scan segment that must be connected to the JTAG_BOUNDARY_UPDATEDR pin on the JTAG_MACRO.

Note: The attribute value corresponds to the value specified for the `-updatedr` option when the boundary-scan segment was defined.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define_dft_boundary_scan_segment](#)

Affects these commands: [insert_dft_boundary_scan](#)
[write_bsdl](#)

JTAG Instruction Attributes

Contains the names and information pertaining to the user-defined and mandatory instructions for the specified design.

- To set a `jtag_instruction` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design/dft -jtag_instruction instruction]
```

- To get a `jtag_instruction` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft -jtag_instruction instruction]
```

Note: These attributes are located at

`/designs/design/dft/boundary_scan/jtag_instructions.`

capture

`capture string`

Read-write `jtag_instruction` attribute. Specifies the values that must be captured into the register listed in the `register` attribute during the CaptureDR state.

Note: The attribute value corresponds to the value specified for the `-capture` option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

Related attribute: [\(jtag_instruction_register\) capture](#) on page 1185

length

`length integer`

Read-write `jtag_instruction` attribute. Specifies the length of the register listed in the `register` attribute.

Note: The attribute value corresponds to the value specified for the `-length` option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

Related attribute: [\(jtag_instruction_register\) length](#) on page 1185

opcode

opcode *string*

Read-write [jtag instruction](#) attribute. Specifies the binary code for this instruction. The number of bits in the opcode is determined by the length of the instruction register.

Note: The attribute value corresponds to the value specified for the `-opcode` option when the instruction was defined.

Related Information

Set by this command: [define dft jtag instruction](#)

private

private {false|true}

Default: false

Read-write [jtag instruction](#) attribute. Indicates whether the instruction is defined for a private register.

Note: The attribute value corresponds to the value specified for the `-private` option when the instruction was defined.

Related Information

Set by this command: [define dft jtag instruction](#)

register

register *string*

Read-write [jtag instruction](#) attribute. Specifies the name of the custom test data register (TDR) for which the instruction is defined.

Note: The attribute value corresponds to the value specified for the `-register` option when the instruction was defined. The attribute has no value for mandatory instructions.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_capturedr

`register_capturedr {pin|port}`

Read-write [jtag_instruction](#) attribute. 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.

Note: The attribute value corresponds to the value specified for the -register_capturedr option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_clockdr

`register_clockdr {pin|port}`

Read-write [jtag_instruction](#) attribute. 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.

Note: The attribute value corresponds to the value specified for the -register_clockdr option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_decode

```
register_decode {pin|port}
```

Read-write jtag_instruction attribute. 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, where *instruction* is the name of the defined instruction.

Note: The attribute value corresponds to the value specified for the -register_decode option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_reset

```
register_reset {pin|port}
```

Read-write jtag_instruction attribute. 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.

Note: The attribute value corresponds to the value specified for the -register_reset option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_reset_polarity

```
register_reset_polarity {high|low}
```

Read-write jtag_instruction attribute. Specifies the polarity of the pin on the custom test data register (TDR) that must be connected to the JTAG_RESET pin on the JTAG_MACRO.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_runidle

```
register_runidle {pin|port}
```

Read-write jtag_instruction attribute. 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.

Note: The attribute value corresponds to the value specified for the -register_runidle option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_shiftdr

```
register_shiftdr {pin|port}
```

Read-write jtag_instruction attribute. 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.

Note: The attribute value corresponds to the value specified for the -register_shiftdr option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_shiftdr_polarity

```
register_shiftdr_polarity {high|low}
```

Read-write jtag_instruction attribute. Specifies the polarity of the pin on the custom test data register (TDR) that must be connected to the JTAG_SHIFTDR pin on the JTAG_MACRO.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_tck

`register_tck {pin|port}`

Read-write jtag_instruction attribute. 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.

Note: The attribute value corresponds to the value specified for the –register_tck option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_tdi

`register_tdi {pin|port}`

Read-write jtag_instruction attribute. 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.

Note: The attribute value corresponds to the value specified for the –register_tdi option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_tdo

`register_tdo {pin|port}`

Read-write jtag_instruction attribute. 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.

Note: The attribute value corresponds to the value specified for the –register_tdo option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

register_updatedr

`register_updatedr {pin|port}`

Read-write jtag_instruction attribute. 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..

Note: The attribute value corresponds to the value specified for the -register_updatedr option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

tap_decode

`tap_decode {pin|port}`

Default: JTAG_INSTRUCTION_DECODE_instruction

Read-write jtag_instruction attribute. Specifies the name of the instruction-specific decode pin that must be created on the JTAG_MACRO.

Note: The attribute value corresponds to the value specified for the -tap_decode option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

tap_tdi

`tap_tdi {pin|port}`

Default: JTAG_TDI

Read-write jtag_instruction attribute. Specifies the name of the test data input pin on the JTAG_MACRO.

Note: The attribute value corresponds to the value specified for the `-tap_tdi` option when the instruction was defined.

Related Information

Set by this command: [define dft jtag_instruction](#)

tap_tdo

`tap_tdo {pin|port}`

Default: JTAG_register_TDO

Read-write jtag_instruction attribute. Specifies the name of the instruction-specific test data output (TDO) pin that must be created on the JTAG_MACRO.

Note: The attribute value corresponds to the value specified for the `-tap_tdo` option when the instruction was defined. The attribute has no value for mandatory instructions.

Related Information

Set by this command: [define dft jtag_instruction](#)

JTAG Instruction Register Attributes

Contain information about the JTAG instruction register for the specified design.

- To set a `jtag_instruction_register` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design/dft -jtag_instruction_register instruction]
```

- To get a `jtag_instruction` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft -jtag_instruction_register instruction]
```

Note: These attributes are for the `/designs/design/dft/boundary_scan/jtag_ir` register.

capture

`capture string`

Read-write `jtag_instruction_register` attribute. Specifies the values that must be captured into the instruction register.

Note: The attribute value corresponds to the value specified for the `-capture` option when the instruction register was defined.

Related Information

Set by this command: [define dft jtag_instruction_register](#)

Related attribute: [\(jtag_instruction\) capture](#) on page 1177

length

`length integer`

Read-write `jtag_instruction_register` attribute. Specifies the length of the instruction register.

Note: The attribute value corresponds to the value specified for the `-length` option when the instruction register was defined.

Related Information

Set by this command: [define dft jtag_instruction_register](#)

Related attribute: [\(jtag_instruction\) length](#) on page 1177

JTAG Macro Attributes

Contain the names and information pertaining to the user-defined JTAG_Macro blocks in the specified design. These attributes are read-write attributes.

- To set a `jtag_macro` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design/dft -jtag_macro jtag_macro]
```

Note: These attributes are located at

`/designs/design/dft/boundary_scan/jtag_macros`

boundary_tdo

`boundary_tdo {pin | port | bus}`

Read-write `jtag_macro` attribute. Specifies the boundary-register TDO input pin on the JTAG_Macro.

Related Information

Set by this command: [define dft jtag_macro](#)

bsr_clockdr

`bsr_clockdr {pin | port | bus}`

Read-write `jtag_macro` attribute. Specifies the clock data register (CLOCKDR) output pin for the boundary-scan register.

Related Information

Set by this command: [define dft jtag_macro](#)

bsr_shiftdr

`bsr_shiftdr {pin | port | bus}`

Read-write `jtag_macro` attribute. Specifies the shift data register (SHIFTDR) output pin for the boundary-scan register.

Related Information

Set by this command: [define dft jtag_macro](#)

bsr_updatedr

`bsr_updatedr {pin | port | bus}`

Read-write [jtag_macro](#) attribute. Specifies the update data register (UPDATEDR) output pin for the boundary-scan register.

Related Information

Set by this command: [define dft jtag_macro](#)

capturedr

`capturedr {pin | port | bus}`

Read-write [jtag_macro](#) attribute. Specifies the capture data register (CAPTUREDR) output pin for the custom test data register.

Related Information

Set by this command: [define dft jtag_macro](#)

clockdr

`clockdr {pin | port | bus}`

Read-write [jtag_macro](#) attribute. Specifies the clock data register (CLOCKDR) output pin for the custom test data register.

Related Information

Set by this command: [define dft jtag_macro](#)

dot6_acdcsel

`dot6_acdcsel {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the logical OR of the decoded EXTEST_PULSE and EXTEST_TRAIN instructions.

Related Information

Set by this command: [define dft jtag_macro](#)

dot6_acmode

`dot6_acmode {pin | port | bus}`

Read-write jtag_macro attribute. Specifies an 1149.6 JTAG Macro TRCELL_ACMODE pin.

Related Information

Set by this command: [define dft jtag_macro](#)

dot6_acpulse

`dot6_acpulse {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the AC test signal output of the JTAG Macro.

Related Information

Set by this command: [define dft jtag_macro](#)

dot6_preset_clock

`dot6_preset_clock {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the positive-active edge-sensitive clock signal to test receivers that have edge-sensitive initialization.

Related Information

Set by this command: [define dft jtag_macro](#)

dot6_trcell_enable

`dot6_trcell_enable {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the logical OR of EXTEST, EXTEST_PULSE and EXTEST_TRAIN used to enable the test receiver cells

Related Information

Set by this command: [define dft jtag_macro](#)

exitdr

`exitdr {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the EXIT1DR data register output pin for the custom test data register.

Related Information

Set by this command: [define dft jtag_macro](#)

highz

`highz {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the highz output pin to place the I/O pads in their HIGHZ state.

Related Information

Set by this command: [define dft jtag_macro](#)

instance

`instance string`

Read-write jtag_macro attribute. Specifies the instance of the module where the JTAG_Macro resides.

Attribute Reference for Encounter RTL Compiler

Design For Test—JTAG Macro Attributes

Related Information

Set by this command: [define dft jtag_macro](#)

mode_a

```
mode_a {pin | port | bus}
```

Read-write [jtag_macro](#) attribute. Specifies the mode_a output pin to configure boundary-cells in the boundary-scan register.

Related Information

Set by this command: [define dft jtag_macro](#)

mode_b

```
mode_b {pin | port | bus}
```

Read-write [jtag_macro](#) attribute. Specifies the mode_b output pin to configure boundary-cells in the boundary-scan register.

Related Information

Set by this command: [define dft jtag_macro](#)

mode_c

```
mode_c {pin | port | bus}
```

Read-write [jtag_macro](#) attribute. Specifies the mode_c output pin to configure boundary-cells in the boundary-scan register.

Related Information

Set by this command: [define dft jtag_macro](#)

por

```
por {pin | port | bus}
```

Read-write jtag_macro attribute. Specifies the power-on reset input pin on the JTAG_Macro.

Related Information

Set by this command: [define dft jtag_macro](#)

reset

```
reset {pin | port | bus}
```

Read-write jtag_macro attribute. Specifies the `reset` output pin indicating the JTAG_Macro is in the Test-Logic-Reset state.

Related Information

Set by this command: [define dft jtag_macro](#)

runidle

```
runidle {pin | port | bus}
```

Read-write jtag_macro attribute. Specifies the `runidle` output pin indicating the JTAG_Macro is in the Run-Test-Idle state.

Related Information

Set by this command: [define dft jtag_macro](#)

shiftdr

`shiftdr {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the shift data register (SHIFTDR) output pin for the custom test data register.

Related Information

Set by this command: [define dft jtag_macro](#)

tck

`tck {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the TAP controller TCK input pin on the JTAG_Macro.

Related Information

Set by this command: [define dft jtag_macro](#)

tdi

`tdi {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the TAP controller TDI input pin on the JTAG_Macro.

Related Information

Set by this command: [define dft jtag_macro](#)

tdo

`tdo {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the TAP controller TDO input pin on the JTAG_Macro.

Related Information

Set by this command: [define dft jtag_macro](#)

tdo_enable

`tdo_enable {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the the enable output pin that drives the JTAG TDO output enable pin.

Related Information

Set by this command: [define dft jtag_macro](#)

tms

`tms_enable {pin | port | bus}`

Read-write jtag_macro attribute. Specifies the TAP controller TMS input pin on the JTAG_Macro.

Related Information

Set by this command: [define dft jtag_macro](#)

Attribute Reference for Encounter RTL Compiler

Design For Test—JTAG Macro Attributes

trst

```
trst {pin | port | bus}
```

Read-write jtag_macro attribute. Specifies the TAP controller TRST input pin on the JTAG_Macro.

Related Information

Set by this command: [define dft jtag_macro](#)

updatedr

```
updatedr {pin | port | bus}
```

Read-write jtag_macro attribute. Specifies the update data register (UPDATEDR) output pin for the custom test data register.

Related Information

Set by this command: [define dft jtag_macro](#)

user_defined_macro

```
user_defined_macro {true | false}
```

Default: true

Read-write jtag_macro attribute. Indicates whether the JTAG_Macro has been defined by the user.

This attribute can have the following values:

true	Indicates that an existing JTAG_Macro has been defined by the user.
------	---

false	Indicates that the JTAG_Macro has been inserted by the tool.
-------	--

Related Information

Set by this command: [define dft jtag_macro](#)

JTAG Port Attributes

Contain boundary-scan related information for the top-level ports in the specified design.

- To set a `jtag_port` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design/dft -jtag_port port]
```

- To get a `jtag_port` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft -jtag_port port]
```

Note: These attributes are located at

```
/designs/design/dft/boundary_scan/jtag_ports
```

aio_pin

```
aio_pin {false | true}
```

Default: false

Read-write `jtag_port` attribute. Specifies whether a JTAG port is an advanced I/O port per the IEEE1149.6 standard.

Related Information

[Inserting a JTAG Macro in Design for Test in Encounter RTL Compiler](#)

[Inserting Boundary-Scan Logic in Design for Test in Encounter RTL Compiler](#)

Set by this command: [insert dft jtag macro](#)

Affects these commands: [insert dft boundary scan](#)

[insert dft jtag macro](#)

bcell_location

```
bcell_location string
```

Read-write `jtag_port` attribute. Specifies the location of the boundary cell for this port.

Attribute Reference for Encounter RTL Compiler

Design For Test—JTAG Port Attributes

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

bcell_required

`bcell_required {false | true}`

Default: false

Read-write [jtag port](#) attribute. Specifies whether a boundary cell is required for this type of port.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

bcell_segment

`bcell_segment string`

Default: none

Read-write [jtag port](#) attribute. Specifies the name of the boundary-scan segment that is connected to the port.

Related Information

[Defining Boundary-Scan Segments in Design for Test in Encounter RTL Compiler](#)

[Inserting Boundary-Scan Logic in Design for Test in Encounter RTL Compiler](#)

Affects these commands: [define dft boundary scan segment](#)
[insert dft boundary scan](#)
[insert dft jtag macro](#)

bcell_type

`bcell_type string`

Default: bc_undefined

Read-write jtag_port attribute. Specifies the name of the boundary cell associated with the port. Initially the attribute value defaults to bc_undefined. The attribute value is updated to reflect the boundary cell name when boundary-scan cells are inserted for the functional ports. For TAP ports and dedicated test ports, the attribute value remains as bc_undefined.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

bdy_enable

`bdy_enable {pin|port}`

Read-write jtag_port attribute. Specifies the name of the pin on which the enable boundary cell for the port is inserted. This attribute is only be set on ports with bidirectional and tristate pads.

A port that was excluded for boundary-scan insertion, will have an attribute value of NULL.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

bdy_in

`bdy_in {pin|port}`

Read-write jtag_port attribute. Specifies the name of the pin on which the boundary cell is or must be inserted. The pin is an output pin of an input pad cell.

A port that was excluded for boundary-scan insertion, will have an attribute value of NULL.

Related Information

Set by these commands: [insert dft boundary_scan](#)
[read io speclist](#)

bdy_out

`bdy_out {pin|port}`

Read-write jtag_port attribute. Specifies the name of the pin on which the boundary cell is or must be inserted. The pin is an input pin of an output pad cell.

A port that was excluded for boundary-scan insertion, will have an attribute value of NULL.

Related Information

Set by these commands: [insert dft boundary_scan](#)
[read io speclist](#)

bsr_dummy_after

`bsr_dummy_after integer`

Default: 0

Read-write jtag_port attribute. Specifies the number of dummy boundary cells to add after the boundary cell for this JTAG port (toward the TDI port) in the BSDL description.

Use this attribute when the (custom) boundary-scan macro has more than one SHIFT/CAPTURE latch associated with the I/O.

Attribute Reference for Encounter RTL Compiler

Design For Test—JTAG Port Attributes

Example

Consider the following snippet from the original BSDL file:

```
"1 (MBC, in1(1), INPUT, X), " &
"2 (MBC, in1(2), INPUT, X), " &
"3 (MBC, in1(3), INPUT, X), " &
"4 (BC_CLKIN, rst, CLOCK, X), " &
...
"10 (BC_OUT, out1(0), OUTPUT2, X)";
```

The following command requires two bitshifts towards the TDI port for the custom boundary cell for port in1(3):

```
set_attribute bsr_dummy_after 2 [find / -jtag_port in1[3]]
```

Snippet of resulting BSDL file:

```
"1 (MBC, in1(1), INPUT, X), " &
"2 (MBC, in1(2), INPUT, X), " &
"3 (MBC, in1(3), INPUT, X), " &
"4 (BSR_DUMMY, *, INTERNAL, X), " &
"5 (BSR_DUMMY, *, INTERNAL, X), " &
"6 (BC_CLKIN, rst, CLOCK, X), " &
...
"12 (BC_OUT, out1(0), OUTPUT2, X)";
```

Related Information

[Inserting a JTAG Macro in Design for Test in Encounter RTL Compiler](#)

[Inserting Boundary-Scan Logic in Design for Test in Encounter RTL Compiler](#)

Affects this command: [insert_dft_boundary_scan](#)

bsr_dummy_before

`bsr_dummy_before integer`

Default: 0

Read-write `jtag_port` attribute. Specifies the number of dummy boundary cells to add before the boundary cell for this JTAG port (toward the tdo port) in the BSDL description.

Use this attribute when the (custom) boundary-scan macro has more than one SHIFT/CAPTURE latch associated with the I/O.

Attribute Reference for Encounter RTL Compiler

Design For Test—JTAG Port Attributes

Example

Consider the following snippet from the original BSDL file:

```
"1 (MBC, in1(1), INPUT, X), " &
"2 (MBC, in1(2), INPUT, X), " &
"3 (MBC, in1(3), INPUT, X), " &
"4 (BC_CLKIN, rst, CLOCK, X), " &
...
"10 (BC_OUT, out1(0), OUTPUT2, X)";
```

The following command requires two bitshifts towards the TDO port for the custom boundary cell for port in1(2):

```
set_attribute bsr_dummy_before 2 [find / -jtag_port in1[2]]
```

Snippet of resulting BSDL file:

```
"1 (MBC, in1(1), INPUT, X), " &
"2 (BSR_DUMMY, *, INTERNAL, X), " &
"3 (BSR_DUMMY, *, INTERNAL, X), " &
"4 (MBC, in1(2), INPUT, X), " &
"5 (MBC, in1(3), INPUT, X), " &
"6 (BC_CLKIN, rst, CLOCK, X), " &
...
"12 (BC_OUT, out1(0), OUTPUT2, X)";
```

Related Information

[Inserting a JTAG Macro in Design for Test in Encounter RTL Compiler](#)

[Inserting Boundary-Scan Logic in Design for Test in Encounter RTL Compiler](#)

Affects this command: [insert_dft_boundary_scan](#)

cell

`cell libcell`

Read-write jtag_port attribute. Specifies the name of the pad cell inserted on this port.

Related Information

Set by these commands: [insert_dft_boundary_scan](#)
[read_io_splist](#)

comp_enable

`comp_enable {low|high}`

Read-write jtag_port attribute. Specifies the compliance enable value of this port.

This attribute applies only for dedicated test-related signals such as the test-mode and shift-enable signals.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

custom_bcell

`custom_bcell string`

Read-write jtag_port attribute. Specifies the name of custom boundary cell (to be) inserted on this port.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

differential

`differential port_name`

Read-write jtag_port attribute. Specifies the name of the negative leg port that is associated with the positive leg port (of a differential port) on which it is defined.

Related Information

Set by this command: [insert dft boundary scan](#)
Related command: [define dft boundary scan segment](#)

index

`index integer`

Default: -1

Read-write `jtag_port` attribute. Specifies the index value of the port. The index value is used to determine the order in which the boundary cells are stitched together in the boundary-scan register.

A value of -1 indicates that the information was not provided in an IOspeclist.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

pin

`pin {pin|port}`

Read-write `jtag_port` attribute. Specifies the name of the corresponding top-level port.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

Related attribute: (test_signal) `pin` on page 1328

pinmap

`pinmap string`

Read-write `jtag_port` attribute. Specifies the name of the corresponding package pin.

This attribute value will be empty if no pinmap file was specified when inserting boundary scan.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

sys_enable

`sys_enable {pin|port}`

Read-write [jtag_port](#) attribute. Specifies the pin used to control the enable pin on bidirectional and tristate pads. System or functional I/O enables can be driven either from internal core logic or from another signal coming on-chip through a top-level port.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

sys_use

`sys_use string`

Read-write [jtag_port](#) attribute. Specifies the functional use of the port.

Possible values are `clock`, `input`, `output`, `enable`, `none` or `undefined`.

Note: For each unique `sys_enable` statement in the IOSpecList input file, a separate `sys_use=ENABLE` statement is also specified to indicate that a boundary cell is to be inserted for the output enable or tristate control pins for `output3` or bidirectional pads, and to define `boundary_cell` position in the boundary-scan register.

A port that was excluded for boundary-scan insertion, will have an attribute value of `none`. A TAP port has an attribute value of `undefined`.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

test_use

`test_use string`

Read-write jtag_port attribute. Specifies the test use of the port.

A port that was excluded for boundary-scan insertion, will have an attribute value of `none`. A TAP port has an attribute value of `undefined`.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

tr_bdy_in

`tr_bdy_in {pin|port/bus}`

Read-write jtag_port attribute. Specifies the test receiver output pin associated with the JTAG port. This is the pin on which the BC_11496_ACTR boundary cell is or must be inserted.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

tr_cell

`trcell string`

Read-write jtag_port attribute. Specifies the test receiver cell for the port.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

trcell_acmode

`trcell_acmode {pin|port/bus}`

Read-write jtag_port attribute. Specifies the test AC mode pin associated with the JTAG port.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

trcell_clock

`trcell_clock {pin|port/bus}`

Read-write jtag_port attribute. Specifies the test receiver clock pin associated with the JTAG port.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

trcell_enable

`trcell_enable {pin|port/bus}`

Read-write jtag_port attribute. Specifies the test receiver enable pin associated with the JTAG port.

Related Information

Set by these commands: [insert dft boundary scan](#)
[read io speclist](#)

type

type *string*

Read-write jtag_port attribute. Specifies the type of the JTAG port. Possible values are tdi, tdo, trst, tms, tck and non_jtag.

Related Information

Set by these commands:

[insert dft boundary_scan](#)

[read_io_speclist](#)

TAP Port Attributes

Contain information about the TAP ports in the specified design. These attributes are set using the `define_dft tap_port` command.

These attributes are read-only attributes, so you cannot set their values.

- To get an `tap_port` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft -tap_port tap_port]
```

Note: These attributes are located at

```
/designs/design/dft/boundary_scan/tap_ports
```

dft hookup pin

```
dft_hookup_pin {pin | port}
```

Read-only `tap_port` attribute. Returns the path to the pin or port where the TAP signal actually hooks up inside the core.

Related Information

Set by this constraint: [define_dft tap_port](#)

dft hookup polarity

```
dft_hookup_polarity {inverted | non_inverted}
```

Read-only `tap_port` attribute. Indicates whether the TAP signal is inverted or not at the hookup pin or port.

Related Information

Set by this constraint: [define_dft tap_port](#)

pin

```
pin string
```

Read-only `tap_port` attribute. Returns the port or hierarchical pin associated with the TAP signal.

Attribute Reference for Encounter RTL Compiler

Design For Test—TAP Port Attributes

Related Information

Set by this constraint: [define_dft tap_port](#)

Related attribute: [\(jtag_port\) pin](#) on page 1202

type

`type test_signal_type`

Read-only tap_port attribute. Returns the type of this signal . This attribute can have the following values: tck, tdi, tdo, tms, trst, or tdo_enable.

Related Information

Set by this constraint: [define_dft tap_port](#)

DFT Configuration Mode Attributes

Contain information about the DFT configuration modes. These attributes are read-only attributes, so you cannot set their values.

- To get a `dft_configuration_mode` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/ -dft_configuration_mode mode]
```

Note: These attributes are located at

```
/designs/design/dft/dft_configuration_modes
```

current_mode

```
current_mode {false | true}
```

Default: false

Read-only `dft_configuration_mode` attribute. Indicates if the queried scan mode is the current mode.

Related Information

[Defining Scan Chain Configuration Modes in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [`define_dft dft_configuration_mode`](#)

Affects these commands: [`check_dft_rules`](#)

[`check_dft_rules -mbist`](#)

[`compress_scan_chains`](#)

[`concat_scan_chains`](#)

[`connect_scan_chains`](#)

decoded_pin

```
decoded_pin {pin| port}
```

Read-write `dft_configuration_mode` attribute. Specifies the output pin of the decoder logic that will be activated when all the test signals that define this configuration mode (of type `wrapper`) attain the value specified in the definition of this configuration mode.

This attribute is set to NULL when no decoder logic was inserted or if the configuration mode is not of type `wrapper`.

Related Information

[Defining Scan Chain Configuration Modes](#) in *Design for Test in Encounter RTL Compiler*.

Set by this command: [insert dft wrapper mode decode block](#)

Affects these commands: [concat scan chains](#)

[connect scan chains](#)

[insert dft wrapper cell](#)

jtag_instruction

`jtag_instruction instruction_name`

Read-only [dft configuration mode](#) attribute. Returns the name of the user-defined JTAG instruction associated with the configuration mode.

Related Information

Set by this command: [define dft dft configuration mode](#)

[compress scan chains](#)

Related command: [define dft jtag instruction](#)

mode_enable_high

`mode_enable_high string`

Read-only [dft configuration mode](#) attribute. Specifies the test signals that are held to active high value for the mode.

Related Information

[Defining Scan Chain Configuration Modes](#) in *Design for Test in Encounter RTL Compiler*.

Set by this command: [define dft dft configuration mode](#)

Affects these commands: [check dft rules](#)

[check dft rules -mbist](#)

[compress scan chains](#)

[concat_scan_chains](#)
[connect_scan_chains](#)
[define_dft_abstract_segment](#)
[report_dft_chains](#)
[write_atpg](#)
[write_dft_abstract_model](#)
[write_et_atpg](#)
[write_scandef](#)

mode_enable_low

`mode_enable_low string`

Read-only dft_configuration_mode attribute. Specifies the test signals that are held to active low value for the mode.

Related Information

[Defining Scan Chain Configuration Modes in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [define_dft_dft_configuration_mode](#)

Affects these commands: [check_dft_rules](#)

[compress_scan_chains](#)
[concat_scan_chains](#)
[connect_scan_chains](#)
[define_dft_abstract_segment](#)
[report_dft_chains](#)
[write_atpg](#)
[write_dft_abstract_model](#)
[write_et_atpg](#)
[write_scandef](#)

type

type *string*

Read-only `dft configuration mode` attribute. Returns the type of the configuration mode. Possible values are `scan`, `mbist`, `compression.`, and `wrapper`.

Related Information

[Defining Scan Chain Configuration Modes in Design for Test in Encounter RTL Compiler.](#)

Set by this command:	define dft dft configuration mode
Affects these commands:	check dft rules compress scan chains concat scan chains connect scan chains define dft abstract segment insert dft wrapper cell insert dft wrapper mode decode block report dft chains write atpg write dft abstract model write et atpg write scandef

usage

usage *string*

Read-only `dft configuration mode` attribute. Returns the usage if the configuration mode is of type `wrapper`. Possible values are `mission`, `intest` and `extest`.

Note: The attribute value corresponds to the value specified for the `-usage` option when the wrapper configuration mode was defined.

Related Information

[Defining Scan Chain Configuration Modes](#) in *Design for Test in Encounter RTL Compiler*.

Set by this command:	<u>define_dft dft configuration mode</u>
Affects these commands:	<u>concat scan chains</u> <u>connect scan chains</u> <u>insert dft wrapper cell</u> <u>insert dft wrapper mode decode block</u>

user_defined

`user_defined string`

Read-only [dft configuration mode](#) attribute. Returns the user-defined configuration mode.

Related Information

[Defining Scan Chain Configuration Modes](#) in *Design for Test in Encounter RTL Compiler*.

Set by this command:	<u>define_dft dft configuration mode</u>
Affects these commands:	<u>check_dft_rules</u> <u>compress_scan_chains</u> <u>concat_scan_chains</u> <u>connect_scan_chains</u> <u>define_dft_abstract_segment</u> <u>report_dft_chains</u> <u>write_atpg</u> <u>write_dft_abstract_model</u> <u>write_et_atpg</u> <u>write_scandef</u>

Memory Data Bit Structure Attributes

Contain information about the physical order of rows and columns for a particular data-bit. These attributes are normally set when the configuration view file is read using the `read_memory_view` command.

These attributes are read-only attributes, so you cannot set their values.

- To get an `memory_data_bit_structure` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/mbist -memory_data_bit_structure data_bit]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/configuration/memory_library_domain/memory_libcell/
address_partition/bank_x/data_bit_x
```

column_order

`column_order string`

Read-only `memory_data_bit_structure` attribute. Returns the order of columns in this data-bit of the memory, which was specified using the `address_partition` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

partial_row_order

`partial_row_order string`

Read-only `memory_data_bit_structure` attribute. Returns the order of rows in the last row group in this data-bit of the memory, which was specified using the `address_partition` specification in the PMBIST configuration file. It differs from the `row_order` attribute, only when the last row group is not fully populated.

Related Information

[Customizing a PMBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

row_order

`row_order string`

Read-only memory_data_bit_structure attribute. Returns the order of rows in this data-bit of the memory, which was specified using the `address_partition` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

Memory Libcell Attributes

Contain information about the memory's physical storage cell layout and addressing scheme. These attributes are normally set when the configuration view file is read using the `read_memory_view` command.

These attributes are read-only attributes, so you cannot set their values.

- To get an `memory_libcell` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/mbist -memory_libcell memory_libcell]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/configuration/memory_library_domain/memory_libcell/
```

address_limit

`address_limit integer`

Read-only `memory libcell` attribute. Returns the number of used or addressable words in the memory, which was specified using the `address_limit` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

data_order

`data_order string`

Read-only `memory libcell` attribute. Returns the physical order of the data-bits within the memory word, which was specified using the `data_order` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

memory_libcell

`memory_libcell string`

Read-only `memory_libcell` attribute. Returns the path to the corresponding memory libcell in the library.

Example

```
rc:/designs/TOP_CORE/dft/mbist/configuration/memory_libcells/None/RF32X32>
get_attr memory_libcell
/libraries/RF32X32_lib/libcells/RF32X32
```

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

read_delay

`read_delay integer`

Read-only `memory_libcell` attribute. Returns the intrinsic read delay of the selected memory modules, which was specified using the `read_delay` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

wrapper

`wrapper string`

Read-only `memory_libcell` attribute. Returns the wrapper module that contains the actual memory module, which was specified using the `wrapper` specification in the MBIST configuration file.

Attribute Reference for Encounter RTL Compiler

Design For Test—Memory Libcell Attributes

Related Information

[Customizing an MBIST Configuration File in *Design for Test in Encounter RTL Compiler*](#)

Set by this command: [read_memory_view](#)

Memory Libpin Action Attributes

Contain information on how to control the memory ports that are not used by the MBIST engine as specified in the port_action section of the configuration view file. These attributes are normally set using the `read_memory_view` command.

These attributes are read-only attributes, so you cannot set their values.

- To get a `memory_libpin_action` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/mbist -memory_libpin_action port]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/configuration/memory_library_domain/memory_libcell/
port_action
```

value

`value string`

Read-only `memory_libpin_action` attribute. Returns the value to which the memory port is controlled, which was specified using the `port_action` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

Memory Libpin Alias Attributes

Contain information on how to alias unrecognized port names to the base port names in the Liberty file or wrapper module. These attributes are normally set when the configuration view file is read using the `read_memory_view` command.

These attributes are read-only attributes, so you cannot set their values.

- To get a `memory_spare_column` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/mbist -memory_libpin_alias port]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/configuration/memory_library_domain/memory_libcell/
port_alias/port
```

base_port_name

`base_port_name string`

Read-only `memory_libpin_alias` attribute. Returns the recognized Liberty file base port name, which was specified using the `port_alias` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

Memory Spare Column Attributes

Contain information about spare column resources. These attributes are normally set when the configuration view file is read using the `read_memory_view` command.

These attributes are read-only attributes, so you cannot set their values.

- To get a `memory_spare_column` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/mbist -memory_spare_column column]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/configuration/memory_library_domain/memory_libcell/
redundancy/column_x
```

address_bits

`address_bits string`

Read-only `memory_spare_column` attribute. Returns the logical address bits used to specify spare columns or blocks of columns of the memory, which was specified using the redundancy specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

banks

`banks string`

Read-only `memory_spare_column` attribute. Returns the banks associated with this spare column resource, which was specified using the `banks` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

data_bits

`data_bits string`

Read-only `memory_spare_column` attribute. Returns the data bits associated with this spare column resource, which was specified using the `data` keyword of the redundancy specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

enable

`enable string`

Read-only `memory_spare_column` attribute. Returns the enable signal associated with the repair register, which was specified using the `map` keyword of the redundancy specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

srclk

`srclk string`

Read-only `memory_spare_column` attribute. Returns the register shift clock associated with the repair register, which was specified using the `map` keyword of the redundancy specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

srsi

srsi *string*

Read-only memory_spare_column attribute. Returns the register shift input associated with the repair register, which was specified using the `map` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

srs0

srs0 *string*

Read-only memory_spare_column attribute. Returns the register shift output associated with the therepair register, which was specified using the `map` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

Memory Spare Column Map Address Attributes

Contain information about spare column resource allocation to values on ports or memory module internal repair registers. These attributes are normally set when the configuration view file is read using the `read_memory_view` command.

These attributes are read-only attributes, so you cannot set their values.

- To get a `memory_spare_column_map_address` attribute value, type

```
get_attribute attribute_name [find /des*/design/dft/mbist \
-memory_spare_column_map_address address_range_x]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/configuration/memory_library_domain/memory_libcell/
redundancy/column_x/memory_spare_column_map/address_range_x
```

address_logical_value

`address_logical_value` *string*

Read-only `memory_spare_column_map_address` attribute. Returns the logical value for the address field, which was specified using the `map` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

address_port

`address_port` *string*

Read-only `memory_spare_column_map_address` attribute. Returns the name of the address port, which was specified using the `map` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

address_port_value

`address_port_value string`

Read-only `memory_spare_column_map_address` attribute. Returns the enable value of the address port, which was specified using the `map` keyword of the redundancy specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

Memory Spare Column Map Data Attributes

Contain information about spare column resource allocation to values on ports or memory module internal repair registers. These attributes are normally set when the configuration view file is read using the `read_memory_view` command.

These attributes are read-only attributes, so you cannot set their values.

- To get a `memory_spare_column_map_data` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/mbist -memory_spare_column_map_data data_range_x]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/configuration/memory_library_domain/memory_libcell/
redundancy/column_x/memory_spare_column_map/data_range_x
```

data_logical_value

`data_logical_value string`

Read-only `memory_spare_column_map_data` attribute. Returns the logical value for the data field, which was specified using the `data` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

data_port

`data_port string`

Read-only `memory_spare_column_map_data` attribute. Returns the name of the data port, which was specified using the `data` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

data_port_value

`data_port_value string`

Read-only `memory_spare_column_map_data` attribute. Returns the enable value of the data port, which was specified using the `data` keyword of the redundancy specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

Memory Spare Row Attributes

Contain information about spare row resources. These attributes are normally set when the configuration view file is read using the `read_memory_view` command.

These attributes are read-only attributes, so you cannot set their values.

- To get a `memory_spare_row` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/mbist -memory_spare_row row]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/configuration/memory_library_domain/memory_libcell/
redundancy/row_x/
```

address_bits

`address_bits string`

Read-only `memory_spare_row` attribute. Returns the logical address bits used to specify spare rows or blocks of rows of the memory, which was specified using the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

banks

`banks string`

Read-only `memory_spare_row` attribute. Returns the banks associated with this spare row resource, which was specified using the `banks` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

data_bits

`data_bits string`

Read-only `memory_spare_row` attribute. Returns the data bits associated with this spare row resource, which was specified using the `data` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

enable

`enable string`

Read-only `memory_spare_row` attribute. Returns the enable signal associated with the repair register, which was specified using the `map` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

srclk

`srclk string`

Read-only `memory_spare_row` attribute. Returns the register shift clock associated with the repair register, which was specified using the `map` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

Attribute Reference for Encounter RTL Compiler

Design For Test—Memory Spare Row Attributes

srsi

`srsi string`

Read-only `memory_spare_row` attribute. Returns the register shift input associated with the repair register, which was specified using the `map` keyword of the redundancy specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

srs0

`srs0 string`

Read-only `memory_spare_row` attribute. Returns the register shift output associated with the repair register, which was specified using the `map` keyword of the redundancy specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

Memory Spare Row Map Address Attributes

Contain information about spare row resource allocation to values on ports or memory module internal repair registers. These attributes are normally set when the configuration view file is read using the `read_memory_view` command

These attributes are read-only attributes, so you cannot set their values.

- To get a `memory_spare_row_map_address` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/mbist -memory_spare_row_map_address address_range_x]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/configuration/memory_library_domain/memory_libcell/
redundancy/row_x/memory_spare_row_map/address_range_x
```

address_logical_value

`address_logical_value` *string*

Read-only `memory_spare_row_map_address` attribute. Returns the logical value for the address field, which was specified using the `map` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

address_port

`address_port` *string*

Read-only `memory_spare_row_map_address` attribute. Returns the name of the address port, which was specified using the `map` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

address_port_value

`address_port_value string`

Read-only `memory_spare_row_map_address` attribute. Returns the enable value of the address port, which was specified using the `map` keyword of the `redundancy` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

Write Mask Bit Attributes

Contain information about the write enable mask and its binding to the different data bits. These attributes are normally set when the configuration view file is read using the `read_memory_view` command.

These attributes are read-only attributes, so you cannot set their values.

- To get a `write_mask_bit` attribute value, type

```
get_attribute attribute_name [find /des*/design/dft/mbist \
    -write_mask_bit memory_libcell/write_mask_*/bit]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/configuration/memory_library_domain/memory_libcell/
    write_mask_binding/bit
```

masked_bits

`masked_bits string`

Read-only `write_mask_bit` attribute. Returns the masked data bits of the memory word, which was specified using the `write_mask_binding` specification in the MBIST configuration file.

Related Information

[Customizing an MBIST Configuration File in Design for Test in Encounter RTL Compiler](#)

Set by this command: [read_memory_view](#)

Direct Access Function Attributes

Contain information about MBIST direct access functions.

These attributes are read-only attributes, so you cannot set their values.

- To get a `direct_access_function` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft -direct_access_function function]
```

Note: These attributes are located at:

```
/designs/design/dft/mbist/direct_access/direct_access_function
```

active

```
active {high | low}
```

Read-only `direct_access_function` attribute. Returns the active value of the MBIST direct access function signal, which was specified using the `-active` option of the `define_dft mbist_direct_access` command.

For the `monitor` function, the active state indicates the expected state when the MBIST completes running successfully.

Related Information

[Inserting Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler](#)

Set by this constraint: [define_dft mbist_direct_access](#)

Related attributes: [clocked_by_mtclk](#) on page 1234

[source](#) on page 1235

clocked_by_mtclk

```
clocked_by_mtclk {false | true}
```

Read-only `direct_access_function` attribute. Indicates whether this MBIST direct access function is clocked by the `mtclk` clock, which was specified using the `-mtclk` option of the `define_dft mbist_direct_access` command.

This attribute only applies to the `burnin_run` or `poweron_run` functions.

Attribute Reference for Encounter RTL Compiler

Design For Test—Direct Access Function Attributes

Related Information

[Inserting Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler](#)

Set by this constraint: [define_dft_mbist_direct_access](#)

Related attributes: [active](#) on page 1234

[source](#) on page 1235

source

`source {pin | port}`

Read-only [direct access function](#) attribute. Returns the source pin or port of the MBIST direct access function.

Related Information

[Inserting Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler](#)

Set by this constraint: [define_dft_mbist_direct_access](#)

Related attributes: [active](#) on page 1234

[clocked_by_mtclk](#) on page 1234

Programmable Direct Access Function Attributes

Contain information about PMBIST direct access functions.

These attributes are read-only attributes, so you cannot set their values.

- To get a `programmable_direct_access_function` attribute value, type
`get_attribute attribute_name \`
`[find /des*/design/dft -programmable_direct_access_function function]`

Note: These attributes are located at:

`/designs/design/dft/mbist/direct_access/programmable_direct_access_function`

active

`active {high | low}`

Read-only `programmable_direct_access_function` attribute. Returns the active value of the PMBIST direct access function signal at the hook-up point, which was specified using the `-active` option of the `define_dft_pmbist_direct_access` command.

Related Information

[Inserting Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler](#)

Set by this constraint: [`define_dft_pmbist_direct_access`](#)

Related attributes: [`source`](#) on page 1236

source

`source {pin | port}`

Read-only `programmable_direct_access_function` attribute. Returns the hook-up pin or port from where to make the connection for this PMBIST direct access function.

Related Information

[Inserting Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler](#)

Set by this constraint: [`define_dft_mbist_direct_access`](#)

Related attributes: [`active`](#) on page 1236

MBIST Clock Attributes

Contain information about the MBIST clocks in the specified design. These attributes are normally set using the `define_dft mbist_clock` command.

These attributes are read-only attributes, so you cannot set their values.

- To get an `mbist_clock` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft -mbist_clock mbist_clock]
```

Note: These attributes are located at

```
/designs/design/dft/mbist/mbist_clocks
```

dft hookup pin

```
dft_hookup_pin {pin | port}
```

Read-only `mbist_clock` attribute. Returns the path to the pin or port where the mbist clock actually hooks up inside the core.

Example

```
rc:/designs/test> get_att dft_hookup_pin dft/mbist/mbist_clocks/clk
/designs/test/instances_comb/clk_mux/pins_out/Y
```

Related Information

Set by this constraint: [define_dft mbist_clock](#)

dft hookup polarity

```
dft_hookup_polarity {inverted | non_inverted}
```

Read-only `mbist_clock` attribute. Indicates whether the test signal is inverted or not at the hookup pin or port.

Example

```
rc:/designs/test> get_att dft_hookup_polarity dft/mbist/mbist_clocks/clk
non_inverted
```

Related Information

Set by this constraint: [define_dft mbist_clock](#)

hookup_period

`hookup_period integer`

Default: same value as period

Read-only [mbist_clock](#) attribute. Returns the value specified using the `-hookup_period` option of the `define_dft mbist_clock` command.

Related Information

Set by this constraint: [define_dft mbist_clock](#)

internal

`internal {false|true}`

Read-only [mbist_clock](#) attribute. Indicates whether the source of the MBIST clock is internal to the design (for example, from an analog block). Set by the `-internal_clock_source` option of the `define_dft mbist_clock` command.

Related Information

Set by this constraint: [define_dft mbist_clock](#)

is_jtag_tck

`is_jtag_tck {false| true}`

Read-only [mbist_clock](#) attribute. Indicates whether this MBIST clock is defined as a JTAG clock.

Related Information

Set by this constraint: [define_dft mbist_clock](#)

period

period integer

Read-only [mbist_clock](#) attribute. Returns the value specified using the –period option of the `define_dft mbist_clock` command.

Related Information

Set by this constraint: [define_dft mbist_clock](#)

sources

sources string

Read-only [mbist_clock](#) attribute. Returns the port that is the source or test time control for an internal clock of the MBIST clock waveform.

Example

```
rc:/> get_att sources [find /*/dft -mbist_clock mbist_clock]  
/designs/core/ports_in/clk1
```

Related Information

Set by this constraint: [define_dft mbist_clock](#)

Domain Macro Parameters Attributes

Contain information about the domain macro parameters defined by the user for a domain macro. These attributes are read-only attributes, so you cannot set their values.

- To get an `domain_macro_parameter` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/opcg -domain_macro_parameter name]
```

These attributes are located at:

```
/designs/design/dft/opcg/domain_macro_parameters
```

counter_length

`counter_length integer`

Read-only `domain_macro_parameter` attribute. Returns the number of bits in the down counter in the domain macro, which was specified using the `-counter_length` option of the `define_dft domain_macro_parameters` command.

Related Information

Set by this constraint:

[define_dft domain_macro_parameters](#)

max_num_pulses

`max_num_pulses integer`

Read-only `domain_macro_parameter` attribute. Returns the number of pulses generated by the domain macro, which was specified using the `-max_num_pulses` option of the `define_dft domain_macro_parameters` command.

Related Information

Set by this constraint:

[define_dft domain_macro_parameters](#)

target_period

`target_period float`

Read-only domain macro parameter attribute. Returns the operating target period for the domain macro in picoseconds, which was specified using the `-min_target_period` option of the `define_dft domain_macro_parameters` command.

Related Information

Set by this constraint: [define_dft domain_macro_parameters](#)

trigger_delay

`trigger_delay float`

Read-only domain macro parameter attribute. Returns the time (in picoseconds) after which the first pulse must be issued by the domain macro after receipt of the TRIGGERRUN signal from the trigger macro, which was specified using the `-max_trigger_delay` option of the `define_dft domain_macro_parameters` command.

Related Information

Set by this constraint: [define_dft domain_macro_parameters](#)

OPCG Domain Attributes

Contain information about the OPCG domain macros. These attributes are read-only attributes, so you cannot set their values.

- To get an `opcg_domain` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/opcg -opcg_domain name]
```

Note: These attributes are located at:

```
/designs/design/dft/opcg/opcg_domains
```

divide_by

```
divide_by integer
```

Read-only `opcg_domain` attribute. Returns the value by which to divide the oscillator source frequency, which was specified using the `-divide_by` option of the `define_dft opcg_domain` command.

Related Information

Set by this constraint: [define_dft opcg_domain](#)

domain_macro_parameter

```
domain_macro_parameter string
```

Read-only `opcg_domain` attribute. Returns the name of the domain macro parameter set that applies to the OPCG domain, which was specified using the `-domain_macro_parameter` option of the `define_dft opcg_domain` command.

Related Information

Set by this constraint: [define_dft opcg_domain](#)

location

```
location {pin | port | bus}
```

Read-only `opcg_domain` attribute. Returns where the domain macro will be inserted, which was specified using the `-location` option of the `define_dft opcg_domain` command.

Attribute Reference for Encounter RTL Compiler

Design For Test—OPCG Domain Attributes

Related Information

Set by this constraint: [define_dft opcg_domain](#)

min_domain_period

`min_domain_period float`

Read-only [opcg_domain](#) attribute. Returns the minimum period at which this OPCG domain can operate, which was specified using the `-min_domain_period` option of the `define_dft opcg_domain` command.

Related Information

Set by this constraint: [define_dft opcg_domain](#)

opcg_trigger

`opcg_trigger string`

Read-only [opcg_domain](#) attribute. Returns the OPCG trigger for this OPCG domain, which was specified using the `-opcg_trigger` option of the `define_dft opcg_domain` command.

Related Information

Set by this constraint: [define_dft opcg_domain](#)

osc_source

`osc_source string`

Read-only [opcg_domain](#) attribute. Returns the oscillator source for this OPCG domain, which was specified using the `-osc_source` option of the `define_dft opcg_domain` command.

Related Information

Set by this constraint: [define_dft opcg_domain](#)

scan_clock

`scan_clock test_clock`

Read-only `opcg_domain` attribute. Specifies the test clock that must drive the flops inside the domain macro during full scan mode.

Related Information

Set by this constraint: [define_dft opcg_trigger](#)

Related command: [define_dft test_clock](#)

OPCG Mode Attributes

Contain information about the OPCG modes. These attributes are read-only attributes, so you cannot set their values.

- To get an `opcg_mode` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/opcg -opcg_mode name]
```

Note: These attributes are located at:

```
/designs/design/dft/opcg/opcg_modes
```

jtag_controlled

```
jtag_controlled {false |true}
```

Default: false

Read-write `opcg_mode` attribute. Specifies whether a JTAG instruction is used to lock the PLLs for OPCG operation, which was specified using the `-jtag_controlled` option of the `define_dft opcg_mode` command.

Related Information

Set by this constraint: [define_dft opcg_mode](#)

mode_init

```
mode_init file
```

Read-write `opcg_mode` attribute. Returns the initialization sequence file for this OPCG mode, which was specified using the `-mode_init` option of the `define_dft opcg_mode` command.

Related Information

Set by this constraint: [define_dft opcg_mode](#)

OPCG Trigger Attributes

Contain information about the OPCG trigger signals defined by the user. These attributes are read-only attributes, so you cannot set their values.

- To get an `opcg_trigger` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/opcg -opcg_trigger signal]
```

Note: These attributes are located at:

```
/designs/design/dft/opcg/opcg_triggers
```

active

```
active {low | high}
```

Read-only `opcg_trigger` attribute. Returns the active value for the OPCG trigger signal, which was specified using the `-active` option of the `define_dft opcg_trigger` command.

Related Information

Set by this constraint: [define_dft opcg_trigger](#)

Related attributes:
(active_scan_segment) [active](#) on page 1262
scan_segment) [active](#) on page 1292
(test signal) [active](#) on page 1320

inside_inst

```
inside_inst hier_instance
```

Read-only `opcg_trigger` attribute. Returns the name of the hierachal instance in which the OPCG trigger macro must be inserted. This was specified using the `-inside` option of the `define_dft opcg_trigger` command.

Related Information

Set by this constraint: [define_dft opcg_trigger](#)

osc_source

`osc_source osc_source`

Read-only [opcg_trigger](#) attribute. Returns the name of the oscillator source, which was specified using the `-osc_source` option of the `define_dft opcg_trigger` command.

Related Information

Set by this constraint: [define_dft opcg_trigger](#)

pin

`pin {pin|port}`

Read-only [opcg_trigger](#) attribute. Returns the name of the driving pin or port of the trigger signal, which was specified using the `-pin` option of the `define_dft opcg_trigger` command.

Related Information

Set by this constraint: [define_dft opcg_trigger](#)

Related attributes: (jtag_port) pin

 (osc_source) pin on page 1250

scan_clock

`scan_clock test_clock`

Read-only [opcg_trigger](#) attribute. Specifies the test clock that must drive the flops inside the trigger macro during full scan mode.

Related Information

Set by this constraint: [define_dft opcg_trigger](#)

Related command: [define_dft test_clock](#)

Osc Source Attributes

Contain information about the oscillator source clocks defined by the user. These attributes are read-only attributes, so you cannot set their values.

- To get an `osc_source` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/opcg -osc_source osc_source]
```

Note: These attributes are located at:

```
/designs/design/dft/opcg/osc_sources
```

max_input_period

`max_input_period float`

Read-only `osc_source` attribute. Returns the maximum period of the input clock of the PLL, which was specified using the `-max_input_period` option of the `define_dft osc_source` command.

Related Information

Set by this constraint:

[define_dft_osc_source](#)

Affects these commands

[define_dft_opcg_domain](#)

[define_dft_opcg_mode](#)

[define_dft_opcg_trigger](#)

max_output_period

`max_output_period float`

Read-only `osc_source` attribute. Returns the maximum period of the output clock of the PLL, which was specified using the `-max_output_period` option of the `define_dft osc_source` command.

Related Information

Set by this constraint:

[define_dft_osc_source](#)

Affects these commands

[define_dft_opcg_domain](#)

[define_dft_opcg_mode](#)

[define_dft_opcg_trigger](#)

min_input_period

min_input_period float

Read-only osc_source attribute. Returns the minimum period of the input clock of the PLL, which was specified using the `-min_input_period` option of the `define_dft osc_source` command.

Related Information

Set by this constraint:	define_dft_osc_source
Affects these commands	define_dft_opcg_domain define_dft_opcg_mode define_dft_opcg_trigger

min_output_period

min_output_period float

Read-only osc_source attribute. Returns the minimum period of the output clock of the PLL, which was specified using the `-min_output_period` option of the `define_dft osc_source` command.

Related Information

Set by this constraint:	define_dft_osc_source
Affects these commands	define_dft_opcg_domain define_dft_opcg_mode define_dft_opcg_trigger

pin

`pin pin`

Read-only `osc_source` attribute. Returns the output pin of the PLL, which was specified using the `-pin` option of the `define_dft osc_source` command.

Related Information

Set by this constraint:	define_dft osc_source
Affects these commands	define_dft opcg_domain define_dft opcg_mode define_dft opcg_trigger
Related attributes:	(jtag_port) pin (opcg_trigger) pin on page 1247

ref_clock_pin

`ref_clock_pin {pin|port}`

Read-only `osc_source` attribute. Returns the reference (input) clock pin or port for the PLL, which was specified using the `-ref_clock_pin` option of the `define_dft osc_source` command.

Related Information

Set by this constraint:	define_dft osc_source
Affects these commands	define_dft opcg_domain define_dft opcg_mode define_dft opcg_trigger

Osc Source Reference Attributes

Contain information about the oscillator sources associated with the OPCG modes. These attributes are read-only attributes, so you cannot set their values.

- To get an `osc_source_reference` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/opcg -osc_source_reference name]
```

Note: These attributes are located at

```
/designs/design/dft/opcg/opcg_modes/mode/osc_source_references
```

osc_source_period

`osc_source_period float`

Read-only `osc_source_reference` attribute. Returns the period of the oscillator source, which was specified using the `-osc_source_parameters` option of the `define_dft opcg_mode` command.

Related Information

Set by this constraint: [define_dft opcg_mode](#)

ref_clk_period

`ref_clk_period float`

Read-only `osc_source_reference` attribute. Returns the period of the reference clock, which was specified using the `-osc_source_parameters` option of the `define_dft opcg_mode` command.

Related Information

Set by this constraint: [define_dft opcg_mode](#)

Actual Scan Chain Attributes

Contain information about the final scan chains connected in the specified design. These attributes are read-only attributes, so you cannot set their values.

- To get an `actual_scan_chain` attribute value, type

```
get_attribute attribute_name \
[find /des*/design -actual_scan_chain chain]
```

Note: These attributes are located at:

```
/designs/design/dft/report/actual_scan_chains
```

analyzed

```
analyzed {true | false}
```

Read-only `actual_scan_chain` attribute. Indicates that the connectivity of an existing scan chain—a scan chain created in a previous RTL Compiler session—was analyzed in the current session.

Related Information

Set by this command:

[define dft scan_chain](#)

compressed

```
compressed {true | false}
```

Read-only `actual_scan_chain` attribute. Identifies if test compression was applied to this chain.

Note: Although this attribute is writable, you are expected not to change this attribute value.

Related Information

Set by this command:

[compress_scan_chains](#)

Related attributes:

[dft_mask_clk](#) on page 1314

[dft_compression_signal](#) on page 1323

connected_shift_enable

`connected_shift_enable {true | false}`

Read-only actual_scan_chain attribute. Indicates if the chain consists exclusively of a preserved or abstract segment whose shift-enable signal was either internally generated or driven by a non shift-enable test signal. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Set by this command:	connect_scan_chains
Related attributes:	(<u>actual_scan_chain</u>) shift_enable on page 1260
	(<u>actual_scan_segment</u>) connected_shift_enable on page 1264
	(<u>scan_segment</u>) connected_shift_enable on page 1294

dft hookup_pin_sdi

`dft hookup_pin_sdi {pin|port}`

Read-only actual_scan_chain attribute. Returns the pin or port used by the tool to make the scan data input connection to the core logic.

Example

```
rc:/> get_attr dft hookup_pin_sdi DFT_chain_1  
/designs/test/ports_in/in[0]
```

Related Information

Affected by these commands:	define_dft_abstract_segment define_dft_scan_chain
-----------------------------	--

dft hookup pin sdo

`dft hookup pin sdo {pin|port}`

Read-only actual scan chain attribute. Returns the pin or port used by the tool to make the the scan data output connection from the core logic.

Example

```
rc:/> get_attr dft hookup pin sdo DFT_chain_1  
/designs/test/ports_out/out[0]
```

Related Information

Affected by these commands: [define dft abstract segment](#)
 [define dft scan chain](#)

domain

`domain string`

Read-only actual scan chain attribute. Returns the DFT clock domain associated with the tool-created scan chain.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by these commands: [connect scan chains](#)
 [define dft scan chain](#)

Related attribute: (scan_chain) [domain](#) on page 1286

edge

`edge {rise | fall | any}`

Read-only actual scan chain attribute. Returns the edge of the DFT clock domain associated with the tool-created scan chain.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by these commands: [connect_scan_chains](#)

[define_dft_scan_chain](#)

Related attribute: (scan_chain) [edge](#) on page 1287

elements

`elements string`

Read-only [actual_scan_chain](#) attribute. Returns a Tcl list of the elements in the tool-created scan chain.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by these commands: [connect_scan_chains](#)

Related attributes: (actual_scan_segment) [elements](#) on page 1268

(scan_segment) [elements](#) on page 1300

has_opcg_segments

`has_opcg_segments {false | true}`

Read-only [actual_scan_chain](#) attribute. Indicates whether the actual scan chain has OPCG segments prepended to it.

Related Information

Set by this command: [connect_opcg_segments](#)

head_lockup

`head_lockup string`

Read-only `actual_scan_chain` attribute. Returns the instance of the head lockup element inserted at the head of the actual scan chain during scan connection. A head lockup is inserted when you set the `dft_capture_clock_edge_for_head_of_scan_chains` attribute to `leading`.

Related Information

[Connecting the Scan Chains in Design for Test in Encounter RTL Compiler](#).

Set by this command: [connect_scan_chains](#)

mode_name

`mode_name string`

Read-only `actual_scan_chain` attribute. Returns the configuration mode to which the scan chain belongs.

Related Information

[Defining Scan Chain Configuration Modes in Design for Test in Encounter RTL Compiler](#)

Set by these commands: [concat_scan_chains](#)
 [connect_scan_chains](#)

other_clocks

`other_clocks string`

Read-only `actual_scan_chain` attribute. Returns a Tcl list containing the other clock pins of the chain and their active values.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by this command: [connect_scan_chains](#)

Related attributes: (actual_scan_segment) [other_clocks](#) on page 1268
(scan_segment) [other_clocks](#) on page 1301

power_domain

`power_domain domain`

Read-only [actual_scan_chain](#) attribute. Returns the list of power domains to which this scan chain belongs.

Related Information

Set by this command: [connect_scan_chains](#)

Related command: [report_dft_chains](#)

reg_count

`reg_count integer`

Read-only [actual_scan_chain](#) attribute. Returns the number of flops in the tool-created scan chain.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by this command: [connect_scan_chains](#)

Related attributes: (actual_scan_segment) [reg_count](#) on page 1270

(scan_segment) [reg_count](#) on page 1302

(violation) [reg_count](#) on page 1281

scan_clock_a

`scan_clock_a string`

Read-only [actual_scan_chain](#) attribute. Returns the path to the scan clock a (signal) of the scan chain.

Related Information

Set by this command:	connect_scan_chains
Related attributes:	(actual_scan_segment) scan_clock_a on page 1271
	(scan_chain) scan_clock_a on page 1288
	(scan_segment) scan_clock_a on page 1303

scan_clock_b

`scan_clock_b string`

Read-only [actual_scan_chain](#) attribute. Returns the path to the scan clock b (signal) of the scan chain.

Related Information

Set by this command:	connect_scan_chains
Related attributes:	(actual_scan_segment) scan_clock_b on page 1271
	(scan_chain) scan_clock_b on page 1288
	(scan_segment) scan_clock_b on page 1303

scan_in

`scan_in string`

Read-only [actual_scan_chain](#) attribute. Returns the scan-data input pin associated with the tool-created scan chain.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by these commands:	connect_scan_chains define_dft_scan_chain
Related attributes:	(actual_scan_segment) scan_in on page 1271 (scan_chain) scan_in on page 1289

([scan_segment](#)) [scan_in](#) on page 1304

scan_out

`scan_out string`

Read-only [actual_scan_chain](#) attribute. Returns the scan-data output pin associated with the tool-created scan chain.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by these commands:

[connect_scan_chains](#)

[define_dft_scan_chain](#)

Related attributes:

([actual_scan_segment](#)) [scan_out](#) on page 1272

([scan_chain](#)) [scan_out](#) on page 1289

([scan_segment](#)) [scan_out](#) on page 1304

sdi_compression_signal

`sdi_compression_signal string`

Read-only [actual_scan_chain](#) attribute. Indicates whether the scan data input pin of this chain is shared with the mask enable signal. If the attribute returns `mask_enable`, the scan data input pin of this chain is shared with the mask enable signal.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by this command:

[compress_scan_chains](#)

Related attribute:

([scan_chain](#)) [sdi_compression_signal](#) on page 1289

shared_output

`shared_output {true | false}`

Read-only actual_scan_chain attribute. Indicates if the scan-data output port of this scan chain is shared with a functional port.

Related Information

Set by these commands: [connect_scan_chains](#)
[define_dft_scan_chain](#)

Related attribute: (scan_chain) shared_output on page 1290

shared_select

`shared_select string`

Read-only actual_scan_chain attribute. Returns the control test signal for the mux of the shared scan data output port.

Related Information

Set by these commands: [connect_scan_chains](#)
[define_dft_scan_chain](#)

Related attribute: (scan_chain) shared_select on page 1290

shift_enable

`shift_enable string`

Read-only actual_scan_chain attribute. Returns the chain-specific shift-enable port or pin for the `muxed_scan` scan style. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Note: This attribute has no value if the `connected_shift_enable` attribute of this chain is true.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by these commands:

[connect_scan_chains](#)

[define_dft_scan_chain](#)

Related attributes:

(actual_scan_chain) [connected_shift_enable](#) on page 1253

(actual_scan_segment) [shift_enable](#) on page 1273

(scan_chain) [shift_enable](#) on page 1290

(scan_segment) [shift_enable](#) on page 1305

terminal_lockup

`terminal_lockup string`

Read-only [actual_scan_chain](#) attribute. Returns the instance of the terminal lockup element in the scan chain.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by these commands:

[connect_scan_chains](#)

[define_dft_scan_chain](#)

Related attribute:

(scan_chain) [terminal_lockup](#) on page 1291

Actual Scan Segment Attributes

Contain information about the final scan segments connected in the specified design. These attributes are read-only attributes, so you cannot set their values.

- To get an `actual_scan_segment` attribute value, type

```
get_attribute attribute_name \
[find /des*/design -actual_scan_segment segment]
```

Note: These attributes are located at:

```
/*/dft/report/actual_scan_segments
```

active

```
active {low | high}
```

Read-only `actual_scan_segment` attribute. Returns the active value of the shift-enable port.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by this command:	<u>connect_scan_chains</u>
Related attributes:	(<code>scan_segment</code>) <u>active</u> on page 1292
	(<code>test signal</code>) <u>active</u> on page 1320

Attribute Reference for Encounter RTL Compiler

Design For Test—Actual Scan Segment Attributes

clock

`clock string`

Read-only actual_scan_segment attribute. Returns the clock port driving the flip-flops at the head (shift-in position) of this segment.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by this command: [connect_scan_chains](#)

Related attribute: (scan_segment) [clock](#) on page 1292

clock_edge

`clock_edge {fall | rise}`

Read-only actual_scan_segment attribute. Returns the active edge of the clock driving the flip-flops at the head (shift-in position) of this segment.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by this command: [connect_scan_chains](#)

Related attribute: (scan_segment) [clock_edge](#) on page 1293

connected_scan_clock_a

connected_scan_clock_a {true | false}

Read-only actual_scan_segment attribute. Indicates if the scan clock a port of the module boundary is driven by external logic (preconnected) or if the scan clock a signal is internally generated within the module boundary.

Related Information

Set by this command: [connect_scan_chains](#)

Related attribute: (scan_segment) [connected_scan_clock_a](#) on page 1293

connected_scan_clock_b

connected_scan_clock_b {true | false}

Read-only actual_scan_segment attribute. Indicates if the scan clock b port of the module boundary is driven by external logic (preconnected) or if the scan clock b signal is internally generated within the module boundary.

Related Information

Set by this command: [connect_scan_chains](#)

Related attribute: (scan_segment) [connected_scan_clock_b](#) on page 1294

connected_shift_enable

connected_shift_enable {true | false}

Read-only actual_scan_segment attribute. Indicates if the shift enable port of the module boundary is driven by external logic (preconnected) or if the shift enable signal is internally generated within the module boundary.

Note: This applies only to abstract segments and preserved segments. For other types of segments this attribute has no value.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

- Set by this command: [connect_scan_chains](#)
- Related attribute: (actual_scan_chain) [connected_shift_enable](#) on page 1253
(scan_segment) [connected_shift_enable](#) on page 1294

core_wrapper

core_wrapper {true | false}

Read-write [actual_scan_segment](#) attribute. Indicates whether the segment was created for a core wrapper cell.

Related Information

- Set by this command: [connect_scan_chains](#)
- Related attribute: (scan_segment) [core_wrapper](#) on page 1294

dft_tail_test_clock

dft_tail_test_clock string

Read-only [actual_scan_segment](#) attribute. Returns the top-level clock object that corresponds to the clock port driving the flip-flops at the tail (shift-out position) of this segment.

Note: This applies only to abstract segments. For other types of segments this attribute has no value.

Example

The following example shows the difference between the `tail_clock` and the `dft_tail_test_clock` attributes. The first attribute returns the clock pin on the blackbox module, while the second returns the DFT test clock object.

```
rc:/designs/test/dft/actual_scan_segments> get_att tail_clock abstract1
/designs/test/instances_hier/u_core/pins_in/clk1
rc:/designs/test/dft/actual_scan_segments> get_att dft_tail_test_clock abstract1
/designs/test/dft/test_clock_domains/clk1/clk1
```

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by these commands: [connect_scan_chains](#)
[check_dft_rules](#)

Related attribute: (scan_segment) [dft_tail_test_clock](#) on page 1298

dft_tail_test_clock_edge

`dft_tail_test_clock_edge {rise | fall}`

Read-only actual_scan_segment attribute. Returns the active edge of the top-level clock object that corresponds to the clock port driving the flip-flops at the tail (shift-out position) of this segment.

Note: This applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by these commands: [connect_scan_chains](#)
[check_dft_rules](#)

Related attribute: (scan_segment) [dft_tail_test_clock_edge](#) on page 1298

dft_test_clock

`dft_test_clock string`

Read-only actual_scan_segment attribute. Returns the top-level clock object that corresponds to the clock port driving the flip-flops at the head (shift-in position) of this segment.

Note: This applies only to abstract segments. For other types of segments this attribute has no value.

Attribute Reference for Encounter RTL Compiler

Design For Test—Actual Scan Segment Attributes

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by these commands:

[connect_scan_chains](#)

[check_dft_rules](#)

Related attributes:

(instance) [dft_test_clock](#) on page 1137

(scan_segment) [dft_test_clock](#) on page 1299

dft_test_clock_edge

`dft_test_clock_edge {rise | fall}`

Read-only actual_scan_segment attribute. Returns the active edge of the top-level clock object that corresponds to the clock port driving the flip-flops at the head (shift-in position) of this segment.

Note: This applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by these commands:

[connect_scan_chains](#)

[check_dft_rules](#)

Related attributes:

(instance) [dft_test_clock_edge](#) on page 1137

(scan_segment) [dft_test_clock_edge](#) on page 1299

elements

`elements string`

Read-only actual_scan_segment attribute. Returns a Tcl list of the elements in the tool-created scan segment.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by this command: [connect_scan_chains](#)

Related attributes: [\(scan_chain\) elements](#) on page 1255

[\(scan_segment\) elements](#) on page 1300

instance

`instance string`

Read-only actual_scan_segment attribute. Returns the instance name of the module for which the abstract segment was defined.

Related Information

Set by this command: [connect_scan_chains](#)

Related attribute: [\(scan_segment\) instance](#) on page 1301

other_clocks

`other_clocks string`

Read-only actual_scan_segment attribute. Returns a Tcl list containing the other clock pins of the segment and their active values.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

- Set by this command: [connect_scan_chains](#)
Related attributes: (actual_scan_chain) [other_clocks](#) on page 1256
(scan_segment) [other_clocks](#) on page 1301

power_domain

`power_domain domain`

Read-only [actual_scan_segment](#) attribute. Returns the power domain of this scan segment.

Note: This attribute applies only to segments inserted by the `insert_dft wrapper_cell` command that also have the following `actual_scan_segment` attribute settings:

```
core_wrapper = true
type = preserved
```

Example

The following command returns the power domain for segment DFT_segment_1.

```
rc:/designs/top_1_port_pnr> get_attr power_domain \
[find /des*/top* -actual_scan_segment DFT_segment_1]
/designs/top_1_port_pnr/power/power_domains/pd2
```

Related Information

[Inserting Core Wrapper Logic in Design for Test in Encounter RTL Compiler](#)

- Set by this command: [insert_dft wrapper_cell](#)
Related attributes: (actual_scan_segment) [core_wrapper](#) on page 1265
(actual_scan_segment) [type](#) on page 1275

reg_count

`reg_count integer`

Read-only actual_scan_segment attribute. Returns the number of flops in the tool-created scan segment.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by this command: [connect_scan_chains](#)

Related attributes: [\(actual_scan_chain\) reg_count](#) on page 1257

[\(scan_segment\) reg_count](#) on page 1302

[\(violation\) reg_count](#) on page 1281

reorderable

`reorderable {true | false}`

Read-only actual_scan_segment attribute. Indicates if the preserved segment is reorderable for scanDEF purposes.

Note: This attribute applies only to preserved segments. For other types of segments this attribute has no value.

Related Information

Set by this command: [connect_scan_chains](#)

Related attribute: [\(scan_segment\) reorderable](#) on page 1303

scan_clock_a

`scan_clock_a string`

Read-only actual_scan_segment attribute. Returns the scan clock A pin for an abstract segment.

Related Information

Set by this command: [connect_scan_chains](#)

Related attributes:
(actual_scan_chain) [scan_clock_a](#) on page 1257
(scan_chain) [scan_clock_a](#) on page 1288
(scan_segment) [scan_clock_a](#) on page 1303

scan_clock_b

`scan_clock_b string`

Read-only actual_scan_segment attribute. Returns the scan clock B pin for an abstract segment.

Related Information

Set by this command: [connect_scan_chains](#)

Related attributes:
(actual_scan_chain) [scan_clock_b](#) on page 1258
(scan_chain) [scan_clock_b](#) on page 1271
(scan_segment) [scan_clock_b](#) on page 1303

scan_in

`scan_in string`

Read-only actual_scan_segment attribute. Returns the scan-data input pin associated with the tool-created scan segment.

Note: This applies only to abstract segments and preserved segments. For other types of segments this attribute has no value.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by this command:

[connect_scan_chains](#)

Related attributes:

(actual_scan_chain) [scan_in](#) on page 1258

(scan_chain) [scan_in](#) on page 1289

(scan_segment) [scan_in](#) on page 1304

scan_out

`scan_out string`

Read-only [actual_scan_segment](#) attribute. Returns the scan-data output pin associated with the tool-created scan segment.

Note: This applies only to abstract segments and preserved segments. For other types of segments this attribute has no value.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by this command:

[connect_scan_chains](#)

Related attributes:

(actual_scan_chain) [scan_out](#) on page 1259

(scan_chain) [scan_out](#) on page 1289

(scan_segment) [scan_out](#) on page 1304

shift_enable

`shift_enable string`

Read-only `actual_scan_segment` attribute. Returns the segment-specific shift-enable port or pin for the `muxed_scan` scan style.

Note: This applies only to abstract segments and preserved segments. For other types of segments this attribute has no value.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by this command: [connect_scan_chains](#)

Related attributes: [\(actual_scan_chain\) shift_enable](#) on page 1260

[\(scan_chain\) shift_enable](#) on page 1290

[\(scan_segment\) shift_enable](#) on page 1305

skew_safe

`skew_safe {true | false}`

Read-only `actual_scan_segment` attribute. Indicates if the segment has a data lockup element connected at the end of its scan chain.

Note: This applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by this command: [connect_scan_chains](#)

Related attribute: [\(scan_segment\) skew_safe](#) on page 1305

tail_clock

`tail_clock string`

Read-only actual_scan_segment attribute. Returns the clock port driving the flip-flops at the tail (shift-out position) of this segment.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by this command: [connect_scan_chains](#)

Related attribute: (scan_segment) [tail_clock](#) on page 1305

tail_clock_edge

`tail_clock_edge {rise | fall}`

Read-only actual_scan_segment attribute. Returns the active edge of the clock driving the flip-flops at the tail (shift-out position) of this segment.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by this command: [connect_scan_chains](#)

Related attribute: (scan_segment) [tail_clock_edge](#) on page 1306

type

```
type {abstract | fixed | floating | preserve | shift_register}
```

Read-only actual scan segment attribute. Returns the type of the tool-created scan segment. This attribute can have the following values:

abstract	Indicates that this segment is an abstract segment (that is, embedded in a blackbox or logic abstract module, or defined for a libcell timing model).
fixed	Indicates that the elements in this segment are connected in the user-specified order.
floating	Indicates that the elements in this segment are not necessarily connected in the user-specified order.
preserve	Indicates that the order of the presumed connected elements was preserved.
shift_register	Indicates that this segment is a shift register. This implies that the order of the elements in the segment is fixed.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by this command: [connect_scan_chains](#)

Related attributes: [\(scan_segment\) type](#) on page 1307

Violations Attributes

Contain information about the violations reported by the DFT rule checker. These attributes are read-only attributes, so you cannot set their values.

- To get a violation attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft/report -violation violation]
```

Note: These attributes are located at:

```
/designs/design/dft/report/violations
```

description

```
description string
```

Read-only violation attribute. Returns the description of the violation. Possible values are listed in the table below.

Asynchronous Signal Violations

[ASYNC-01]	detected a loop while tracing async signal
[ASYNC-02]	async signal driven to a constant active value, possibly due to a polarity conflict
[ASYNC-03]	async signal has no driver
[ASYNC-04]	internal or gated async signal
[ASYNC-05]	async signal driven by a sequential element
[ASYNC-06]	async signal has multiple drivers
[ASYNC-07]	async signal also used as a clock signal
[ASYNC-08]	misc. async signal violation
[ASYNC-09]	async signal is not controllable
[ASYNC-10]	async signal driven by a primary input (not defined as a test_mode signal)

Abstract Segment Test Mode Signal Violations

[ABS-TM-01]	detected a loop while tracing abstract segment test mode signal
[ABS-TM-02]	abstract segment test mode signal driven to a constant active value, possibly due to a polarity conflict

Attribute Reference for Encounter RTL Compiler

Design For Test—Violations Attributes

[ABS-TM-03]	abstract segment test mode signal has no driver
[ABS-TM-04]	internal or gated abstract segment test mode signal
[ABS-TM-05]	abstract segment test mode signal driven by a sequential element
[ABS-TM-06]	abstract segment test mode signal has multiple drivers
[ABS-TM-07]	conflict in abstract segment test mode signal driver which is driven by test signal of opposite polarity
[ABS-TM-08]	misc. abstract segment test mode signal violation
[ABS-TM-09]	Abstract Segment Test Mode signal is not controllable

Clock and Data Race Violations

[RACE-01]	test clock either directly or indirectly drives a non-clock input pin of a sequential instance.
-------------	---

Clock Signal Violations

[CLOCK-01]	detected a loop while tracing clock signal
[CLOCK-02]	clock signal driven to a constant value
[CLOCK-03]	controlled to opposite of required off-state of clock signal
[CLOCK-04]	clock signal has no driver
[CLOCK-05]	internal or gated clock signal
[CLOCK-06]	clock signal driven by a sequential element
[CLOCK-07]	clock signal has multiple drivers
[CLOCK-08]	clock signal driven by a BlackBox (unresolved) element
[CLOCK-09]	misc. clock signal violation
[CLOCK-10]	clock signal driven by a primary input (not defined as a test clock signal)

Same Asynchronous Set and Reset Violations

[RACE-02]	test signal drives both the asynchronous set and reset pins of same register.
-------------	---

Shift Register Violations

[SHIFTREG-01]	Synchronous pins of <i>flop</i> are not appropriately controlled in test mode.
-----------------	--

Attribute Reference for Encounter RTL Compiler

Design For Test—Violations Attributes

[SHIFTREG-02]	Output of <i>flop</i> driven to a constant value 0, possibly due to settings of synchronous pins in test mode.
[SHIFTREG-03]	Output of <i>flop</i> driven to a constant value 1, possibly due to settings of synchronous pins in test mode.
[SHIFTREG-04]	Shift register segment not connected properly, could not trace back to a register from <i>flop</i> .
[SHIFTREG-05]	Could not trace back to a register from <i>flop</i> .

Tristate Net Contention Violations

[TRISTATE_NET-01]	tristate net <i>netName</i> connected to pin <i>pinName</i> potentially driven by conflicting values.
---------------------	---

X-Source Violations

[BBOX-01]	output pin of unresolved instance or timing model drives an input pin of a sequential instance.
-------------	---

Example

In this example the clock signal is driven by a sequential element. Because the clock driver is not a user-defined test-clock pin—specified as being controllable—the clock driver is considered uncontrollable.

```
rc:/> get_att description /design/top/dft/report/violations/vid_1_async  
[CLOCK-06] clock signal driven by a sequential element
```

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [check_dft_rules](#)

Affects this command: [report_dft_violations](#)

Related attributes: (instance) [dftViolation](#) on page 1138

endpoints

`endpoints string`

Read-only [violation](#) attribute. Returns a Tcl list of test-mode pins and ports of abstraction models that are affected by the same test-mode violation.

Note: This attribute applies only to abstract segment violations.

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [check_dft_rules](#)

Affects this command: [report_dft_violations](#)

file_name

`file_name string`

Read-only violation attribute. Returns the name of the file in which the problem was detected. You need to set the `hdl_track_filename_row_col` root attribute to `true` before you run the DFT rule checker, otherwise an empty string is returned.

Example

```
rc:/> get_att file_name [find /designs/top -violation vid_1_async]  
test10.v
```

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [check_dft_rules](#)

Affects this command: [report_dft_violations](#)

Affected by this attribute: [hdl_track_filename_row_col](#) on page 257

fixed

`fixed {true | false}`

Read-only violation attribute. Indicates whether the violation has been fixed with the `fix_dft_violations` command.

This attribute value is only set to `true` when you run the `fix_dft_violations` command with the `-dont_check_dft_rules` option. If you omit this option, the entire violations directory is updated and only true violations are kept.

Attribute Reference for Encounter RTL Compiler

Design For Test—Violations Attributes

Note: You can also fix a violation using `insert_dft test_point` or `insert_dft user_test_point` commands. However, these commands have no information about the violation ID that is being fixed and therefore they do not affect the value of the attribute.

Example

```
rc:/> get_att fixed [find /designs/top -violation vid_1_async]
false
```

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [check_dft_rules](#)

Related attributes: (instance) [dft_status](#) on page 1136

id

`id integer`

Read-only [violation](#) attribute. Returns the violation ID number given by `check_dft_rules`.

Example

```
rc:/> get_att id /design/top/dft/report/violations/vid_1_async
1
```

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [check_dft_rules](#)

Affects this command: [report_dft_violations](#)

Related attributes: (instance) [dftViolation](#) on page 1138

line_number

```
line_number integer
```

Read-only violation attribute. Returns the line number at which the problem was detected. You need to set the `hdl_track_filename_row_col` root attribute to `true` before you run the DFT rule checker, otherwise the value is set to 0.

Example

In this example, the `hdl_track_filename_row_col` root attribute was not set to `true`.

```
rc:/> get_att line_number [find /designs/top -violation vid_1_async]  
0
```

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler](#).

Set by this command: [check_dft_rules](#)

Affects this command: [report_dft_violations](#)

Affected by this attribute: [hdl_track_filename_row_col](#) on page 257

reg_count

```
reg_count integer
```

Read-only violation attribute. Returns the number of registers this violation affects. If the violation also affects abstract segments, the count also includes the length of the abstract segments.

Example

In this example four registers are affected by this violation.

```
rc:/> get_att reg_count [find /designs/top -violation vid_1_async]  
4
```

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [check_dft_rules](#)

Affects this command: [report_dft_violations](#)

Related attributes:

- (actual_scan_chain) [reg_count](#) on page 1257
- (actual_scan_segment) [reg_count](#) on page 1270
- (scan_segment) [reg_count](#) on page 1302

registers

`registers instance_list`

Read-only [violation](#) attribute. Returns a list of the instance names of the registers affected by this violation.

Example

```
rc:> get_att registers [find /designs/top -violation vid_1_async]
{/designs/top/instances_seq/out2_reg[0]}{/designs/top/instances_seq/out2_reg[1]}
{/designs/top/instances_seq/out2_reg[2]}{/designs/top/instances_seq/out2_reg[3]}
```

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [check_dft_rules](#)

Affects this command: [report_dft_violations](#)

root_node

```
root_node {pin | port | bus}
```

Read-only violation attribute. Returns the pin, port or bus at which the violation originates.

Example

In this example, the violation is caused because the clock signal is driven by a sequential element. The `root_node` attribute returns the clock driver which is the Q pin of the sequential element.

```
rc:/> get_att root_node [find /designs/top -violation vid_1_clock]
/designs/top/instances_seq/divClk_reg/pins_out/q
```

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [check_dft_rules](#)

segments

```
segments list
```

Read-only violation attribute. Returns a list of the abstract segments affected by this violation.

Example

```
rc:/> get_att segments [find /designs/test -violation vid_0_abs]
/designs/test/dft/scan_segments/core1_1 /designs/test/dft/scan_segments/core1_2
```

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [check_dft_rules](#)

Affects this command: [report_dft_violations](#)

tristate_net_drivers

```
tristate_net_drivers {pin|port|bus}
```

Read-only violation attribute. Returns the tristate net driver pins where the violation originates.

tristate_net_load

```
tristate_net_load {pin|port|bus}
```

Read-only violation attribute. Returns the load pin for tristate net where the violation originates.

type

```
type {async | clock | abstract segment test mode | shift register | tristate net |  
      race | xsource }
```

Read-only violation attribute. Returns the type of the violation.

Example

```
rc:/designs/top> get_att type [find /designs/top/dft -violation vid_1]\  
clock
```

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command: [check_dft_rules -advanced](#)

Affects this command: [report dft violations](#)

Related attributes: (instance) [dft violation](#) on page 1138

Scan Chain Attributes

Contain information about the user-defined scan chains.

- To set a `scan_chain` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -scan_chain chain]
```

- To get a `scan_chain` attribute value, type

```
get_attribute attribute_name \
[find /des*/design -scan_chain chain]
```

Note: These attributes are located at:

`/designs/design/dft/scan_chains`.

body

`body string`

Read-only `scan_chain` attribute. Returns the name of the body segment of the user-specified scan chain.

Related Information

Set by this constraint: [define dft scan_chain](#)

complete

`complete {true | false}`

Read-only `scan_chain` attribute. Indicates if the user-defined scan chain was completely user-specified (true) or was completed by analysis (false).

If a scan chain is marked `complete`, it means that all its components were specified through options of the `define_dft scan_chain` constraint, that is, the scan-data input and output, and the head, tail, and body segments were all specified by the user.

Related Information

Set by this constraint: [define dft scan_chain](#)

dft hookup pin_sdi

`dft hookup pin_sdi {pin|port}`

Read-only scan_chain attribute. Returns the pin or port used by the tool to make the scan data input connection to the core logic.

Example

```
rc:/> get_attr dft hookup pin_sdi DFT_chain_1  
/designs/test/ports_in/in[0]
```

Related Information

Affected by these commands: [define dft abstract segment](#)
 [define dft scan chain](#)

dft hookup pin_sdo

`dft hookup pin_sdo {pin|port}`

Read-only scan_chain attribute. Returns the pin or port used by the tool to make the the scan data output connection from the core logic.

Example

```
rc:/> get_attr dft hookup pin_sdo DFT_chain_1  
/designs/test/ports_out/out[1]
```

Related Information

Affected by these commands: [define dft abstract segment](#)
 [define dft scan chain](#)

domain

`domain string`

Read-only scan_chain attribute. Returns the DFT clock domain associated with the user-specified scan chain.

Related Information

Set by this constraint: [define dft scan_chain](#)
Related attribute: (actual_scan_chain) [domain](#) on page 1254

edge

`edge {rise | fall | any}`

Read-only [scan_chain](#) attribute. Returns the edge of the DFT clock domain associated with the user-specified scan chain.

Related Information

Set by this constraint: [define dft scan_chain](#)
Related attribute: (actual_scan_chain) [edge](#) on page 1254

head

`head string`

Read-only [scan_chain](#) attribute. Returns the name of the head segment of the user-specified scan chain.

Related Information

Set by this constraint: [define dft scan_chain](#)

max_length

`max_length integer`

Read-only [scan_chain](#) attribute. Returns the maximum length that was allowed for this user-specified scan chain. This length might be slightly different than the actual length.

Related Information

Set by this constraint: [define dft scan_chain](#)

scan_clock_a

`scan_clock_a string`

Read-only `scan_chain` attribute. Returns the path to the scan clock a (signal) of the scan chain.

Related Information

Set by this constraint:	define dft scan_chain
Related attributes:	(actual_scan_chain) scan_clock_a on page 1257
	(actual_scan_segment) scan_clock_a on page 1271
	(scan_segment) scan_clock_a on page 1303

scan_clock_b

`scan_clock_b string`

Read-only `scan_chain` attribute. Returns the path to the scan clock b (signal) of the scan chain.

Related Information

Set by this constraint:	define dft scan_chain
Related attributes:	(actual_scan_chain) scan_clock_b on page 1258
	(actual_scan_segment) scan_clock_b on page 1271
	(scan_segment) scan_clock_b on page 1303

scan_in

`scan_in string`

Read-only `scan_chain` attribute. Returns the scan-data input pin of the user-specified scan chain.

Related Information

Set by this constraint:

[define dft scan_chain](#)

Related attributes:

(actual_scan_chain) [scan_in](#) on page 1258

(actual_scan_segment) [scan_in](#) on page 1271

(scan_segment) [scan_in](#) on page 1304

scan_out

`scan_out string`

Read-only `scan_chain` attribute. Returns the scan-data output pin of the user-specified scan chain.

Related Information

Set by this constraint:

[define dft scan_chain](#)

Related attributes:

(actual_scan_chain) [scan_out](#) on page 1259

(actual_scan_segment) [scan_out](#) on page 1272

(scan_segment) [scan_out](#) on page 1304

sdi_compression_signal

`sdi_compression_signal string`

Read-only `scan_chain` attribute. If the attribute returns `mask_enable`, the scan data input pin of this chain is shared with the mask enable signal.

Related Information

Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler

Set by these commands: [compress_scan_chains](#)

Related attribute: (actual_scan_chain) [sdi_compression_signal](#) on page 1259

shared_output

`shared_output {true | false}`

Read-only [scan_chain](#) attribute. Indicates if the scan-data output port of this user-specified scan chain is shared with a functional port.

Related Information

Set by this constraint: [define_dft_scan_chain](#)

Related attribute: (actual_scan_chain) [shared_output](#) on page 1260

shared_select

`shared_select string`

Read-only [scan_chain](#) attribute. Returns the control test signal for the mux of the shared scan data output port.

Related Information

Set by these commands: [define_dft_scan_chain](#)

Related attribute: (actual_scan_chain) [shared_select](#) on page 1260

shift_enable

`shift_enable string`

Read-only [scan_chain](#) attribute. Returns the chain-specific shift-enable port or pin for the muxed_scan scan style. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Set by this constraint:	define dft scan_chain
Related attributes:	(actual_scan_chain) shift_enable on page 1260
	(actual_scan_segment) shift_enable on page 1273
	(scan_segment) shift_enable on page 1305

tail

tail string

Read-only [scan_chain](#) attribute. Returns the name of the tail segment of the user-specified scan chain.

Related Information

Set by this constraint:	define dft scan_chain
-------------------------	---------------------------------------

terminal_lockup

`terminal_lockup {none | level_sensitive | edge_sensitive}`

Read-only [scan_chain](#) attribute. Returns the type of the terminal lockup element inserted at the tail end of the user-specified chain. This attribute can have the following values:

edge-sensitive	Indicates that the terminal lockup element inserted is a flip-flop.
level-sensitive	Indicates that the terminal lockup element inserted is a latch.
none	Indicates that a scan chain does not have a terminal lockup.

Related Information

Set by this constraint:	define dft scan_chain
Related attribute:	(actual_scan_chain) terminal_lockup on page 1261

Scan Segment Attributes

Contain information about the user-defined scan segments.

- To set a `scan_segment` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -scan_segment name]
```

- To get a `scan_segment` attribute value, type

```
get_attribute attribute_name [find /des*/design -scan_segment name]
```

Note: These attributes are located at:

```
/designs/design/dft/scan_segments
```

active

```
active {low | high}
```

Read-only `scan_segment` attribute. Specifies the active value of the shift-enable port at the boundary of the blackbox in which this segment is defined.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Set by this constraint: [define dft abstract_segment](#)

Related attributes: [\(actual_scan_segment\) active](#) on page 1262

[\(test signal\) active](#) on page 1320

clock

```
clock string
```

Read-only `scan_segment` attribute. Returns the clock port driving the flip-flops at the head (shift-in position) of this user-defined scan segment.

Note: This attribute applies only to abstract scan segments. For other types of segments this attribute has no value.

Related Information

Set by this constraint: [define dft abstract_segment](#)

Related attribute: (actual_scan_segment) [clock](#) on page 1263

clock_edge

clock_edge {fall | rise}

Read-only [scan_segment](#) attribute. Returns the active edge of the clock driving the flip-flops at the head (shift-in position) of this user-defined scan segment.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Set by this constraint: [define_dft_abstract_segment](#)

Related attribute: (actual_scan_segment) [clock_edge](#) on page 1263

connected_scan_clock_a

connected_scan_clock_a {true | false}

Read-only [scan_segment](#) attribute. Indicates if the scan clock a port of the module boundary is driven by external logic (preconnected) or if the scan clock a signal is internally generated within the module boundary.

Related Information

Set by this constraint: [define_dft_preserved_segment](#)

Related attribute: (actual_scan_segment) [connected_scan_clock_a](#) on page 1264

connected_scan_clock_b

`connected_scan_clock_b {true | false}`

Read-only scan_segment attribute. Indicates if the scan clock b port of the module boundary is driven by external logic (preconnected) or if the scan clock b signal is internally generated within the module boundary.

Related Information

Set by this constraint: [define_dft_preserved_segment](#)

Related attribute: (actual_scan_segment) [connected_scan_clock_b](#) on page 1264

connected_shift_enable

`connected_shift_enable {true | false}`

Read-only scan_segment attribute. Indicates if the shift enable port of the module boundary is driven by external logic (preconnected) or if the shift enable signal is internally generated within the module boundary.

Note: This attribute applies only to abstract segments and preserved segments. For other types of segments this attribute has no value.

Related Information

Set by these constraints: [define_dft_abstract_segment](#)

[define_dft_preserved_segment](#)

Related attribute: (actual_scan_chain) [connected_shift_enable](#) on page 1253

(actual_scan_segment) [connected_shift_enable](#) on page 1264

core_wrapper

`core_wrapper {true | false}`

Read-write scan_segment attribute. Indicates whether the segment was created for a core wrapper cell.

Attribute Reference for Encounter RTL Compiler

Design For Test—Scan Segment Attributes

Related Information

[Inserting Core Wrapper Logic in Design for Test in Encounter RTL Compiler](#)

Set by this command: [insert_dft wrapper_cell](#)

Related attribute: (active scan segment) [core_wrapper](#) on page 1265

core_wrapper_ports

`core_wrapper_ports {pins|ports}`

Read-write [scan_segment](#) attribute. Specifies the pins or ports associated with this wrapper segment. More than one port can be associated with a shared wrapper segment.

Note: This attribute only applies to segments for which the `core_wrapper` attribute is set to true.

Related Information

[Inserting Core Wrapper Logic in Design for Test in Encounter RTL Compiler](#)

Set by this command: [insert_dft wrapper_cell](#)

core_wrapper_type

`core_wrapper_type {dedicated | shared}`

Read-write [scan_segment](#) attribute. Indicates whether this wrapper segment is shared or dedicated.

Note: This attribute only applies to segments for which the `core_wrapper` attribute is set to true.

Related Information

[Inserting Core Wrapper Logic in Design for Test in Encounter RTL Compiler](#)

Set by this command: [insert_dft wrapper_cell](#)

core_wrapper_usage

```
core_wrapper_usage {input_bounding | output_bounding}
```

Read-write scan_segment attribute. Indicates whether the wrapper segment is used for input or output bounding.

Based on the fanout analysis from an input port or fanin analysis from an output port, the tool identifies which sets of functional flops are used for input bounding (meaning that the wrapper cells will load the test data when WINT=1 to test the core logic), or for output bounding (meaning that the wrapper cell will load the test data when WEXT=1 to provide test data external to the core to test the interconnect and surrounding logic. WINT and WEXT are mutually exclusive signals.

Note: This attribute only applies to segments for which the `core_wrapper` attribute is set to true.

Related Information

[Inserting Core Wrapper Logic in Design for Test in Encounter RTL Compiler](#)

Set by this command: [insert_dft_wrapper_cell](#)

dft_dont_scan

```
dft_dont_scan {false | true}
```

Default: false

Read-write scan_segment attribute. Controls the inclusion of the specified abstract segment into a scan chain. This attribute can have the following values:

false	Allows the abstract segment to be included in a scan chain if it passes the DFT rule checks.
true	Prevents the abstract segment from being included in a scan chain.

You must set this attribute prior to running the `check_dft_rules` command.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Attribute Reference for Encounter RTL Compiler

Design For Test—Scan Segment Attributes

Note: The tool will automatically set this attribute to `true` when the parent of the hierarchical instance for which the abstract segment is defined has its `dft_dont_scan` attribute set to `true`.

Related Information

[Marking Objects not to be Mapped to Scan Flip-Flops in Design for Test in Encounter RTL Compiler.](#)

Affects these commands:

[check_dft_rules](#)

[connect_scan_chains](#)

[synthesize](#)

Related attributes:

(design) [dft_dont_scan](#) on page 1121

(instance) [dft_dont_scan](#) on page 1133

(subdesign) [dft_dont_scan](#) on page 1153

dft_status

`dft_status {Passes DFT Rules | Fails DFT Rules}`

Read-only [scan_segment](#) attribute. Returns the DFT rule check (scan) status of a segment.

Fails DFT Rules Indicates that some flops of the segment failed the DFT rules.

Passes DFT Rules Indicates that all flops of the segment passed the DFT rules.

Note: This attribute has no value if the DFT rule checker has not yet been run.

Related Information

[Running the DFT Rule Checker in Design for Test in Encounter RTL Compiler.](#)

Set by this command:

[check_dft_rules](#)

Related attributes:

(instance) [dft_status](#) on page 1136

dft_tail_test_clock

`dft_tail_test_clock string`

Read-only scan_segment attribute. Returns the top-level clock object that corresponds to the clock port driving the flip-flops at the tail (shift-out position) of this segment.

This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Example

The following example shows the difference between the `tail_clock` and the `dft_tail_test_clock` attributes. The first attribute returns the clock pin on the blackbox module, while the second returns the DFT test clock object.

```
rc:/designs/test/dft/scan_segments> get_att tail_clock abstract1  
/designs/test/instances_hier/u_core/pins_in/clk1  
rc:/designs/test/dft/scan_segments> get_att dft_tail_test_clock abstract1  
/designs/test/dft/test_clock_domains/clk1/clk1
```

Related Information

Set by this command: [check_dft_rules](#)

Related attribute: (actual_scan_segment) [dft_tail_test_clock](#) on page 1265

dft_tail_test_clock_edge

`dft_tail_test_clock_edge {rise | fall}`

Read-only scan_segment attribute. Returns the active edge of the top-level clock object that corresponds to the clock port driving the flip-flops at the tail (shift-out position) of this segment.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Set by this command: [check_dft_rules](#)

Related attribute: (actual_scan_segment) [dft_tail_test_clock_edge](#) on page 1266

dft_test_clock

`dft_test_clock string`

Read-only `scan_segment` attribute. Returns the top-level clock object that corresponds to the clock port driving the flip-flops at the head (shift-in position) of this segment.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Set by this command: [check_dft_rules](#)

Related attributes: (instance) [dft_test_clock](#) on page 1137

(actual_scan_segment) [dft_test_clock](#) on page 1266

dft_test_clock_edge

`dft_test_clock_edge {rise | fall}`

Read-only `scan_segment` attribute. Returns the active edge of the top-level clock object that corresponds to the clock port driving the flip-flops at the head (shift-in position) of this segment.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Set by this command: [check_dft_rules](#)

Related attributes: (instance) [dft_test_clock_edge](#) on page 1137

(actual_scan_segment) [dft_test_clock_edge](#) on page 1267

dftViolation

```
dftViolation {abstractTestmode | clock | asyncSet | asyncReset}  
 #(violationIdNumber)
```

Read-only scan_segment attribute. Returns the type of violation (abstract segment test mode rule violation, clock rule violation or asynch rule violation) for the abstract segment together with the violation ID number given by the `check_dft_rules` command.

This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Note: This attribute has no value if the DFT rule checker has not yet been run, or for segments that are not abstract segments.

Example

```
rc:/> get_att dftViolation /des*/*/*seq/*1  
clock #(0) asyncSet #(1)
```

Related Information

Set by this command: [check_dft_rules](#)

Related attributes: (instance) [dftViolation](#) on page 1138

elements

`elements string`

Read-only scan_segment attribute. Returns a Tcl list of the elements in this user-defined scan segment.

Related Information

Set by these constraints: [define_dft_abstract_segment](#)

[define_dft_fixed_segment](#)

[define_dft_floating_segment](#)

[define_dft_preserved_segment](#)

[define_dft_shift_register_segment](#)

Related attribute: (actual_scan_chain) [elements](#) on page 1255

(actual_scan_segment) [elements](#) on page 1268

instance

`instance string`

Read-only `scan_segment` attribute. Returns the instance name of the module for which the abstract segment was defined.

Related Information

Set by this constraint: [define dft abstract segment](#)

Related attribute: (actual_scan_segment) [instance](#) on page 1268

other_clocks

`other_clocks string`

Read-only `scan_segment` attribute. Returns a Tcl list containing the other clock pins of the segment and their active values.

Related Information

[Reporting on Specific Aspects of Chains or Segments in Design for Test in Encounter RTL Compiler](#)

Set by these constraints: [define dft abstract segment](#)

[define dft fixed segment](#)

[define dft floating segment](#)

[define dft preserved segment](#)

[define dft shift register segment](#)

Related attributes: (actual_scan_chain) [other_clocks](#) on page 1256

(actual_scan_segment) [other_clocks](#) on page 1268

power_domain

`power_domain domain`

Read-only `scan_segment` attribute. Returns the power domain of this scan segment.

Attribute Reference for Encounter RTL Compiler

Design For Test—Scan Segment Attributes

Note: This attribute applies only to segments inserted by the `insert_dft wrapper_cell` command that also have the following `scan_segment` attribute settings:

```
core_wrapper = true  
type = preserved
```

Example

The following command returns the power domain for segment `DFT_segment_1`.

```
rc:/designs/top_1_port_pnr> get_attr power_domain \  
[find /des*/top* -scan_segment DFT_segment_1]  
/designs/top_1_port_pnr/power/power_domains/pd2
```

Related Information

[Inserting Core Wrapper Logic in Design for Test in Encounter RTL Compiler](#)

Set by this command: [insert_dft wrapper_cell](#)

Related attributes: [\(scan_segment\) core_wrapper](#) on page 1294
[\(scan_segment\) type](#) on page 1307

reg_count

```
reg_count integer
```

Read-only `scan_segment` attribute. Returns the number of flops in this user-defined scan segment.

Related Information

Set by these constraints: [define_dft_abstract_segment](#)

[define_dft_fixed_segment](#)

[define_dft_floating_segment](#)

[define_dft_preserved_segment](#)

[define_dft_shift_register_segment](#)

Related attributes: [\(actual_scan_chain\) reg_count](#) on page 1257

[\(actual_scan_segment\) reg_count](#) on page 1270

[\(violation\) reg_count](#) on page 1281

reorderable

```
reorderable {true | false}
```

Read-only scan_segment attribute. Indicates if the preserved segment is reorderable for scanDEF purposes.

Note: This attribute applies only to preserved segments. For other types of segments this attribute has no value.

Related Information

Set by this command: [define dft preserved segment](#)

Related attribute: (actual_scan_segment) [reorderable](#) on page 1270

scan_clock_a

```
scan_clock_a string
```

Read-only scan_segment attribute. Returns the scan clock A pin for an abstract segment.

Related Information

Set by this constraint: [define dft abstract segment](#)

Related attributes: (actual_scan_chain) [scan_clock_a](#) on page 1257

(actual_scan_segment) [scan_clock_a](#) on page 1271

(scan_chain) [scan_clock_a](#) on page 1288

scan_clock_b

```
scan_clock_b string
```

Read-only scan_segment attribute. Returns the scan clock B pin for an abstract segment.

Related Information

Set by this command: [define dft abstract segment](#)

Related attributes: (actual_scan_chain) [scan_clock_b](#) on page 1258

(actual_scan_segment) [scan_clock_b](#) on page 1271

(*scan_chain*) [scan_clock_b](#) on page 1288

scan_in

scan_in string

Read-only [scan_segment](#) attribute. Returns the scan-data input of this user-defined scan segment.

Note: This attribute applies only to abstract segments and preserved segments. For other types of segments this attribute has no value.

Related Information

Set by these constraints:

[define dft abstract segment](#)

[define dft preserved segment](#)

Related attributes:

(*actual_scan_chain*) [scan_in](#) on page 1258

(*actual_scan_segment*) [scan_in](#) on page 1271

(*scan_chain*) [scan_in](#) on page 1289

scan_out

scan_out string

Read-only [scan_segment](#) attribute. Returns the scan-data output of this user-defined scan segment.

Note: This attribute applies only to abstract segments and preserved segments. For other types of segments this attribute has no value.

Related Information

Set by this constraint:

[define dft abstract segment](#)

Related attributes:

(*actual_scan_chain*) [scan_out](#) on page 1259

(*actual_scan_segment*) [scan_out](#) on page 1272

(*scan_chain*) [scan_out](#) on page 1289

shift_enable

`shift_enable string`

Read-only scan_segment attribute. Returns the shift-enable port at the boundary of the blackbox in which this segment is defined.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Set by this constraint: [define dft abstract segment](#)

Related attributes:
 (aactual_scan_chain) shift_enable on page 1260
 (aactual_scan_segment) shift_enable on page 1273
 (scan_chain) shift_enable on page 1290

skew_safe

`skew_safe {true | false}`

Read-only scan_segment attribute. Indicates if this user-defined scan segment has a data lockup element connected at the end of its scan chain.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Set by these constraints: [define dft abstract segment](#)

Related attribute: (aactual_scan_segment) skew_safe on page 1273

tail_clock

`tail_clock string`

Read-only scan_segment attribute. Returns the clock port driving the flip-flops at the tail (shift-out position) of this user-defined scan segment.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Set by this constraint: [define dft abstract segment](#)

Related attribute: (actual_scan_segment) [tail_clock](#) on page 1274

tail_clock_edge

`tail_clock_edge {rise | fall}`

Read-only [scan_segment](#) attribute. Returns the active edge of the clock driving the flip-flops at the tail (shift-out position) of this user-defined scan segment.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Related Information

Set by this constraint: [define dft abstract segment](#)

Related attribute: (actual_scan_segment) [tail_clock_edge](#) on page 1274

test_modes

`test_modes string`

Read-only [scan_segment](#) attribute. Returns a Tcl list containing the test mode pins of the segment and their active values.

Note: This attribute applies only to abstract segments. For other types of segments this attribute has no value.

Example

```
rc:/> get_att test_modes abstract1  
/designs/test/instances_hier/u_core/pins_in/RST high
```

Related Information

Set by this constraint: [define dft abstract segment](#)

Related attribute: (actual_scan_segment) [tail_clock_edge](#) on page 1274

Attribute Reference for Encounter RTL Compiler

Design For Test—Scan Segment Attributes

type

```
type {abstract | fixed | floating | preserve | shift_register}
```

Read-only scan_segment attribute. Returns the type of this user-defined scan segment. This attribute can have the following values:

abstract	Indicates that this segment is an abstract segment (that is, embedded in a blackbox module).
fixed	Indicates that the elements in the segment are connected in the user-specified order.
floating	Indicates that the elements in the segment are not necessarily connected in the user-specified order.
preserve	Indicates that the order of the presumed connected elements was preserved.
shift_register	Indicates that this segment is a shift register. This implies that the order of the elements in the segment is fixed.

Related Information

Set by these constraints:

[define dft abstract segment](#)

[define dft fixed segment](#)

[define dft floating segment](#)

[define dft preserved segment](#)

[define dft shift register segment](#)

Related attributes:

(actual_scan_segment) [type](#) on page 1275

user_defined_segment

`user_defined_segment {true | false}`

Read-only `scan_segment` attribute. Indicates whether the segment is a user-defined segment or a shift register segment that was automatically identified by RTL Compiler.

Related Information

Set by these constraints:

[define dft abstract segment](#)

[define dft fixed segment](#)

[define dft floating segment](#)

[define dft preserved segment](#)

[define dft shift register segment](#)

Set by this command:

[identify shift register scan segments](#)

Test Bus Port Attributes

Contain information about the test bus ports in the specified design. These attributes are set using the `define_dft test_bus_port` command.

These attributes are read-only attributes, so you cannot set their values.

- To get an `test_bus_port` attribute value, type

```
get_attribute attribute_name \
[find /des*/design/dft -test_bus_port test_bus_port]
```

Note: These attributes are located at:

```
/designs/design/dft/test_bus_ports
```

dft hookup_pin

```
dft_hookup_pin {pin | port}
```

Read-only `test bus port` attribute. Returns the path to the pin or port where the test signal actually hooks up inside the core.

Related Information

Set by this constraint: [define_dft test_bus_port](#)

dft hookup_polarity

```
dft_hookup_polarity {inverted | non_inverted}
```

Read-only `test bus port` attribute. Indicates whether the test signal is inverted or not at the hookup pin or port.

Related Information

Set by this constraint: [define_dft test_bus_port](#)

function

`function string`

Read-only test_bus_port attribute. Returns the function of the test bus port.

For a list of possible functions refer to the *Command Reference for Encounter RTL Compiler*.

Related Information

Hierarchical Test in *Design for Test in Encounter RTL Compiler*

Set by this constraint: define_dft test_bus_port

index

`index integer`

Read-only test_bus_port attribute. Returns the index of the test bus port.

Test bus ports with the same function have an index to distinguish them. For example, the ports of an n-bit scan data input bus have an index between 0 and n-1.

Related Information

Hierarchical Test in *Design for Test in Encounter RTL Compiler*

Set by this constraint: define_dft test_bus_port

pin

`pin {pin| port}`

Read-only test_bus_port attribute. Returns the pin or port on which the test bus port is defined.

Related Information

Hierarchical Test in *Design for Test in Encounter RTL Compiler*

Set by this constraint: define_dft test_bus_port

Test Clock Attributes

Contain information about the DFT test clocks either identified by the DFT rule checker or defined by the user.

- To set a `test_clock` attribute, type

```
set_attribute attribute_name attribute_value [find /des*/design -test_clock  
name]
```

- To get a `test_clock` attribute value, type

```
get_attribute attribute_name [find /des*/design -test_clock name]
```

Note: These attributes are located at `/designs/design/dft/test_clock_domains`.

If the test clock is not defined by a constraint, its attributes are set by `check_dft_rules`.

at_speed

```
at_speed {false | true}
```

Default: false

Read-write `test_clock` attribute. Specifies whether the test clock is an at-speed test clock.

When you use a test synthesis flow with OPCG logic insertion, the tool automatically identifies the test clocks generated by the OPCG domain macros and sets the `at_speed` attribute to `true` for these test clocks.

When you perform synthesis on a block in a bottom-up test synthesis flow, and you apply a test clock to the block that is generated by an OPCG domain macro located outside the block, you must set this attribute manually on the clock input pin.

When you perform synthesis on a block in a bottom-up test synthesis flow, and a test clock is internally generated in the block and connected to an output pin or port of the block, the `write_dft_abstract_model` command will set this attribute to `true` for this test clock and add it to the scan abstract model.

Related Information

[Block-Level Domain-Blocking Flow in Design for Test in Encounter RTL Compiler](#)

Related commands:

[insert_dft_opcg](#)

[write_dft_abstract_model](#)

atpg_use

```
atpg_use {none | mask_clock | misr_clock | opcg_load_clock | opcg_ref_clock}
```

Default: none

Read-write test_clock attribute. Specifies the ATPG purpose of the test clock. This attribute can have the following values:

mask_clock	Test clock that controls the mask registers.
misr_clock	Test clock that controls the MISR registers.
none	Signal has no special ATPG purpose.
opcg_load_clock	Test clock used to clock the side scan chains in the OPCG logic.
opcg_ref_clock	Test clock that is used as a reference clock for a PLL.

Related Information

Set by these commands: [compress_scan_chains](#)

[define_dft_osc_source](#)

[insert_dft_opcg](#)

Related attribute (test_signal) [atpg_use](#) on page 1321

blocking_se

`blocking_se test_signal`

Read-write test_clock attribute. Specifies the blocking shift-enable signal to be used for the specified test clock.

When you use a test synthesis flow with OPCG logic insertion, the tool automatically identifies the blocking shift-enable signal for the generated test clocks and sets the `blocking_se` attribute on the test clocks.

When you perform synthesis on a block in a bottom-up test synthesis flow, and you apply a test clock to the block that is generated by an OPCG domain macro located outside the block, you must set this attribute manually on the clock input pin.

When you perform synthesis on a block in a bottom-up test synthesis flow, and a test clock is internally generated in the block and connected to an output pin or port of the block, the

Attribute Reference for Encounter RTL Compiler

Design For Test—Test Clock Attributes

`write_dft_abstract_model` command will set this attribute for this test clock and add it to the scan abstract model.

Related Information

[Block-Level Domain-Blocking Flow](#) in *Design for Test in Encounter RTL Compiler*

Related commands: [insert_dft opcg](#)
[write_dft_abstract_model](#)

controllable

`controllable {true | false}`

Default: true

Read-write `test_clock` attribute. Indicates whether the test clock is controllable in test mode.

dft hookup pin

`dft hookup_pin {pin|port|bus}`

Read-only `test_clock` attribute. Returns the path to the pin or port where the test signal or scan data input or output actually hooks up inside the core.

Example

```
rc:/designs/test> get_att dft_hookup_pin dft/test_clock_domains/clk/clk  
/designs/test/ports_in/clk
```

Related Information

Affected by these commands: [define_dft test_clock](#)

Related attribute: (test_signal) [dft hookup pin](#) on page 1323

dft hookup polarity

`dft hookup_polarity {inverted | non_inverted}`

Read-only `test_clock` attribute. Indicates whether the test signal is inverted or not at the hookup pin or port.

Example

```
rc:/designs/test> get_att dft hookup_polarity /designs/test/dft/
test_clock_domains/clk/clk
non_inverted
```

Related Information

Affected by these commands: [define_dft test_clock](#)

Related attribute: (test_signal) [dft hookup polarity](#) on page 1324

dft_mask_clk

```
dft_mask_clock {false | true}
```

Default: false

Read-write [test_clock](#) attribute. Indicates whether the test clock is used for compression mask logic. This attribute is used by the `write_atpg` and `write_et` commands to produce the correct assign files when a different clock is used to control the mask.

Related Information

Set by this command: [compress_scan_chains](#)

Related attributes: [compressed](#) on page 1252

[dft_compression_signal](#) on page 1323

dft_misr_clock

```
dft_misr_clock {false | true}
```

Default: false

Read-write [test_clock](#) attribute. Indicates whether the test clock is used for compression misr logic. This attribute is used by the `write_atpg` and `write_et` commands to produce the correct assign files when a different clock is used to control the misr.

Related Information

Set by this command: [compress_scan_chains](#)

Related attributes: [compressed](#) on page 1252

[dft_compression_signal](#) on page 1323

divide_fall

`divide_fall integer`

Default: 100

Read-write `test_clock` attribute. Returns the value specified using the `-divide_fall` option of the `define_dft test_clock` command.

Related Information

Set by this constraint: [define_dft_test_clock](#)

Set by this command: [check_dft_rules](#)

Related attribute: (test_signal) [divide_fall](#) on page 1324

divide_period

`divide_period integer`

Default: 1

Read-write `test_clock` attribute. Returns the value specified using the `-divide_period` option of the `define_dft test_clock` command.

Related Information

Set by this constraint: [define_dft_test_clock](#)

Set by this command: [check_dft_rules](#)

Related attribute: (test_signal) [divide_period](#) on page 1325

divide_rise

`divide_rise integer`

Default: 100

Read-write `test_clock` attribute. Returns the value specified using the `-divide_rise` option of the `define_dft test_clock` command.

Related Information

Set by this constraint: [define_dft test_clock](#)

Set by this command: [check_dft_rules](#)

Related attribute: (test_signal) [divide_rise](#) on page 1325

fall

`fall integer`

Default: 90 (falling edge at 90 percent of the cycle period)

Read-write [test_clock](#) attribute. Returns the value specified using the `-fall` option of the `define_dft test_clock` command.

Related Information

Set by this constraint: [define_dft test_clock](#)

Set by this command: [check_dft_rules](#)

Related attribute: (test_signal) [fall](#) on page 1325

off_state

`off_state {0 | 1}`

Read-only [test_clock](#) attribute. Indicates the off-state of the system clock in scan-shift mode.

Note: This attribute applies only to the clocked LSSD scan style. For other scan styles this attribute has no value.

Related Information

Set by this command: [check_dft_rules](#)

Affected by this attribute: [dft_scan_style](#) on page 1110

period

`period integer`

Default: 50000 (in picoseconds, corresponds to a clock frequency of 20MHz)

Read-write `test_clock` attribute. Returns the value specified using the `-period` option of the `define_dft test_clock` command.

Note: Even when you did not define any test clock, they are automatically identified by the `check_dft_rules` command and given default values for the clock.

Related Information

Set by this constraint: [define_dft test_clock](#)

Set by this command: [check_dft_rules](#)

Related attribute: (test_signal) [period](#) on page 1328

rise

`rise integer`

Default: 50 (rising edge at 50 percent of the cycle period)

Read-write `test_clock` attribute. Returns the value specified using the `-rise` option of the `define_dft test_clock` command.

Related Information

Set by this constraint: [define_dft test_clock](#)

Set by this command: [check_dft_rules](#)

Related attribute: (test_signal) [rise](#) on page 1330

root_source_pins

`root_source_pins list`

Read-only `test_clock` attribute. Returns a Tcl list of the ports and pins that correspond to the drivers of the clock.

Attribute Reference for Encounter RTL Compiler

Design For Test—Test Clock Attributes

Note: For a primary clock, or an internal test clock defined with the `-controllable` option, the value of this attribute is the same as that of the `sources` attribute.

Related Information

[Defining Internal Clock Branches as Separate Test Clocks in Design for Test in Encounter RTL Compiler](#)

Set by this constraint: [define_dft test_clock](#)

Set by this command: [check_dft_rules](#)

root_source_polarity

`root_source_polarity {inverting | non_inverting}`

Read-only [test_clock](#) attribute. Returns the polarity between the internal clock pin and its root pin.

Note: For a primary clock, or an internal test clock defined with the `-controllable` option, the value of this attribute is always `non_inverting`.

Related Information

Set by this constraint: [define_dft test_clock](#)

Set by this command: [check_dft_rules](#)

sources

`sources string`

Read-write [test_clock](#) attribute. Returns a Tcl list of the ports and pins that are sources of the test clock waveform.

Example

```
rc:/> get_att sources [find /*/dft -test_clock test_clock]
/designs/core/ports_in/clk1 /designs/core/ports_in/clk2 /designs/core/ports_in/
clk3
```

user_defined_signal

`user_defined_signal {true|false}`

Read-write `test_clock` attribute. Indicates whether the test clock was user-defined (through a `define_dft test_clock` constraint) or determined by the DFT rule checker.

Note: Although this attribute is writable, you are expected not to change this attribute value.

Related Information

Affected by this constraint: [define_dft test_clock](#)

Affected by this command: [check_dft_rules](#)

Related attribute: (test signal) [user_defined_signal](#) on page 1331

Test Signal Attributes

Contain information about user-defined `shift_enable`, `test_mode`, or `scan` clock signals.

- To set a `test_signal` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -test_signal name]
```

- To get a `test_signal` attribute value, type

```
get_attribute attribute_name [find /des*/design -test_signal name]
```

Note: These attributes are located at:

`/designs/design/dft/test_signals`

active

`active dft_active_value`

Read-write `test_signal` attribute. Specifies the active value of the test signal. The value can be either `high` or `low`.

Related Information

Set by these constraints:

[define dft shift_enable](#)

[define dft test_mode](#)

Set by this command:

[check_dft_rules](#)

Related attributes:

(actual_scan_segment) [active](#) on page 1262

(scan_segment) [active](#) on page 1292

atpg_use

```
atpg_use {none | master_test_enable | misr_reset_clock | opcg_blocking_se |  
          opcg_custom_se | opcg_enable | opcg_edge_mode | opcg_trigger}
```

Default: none

Read-write test_signal attribute. Specifies the ATPG purpose of the test signal. This attribute can have the following values:

master_test_enable	Signal is used to gate the compression enable signal used for compression.
misr_reset_clock	Signal is used to asynchronously reset the MISR.
none	Signal has no special ATPG purpose.
opcg_blocking_se	Signal is used as domain blocking scan enable.
opcg_custom_se	Signal is used as special blocking scan enable generated by that OPCG domain's domain macro. OPCG domain macro.
opcg_enable	Signal is used to enable the on-product clock generation logic.
opcg_edge_mode	Signal is used to connect to the toggle muxes added to the input flops of an OPCG domain.
opcg_trigger	Signal is used to trigger the generation of delay test pulses.

Related Information

Set by these commands:

[compress_scan_chains](#)
[connect_scan_chains](#)
[define_dft_opcg_trigger](#)
[define_dft_osc_source](#)
[insert_dft_opcg](#)

Related attribute

(test_clock) [atpg_use](#) on page 1312

dedicated_pin

```
dedicated_pin {false | true}
```

Default: false

Read-write test_signal attribute. Indicates whether the driving pin (port) of a test signal is considered dedicated for test. The driving port is considered dedicated for test if the port was created by the RC-DFT engine, or if the existing port was not specified as a shared functional data port when defining the test signal.

Related Information

Set by these constraints:

define_dft_scan_clock_a
define_dft_scan_clock_b
define_dft_shift_enable
define_dft_test_mode

default_shift_enable

```
default_shift_enable {false | true}
```

Default: false

Read-write test_signal attribute. Indicates if this test signal is the default shift-enable signal.

Note: This attribute applies only to test signals of type `shift_enable`. For other types of test signals this attribute has no value.

Related Information

Set by this constraint:

define_dft_shift_enable

Set by this command:

check_dft_rules

dft_compression_signal

```
dft_compression_signal {none | mask_enable | mask_load | compression_enable  
| spreader}
```

Default: none

Read-write test signal attribute. Indicates the type of the compression signal. This attribute can have the following values:

compression_enable	Indicates that the test signal is the compression (space compactor) enable signal.
mask_enable	Indicates that the test signal is the channel mask enable signal.
mask_load	Indicates that the test signal is the channel mask load signal.
none	Indicates that the test signal is not related to chain compression.
spreader	Indicates that the test signal is the decompressor spread enable signal.

Related Information

- Set by this command: [compress_scan_chains](#)
Related attributes: [compressed on page 1252](#)
 [dft_mask_clk on page 1314](#)

dft hookup pin

```
dft_hookup_pin {pin|port|bus}
```

Read-only test signal attribute. Returns the path to the pin or port where the test signal or scan data input or output actually hooks up inside the core.

Example

```
rc:/> get_att dft_hookup_pin test_signals/SE  
/designs/test/ports_in/SE
```

Related Information

- Affected by these constraints: [define_dft_scan_clock_a](#)
 [define_dft_scan_clock_b](#)

[define_dft shift_enable](#)

[define_dft test_mode](#)

Related attribute: (test_clock) [dft hookup_pin](#) on page 1313

dft hookup polarity

`dft hookup_polarity {inverted | non_inverted}`

Read-only [test signal](#) attribute. Indicates whether the test signal is inverted or not at the hookup pin or port.

Example

```
rc:/> get_att dft hookup_polarity test_signals/SE  
non_inverted
```

Related Information

Affected by these constraints:

[define_dft scan_clock_a](#)

[define_dft scan_clock_b](#)

[define_dft shift_enable](#)

[define_dft test_mode](#)

Related attribute: (test_clock) [dft hookup_polarity](#) on page 1313

divide_fall

`divide_fall integer`

*Default:*100

Read-write [test signal](#) attribute. Returns the value specified using the `-divide_fall` option of the `define_dft scan_clock_a` or `define_dft scan_clock_b` command.

Related Information

Set by these constraints:

[define_dft scan_clock_a](#)

[define_dft scan_clock_b](#)

Related attribute: (test_clock) [divide_fall](#) on page 1315

divide_period

`divide_period integer`

Default: 1

Read-write `test_signal` attribute. Returns the value specified using the `-divide_period` option of the `define_dft scan_clock_a` or `define_dft scan_clock_b` command.

Related Information

Set by these constraints: [define_dft scan_clock_a](#)
 [define_dft scan_clock_b](#)

Related attribute: (test_clock) [divide_period](#) on page 1315

divide_rise

`divide_rise integer`

Default: 100

Read-write `test_signal` attribute. Returns the value specified using the `-divide_rise` option of the `define_dft scan_clock_a` or `define_dft scan_clock_b` command.

Related Information

Set by these constraints: [define_dft scan_clock_a](#)
 [define_dft scan_clock_b](#)

Related attribute: (test_clock) [divide_rise](#) on page 1315

fall

`fall integer`

Default: 60 (80) for a test signal of type `scan_clock_a` (`scan_clock_b`)

Read-write `test_signal` attribute. Returns the value specified using the `-fall` option of the `define_dft scan_clock_a` or `define_dft scan_clock_b` command.

Attribute Reference for Encounter RTL Compiler

Design For Test—Test Signal Attributes

Related Information

Set by these constraints: [define_dft_scan_clock_a](#)

[define_dft_scan_clock_b](#)

Related attribute: (test_clock) [fall](#) on page 1316

has_fanout

`has_fanout {false | true | test_only}`

Read-write [test_signal](#) attribute. Specifies whether the `test_mode` pin has a fanout to timing endpoints when the test signal was defined. If the test signal was defined with the `-test_only` option, this attribute will be set to `test_only`.

Related Information

Set by these constraints: [define_dft_test_mode](#)

Set by this command: [check_dft_rules](#)

ideal

`ideal {true| false}`

Default: true

Read-write [test_signal](#) attribute. Indicates if this test signal is marked as ideal. Marking a test signal as ideal, prevents buffering of the shift-enable or test-mode network during optimization.

Related Information

Set by these constraints: [define_dft_scan_clock_a](#)

[define_dft_scan_clock_b](#)

[define_dft_shift_enable](#)

[define_dft_test_mode](#)

Set by this command: [check_dft_rules](#)

lec_value

```
lec_value {auto | 0 | 1 | no_value}
```

Default: auto

Read-write test signal attribute. Specifies how to constrain the test pin for LEC validation with the `write_do_lec` command. This attribute can have the following values:

0	Indicates that a logic 0 will be specified for the <code>add pin</code> constraint command in the do file.
1	Indicates that a logic 1 will be specified for the <code>add pin</code> constraint command in the do file.
auto	Indicates that the opposite of the test mode active value will be specified for the <code>add pin</code> constraint command in the do file.
no_value	Indicates that no <code>add pin</code> constraint command will be written for this test signal in the do file.

Related Information

Rules for Constraining Test Signals in DFT-Related Commands *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Set by these constraints: [define_dft shift enable](#)

[define_dft test mode](#)

Affects this command: [write do lec](#)

master_signal

```
master_signal {true| false}
```

Default: true

Read-write test signal attribute. Specifies whether this test signal is a master test signal.

Note: This attribute is relevant when multiple test signals are defined on a pin.

period

period integer

Default: 50000 (in picoseconds, corresponds to a clock frequency of 20MHz)

Read-write [test_signal](#) attribute. Returns the value specified using the -period option of the `define_dft scan_clock_a` or `define_dft scan_clock_b` command.

Related Information

Set by these constraints: [define_dft_scan_clock_a](#)
 [define_dft_scan_clock_b](#)

Related attribute: (test_clock) [period](#) on page 1317

pin

pin string

Read-write [test_signal](#) attribute. Specifies the port or hierarchical pin associated with the test signal.

Related Information

Set by these constraints: [define_dft_scan_clock_a](#)
 [define_dft_scan_clock_b](#)
 [define_dft_shift_enable](#)
 [define_dft_test_mode](#)

Set by this command: [check_dft_rules](#)

Related attribute: (jtag_port) [pin](#) on page 1202

pmbist_use

```
pmbist_use {none | all | test_block_async_reset | test_clock_select |  
test_rambypass}
```

Default: none

Read-write `test_signal` attribute. Specifies how the test signal should be used to control the programmable MBIST logic during ATPG. Three controls can be used for the PMBIST logic during ATPG: `test_block_async_reset`, `test_clock_reset`, and `test_rambypass`. This attribute can have the following values:

all	Uses this test signal for all three controls of the PMBIST logic.
none	Does not use this test signal to control PMBIST during ATPG.
test_block_async_reset	Applies this test signal to the input of the PMBIST logic which controls the async reset register inputs used within the design. When asserted, the test signal blocks the async reset input to registers within the PMBIST logic, allowing some control in testing these async reset paths. You must specify this control.
test_clock_select	Applies this test signal to the input of the PMBIST logic which controls the selection of the clock source for the PMBIST test data registers which are shared by the JTAG and direct access functions. Selection of the clock source may be required for ATPG true time analysis. When you use the JTAG access and direct access with two different clock sources for these access methods, you must specify the control. When this test signal is set to its active state, it selects the PMBIST direct access <code>mda_tck</code> clock source; otherwise the JTAG clock source is selected.
test_rambypass	Applies this test signal to the input of the PMBIST logic which is necessary for SRAMs which lack internal ATPG bypass logic. The tool inserts such logic to support logic testing around the SRAMs during ATPG. The test signal is only required when such bypass logic is requested at the time of PMBIST logic insertion. When this test signal is set to its active state, it forces the memory data inputs to bypass the memory and be routed to the data outputs.

Related Information

Affects this command: [insert_dft pmbist](#)

rise

`rise integer`

Default: 50 (70) for a test signal of type `scan_clock_a` (`scan_clock_b`)

Read-write [test_signal](#) attribute. Returns the value specified using the `-rise` option of the `define_dft scan_clock_a` or `define_dft scan_clock_b` command.

Related Information

Set by this constraint: [define_dft scan_clock_a](#)

[define_dft scan_clock_b](#)

Related attribute: (test_clock) [rise](#) on page 1317

scan_shift

`scan_shift {false | true}`

Default: false

Read-write [test_signal](#) attribute. Specifies whether this signal will be captured as a clock for ATPG

Related Information

Set by this constraint: [define_dft test_mode](#)

type

`type test_signal_type`

Read-write `test_signal` attribute. Specifies whether this signal is a scan clock of an LSSD scan cell, a shift-enable or a test-mode signal. This attribute can have the following values: `scan_clock_a`, `scan_clock_b`, `shift_enable`, or `test_mode`.

Related Information

Set by these constraints: [define dft shift_enable](#)

[define dft test mode](#)

Set by this command: [check dft rules](#)

user_defined_signal

`user_defined_signal {true|false|simulated}`

Default: true

Read-write `test_signal` attribute. Indicates whether the test signal was user-defined (through a `define_dft` constraint), determined by the DFT rule checker, or identified by simulating a mode initialization sequence.

Note: Although this attribute is writable, you are expected not to change this attribute value.

Related Information

Set by this constraint: [define dft test mode](#)
 [identify test mode registers](#)

Set by this command: [check dft rules](#)

Related attribute: (test clock) [user_defined_signal](#) on page 1319

Attribute Reference for Encounter RTL Compiler

Design For Test—Test Signal Attributes

Low Power Synthesis

Root Attributes

- [lp_leakage_power_effort](#) on page 1338
- [lp_clock_gating_exceptions_aware](#) on page 1339
- [lp_clock_gating_infer_enable](#) on page 1339
- [lp_clock_gating_prefix](#) on page 1340
- [lp_clock_gating_register_aware](#) on page 1341
- [lp_default_probability](#) on page 1341
- [lp_default_toggle_rate](#) on page 1342
- [lp_display_negative_internal_power](#) on page 1342
- [lp_insert_clock_gating](#) on page 1343
- [lp_power_analysis_effort](#) on page 1344
- [lp_power_unit](#) on page 1345
- [lp_pso_aware_estimation](#) on page 1346
- [lp_toggle_rate_unit](#) on page 1346
- [lp_x_transition_probability_count](#) on page 1347
- [lp_x_transition_toggle_count](#) on page 1347
- [lp_z_transition_probability_count](#) on page 1348
- [lp_z_transition_toggle_count](#) on page 1348

Design Attributes

- [lp_clock_gating_add_obs_port](#) on page 1349

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—List

- [lp_clock_gating_add_reset](#) on page 1350
- [lp_clock_gating_auto_cost_group_initial_target](#) on page 1351
- [lp_clock_gating_auto_cost_grouping](#) on page 1351
- [lp_clock_gating_auto_path_adjust](#) on page 1352
- [lp_clock_gating_auto_path_adjust_fixed_delay](#) on page 1352
- [lp_clock_gating_auto_path_adjust_modes](#) on page 1353
- [lp_clock_gating_auto_path_adjust_multiplier](#) on page 1354
- [lp_clock_gating_cell](#) on page 1354
- [lp_clock_gating_control_point](#) on page 1355
- [lp_clock_gating_exclude](#) on page 1357
- [lp_clock_gating_extract_common_enable](#) on page 1357
- [lp_clock_gating_max_flops](#) on page 1358
- [lp_clock_gating_min_flops](#) on page 1358
- [lp_clock_gating_module](#) on page 1359
- [lp_clock_gating_style](#) on page 1359
- [lp_clock_gating_test_signal](#) on page 1360
- [lp_clock_tree_buffers](#) on page 1361
- [lp_clock_tree_leaf_max_fanout](#) on page 1361
- [lp_default_probability](#) on page 1361
- [lp_default_toggle_percentage](#) on page 1362
- [lp_default_toggle_rate](#) on page 1363
- [lp_internal_power](#) on page 1363
- [lp_leakage_power](#) on page 1364
- [lp_net_power](#) on page 1364
- [lp_power_optimization_weight](#) on page 1365
- [max_dynamic_power](#) on page 1366
- [max_leakage_power](#) on page 1367

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—List

Instance Attributes

- [instance internal power](#) on page 1368
- [instance leakage power](#) on page 1369
- [lp_clock_gating add reset](#) on page 1369
- [lp_clock_gating cell](#) on page 1370
- [lp_clock_gating exclude](#) on page 1371
- [lp_clock_gating gated_clock_gates](#) on page 1372
- [lp_clock_gating gated_flops](#) on page 1372
- [lp_clock_gating is flop_rc_gated](#) on page 1373
- [lp_clock_gating is flop_user_gated](#) on page 1373
- [lp_clock_gating is leaf_clock_gate](#) on page 1373
- [lp_clock_gating module](#) on page 1374
- [lp_clock_gating_rc_inserted](#) on page 1374
- [lp_clock_gating_stage](#) on page 1374
- [lp_default_probability](#) on page 1376
- [lp_default_toggle_rate](#) on page 1377
- [lp_dynamic_analysis_scope](#) on page 1377
- [lp_internal_power](#) on page 1378
- [lp_leakage_power](#) on page 1378
- [lp_net_power](#) on page 1379

Pin Attributes

- [lp_asserted_probability](#) on page 1380
- [lp_asserted_toggle_rate](#) on page 1381
- [lp_computed_probability](#) on page 1381
- [lp_computed_toggle_rate](#) on page 1382
- [lp_default_probability](#) on page 1383

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—List

- [lp_default_toggle_rate](#) on page 1384
- [lp_net_power](#) on page 1385
- [lp_probability_type](#) on page 1386
- [lp_toggle_rate_type](#) on page 1387

Net Attributes

- [lp_asserted_probability](#) on page 1389
- [lp_asserted_toggle_rate](#) on page 1390
- [lp_computed_probability](#) on page 1390
- [lp_computed_toggle_rate](#) on page 1391
- [lp_net_power](#) on page 1392
- [lp_probability_type](#) on page 1393
- [lp_toggle_rate_type](#) on page 1394

Port Attributes

- [lp_asserted_probability](#) on page 1396
- [lp_asserted_toggle_rate](#) on page 1396
- [lp_computed_probability](#) on page 1397
- [lp_computed_toggle_rate](#) on page 1398
- [lp_default_probability](#) on page 1399
- [lp_default_toggle_rate](#) on page 1400
- [lp_net_power](#) on page 1401
- [lp_probability_type](#) on page 1402
- [lp_toggle_rate_type](#) on page 1403

Subdesign Attributes

- [lp_clock_gating_add_reset](#) on page 1404
- [lp_clock_gating_auto_path_adjust](#) on page 1405

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—List

- [lp_clock_gating_auto_path_adjust_fixed_delay](#) on page 1406
- [lp_clock_gating_auto_path_adjust_multiplier](#) on page 1406
- [lp_clock_gating_cell](#) on page 1407
- [lp_clock_gating_exclude](#) on page 1408
- [lp_clock_gating_module](#) on page 1409

Support Attributes

- [lp_asserted_probability](#) on page 1411
- [lp_asserted_toggle_rate](#) on page 1412
- [lp_computed_probability](#) on page 1412
- [lp_computed_toggle_rate](#) on page 1413
- [lp_net_power](#) on page 1414
- [lp_probability_type](#) on page 1415
- [lp_toggle_rate_type](#) on page 1416

Clock Attributes

- [lp_default_toggle_percentage](#) on page 1417

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

leakage_power_effort

```
leakage_power_effort {none | low | medium | high}
```

Default: none

Read-write `root` attribute. Enables and controls leakage power optimization, together with the leakage power constraint (set using the `max_leakage_power` root attribute). When performing leakage power optimization, the tool will make a trade off between leakage power and delay, area, and runtime.

high	Performs the best leakage power optimization with a higher trade-off on delay, area, and runtime. In this case, the runtime can be significantly higher.
low	Performs a low-effort leakage power optimization with minimum trade-off on delay, area, and runtime.
medium	Performs a medium-effort leakage power optimization with some trade-off on delay, area, and runtime. The result is in between what you would get with low and high effort levels.
none	Disables leakage power optimization only when you do not specify an explicit leakage power constraint (that is, set the <code>max_leakage_power</code> attribute).

Note: Without an explicit leakage power constraint setting, and when the `leakage_power_effort` attribute is set to either `low`, `medium`, or `high`, the tool assumes that the `max_leakage_power` attribute is set to 0.

Related Information

[Controlling Tradeoff between Leakage Power, and Area, Delay, and Runtime in Low Power in Encounter RTL Compiler](#)

Affects these commands: [synthesize](#)
[report power](#)

Affected by this attribute: [max_leakage_power](#) on page 1367

lp_clock_gating_exceptions_aware

`lp_clock_gating_exceptions_aware {true | false}`

Default: true

Read-write [root](#) attribute. Specifies whether to take timing exceptions set on flop instances or on their clock, enable, and reset pins into account during clock gating.

Related Information

[Controlling Handling of Timing Exceptions during Clock Gating in Low Power in Encounter RTL Compiler.](#)

Affects these commands: [synthesize](#)
[report clock_gating](#)

lp_clock_gating_infer_enable

`lp_clock_gating_infer_enable {true | false | set_reset_flops_only}`

Default: true

Read-write [root](#) attribute. When clock-gating insertion is enabled, enabling this attribute will invoke an advanced algorithm that identifies additional clock-gating opportunities—even in the absence of a feedback loop—that cannot be identified using the basic algorithm.

false Does not invoke the advanced algorithm.

`set_reset_flops_only` Only invokes the advanced algorithm for set and reset flops.

true Performs the advanced algorithm without restrictions.

Note: This attribute is ignored if you use the `synthesize` command with the `-effort` option set to `low`, because the low-effort synthesis always disables `infer_enable`.

Related Information

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

Affects these commands: [synthesize](#)
[clock_gating_insert_in_netlist](#)
[report_clock_gating](#)

Affected by this attribute: [lp_insert_clock_gating](#) on page 1343

lp_clock_gating_prefix

`lp_clock_gating_prefix string`

Read-write [root](#) attribute. Specifies the prefix to be added to all clock-gating modules, observability flip-flops, generated clock nets, and the ports created by clock-gating insertion.

Example

If you set this attribute to `lowp`, the names of the clock-gating instances will be similar to `lowp_RC(CG)_HIER_INST_xx`, while the name of the gated clock net will be similar to `lowp_rc_gclk_xx`.

Related Information

[Controlling Naming of Modules, Nets, and Ports in Low Power in Encounter RTL Compiler.](#)

Affects these commands: [elaborate](#)
[synthesize](#)
[report_clock_gating](#)

lp_clock_gating_register_aware

```
lp_clock_gating_register_aware {false | true}
```

Default: false

Read-write root attribute. Controls clock-gating for register banks.

Example

Assume flops a[0], a[1], a[2], b[0], b[1], and b[2] belong to the same hierarchy.

- `set_attribute lp_clock_gating_register_aware false /`
By default, one clock-gating instance may be inserted for all these flops.
- `set_attribute lp_clock_gating_register_aware true /`
In this case, one clock-gating instance will be added for a[0], a[1], and a[2], and one clock-gating instance will be added for b[0], b[1], and b[2].

Related Information

Affects these commands: [elaborate](#)
[synthesize](#)
[report clock_gating](#)

lp_default_probability

```
lp_default_probability float
```

Default: 0.5

Read-write root attribute. Sets the default probability value (probability of a net being high) for power estimation. RTL Compiler uses this value for those nets that have no user-specified probability value. Specify any value between zero and one.

Related Information

[Sources of Switching Activity Information in Low Power in Encounter RTL Compiler](#)

Affects this command: [report power](#)

Affects this attribute: (design) [lp_default_probability](#) on page 1361

Related attributes:	(instance) lp_default_probability on page 1376
	(pin) lp_default_probability on page 1383
	(port) lp_default_probability on page 1399

lp_default_toggle_rate

`lp_default_toggle_rate float`

Default: 0.02

Read-write [root](#) attribute. Specifies the default toggle rate (toggle count per toggle rate unit). RTL Compiler uses this value for those nets that have no user-specified toggle rate. You can specify any positive value, including zero. The unit is determined by the [lp_toggle_rate_unit](#) attribute.

Related Information

[Sources of Switching Activity Information in Low Power in Encounter RTL Compiler](#)

Affects this command:	report power
Affects this attribute:	(design) lp_default_toggle_rate on page 1363
Related attributes:	(instance) lp_default_toggle_rate on page 1377 (pin) lp_default_toggle_rate on page 1384 (port) lp_default_toggle_rate on page 1400

lp_display_negative_internal_power

`lp_display_negative_internal_power {true | false}`

Default: true

Read-write [root](#) attribute. Controls how negative internal power results are reported. By default, negative internal power results are reported. Set this attribute to false, to report negative internal power as zero.

Related Information

Power Analysis in Low Power in Encounter RTL Compiler

Affects these commands:	report power report gates -power report instance -power report qor
Affects these attributes:	(design) lp_internal_power on page 1363 (instance) lp_internal_power on page 1378

lp_insert_clock_gating

`lp_insert_clock_gating {false | true}`

Default: false

Read-write [root](#) attribute. Controls insertion of clock-gating logic during synthesis.

Related Information

Enabling Clock Gating in Low Power in Encounter RTL Compiler.

Affects these commands:	synthesize -to_clock_gated synthesize -to_mapped report clock_gating
-------------------------	--

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Root Attributes

lp_power_analysis_effort

`lp_power_analysis_effort {medium | low | high}`

Default: medium

Read-write root attribute. Controls whether to propagate the switching activities in the design. The attribute can have the following values:

high	Propagates the switching activities in the design with higher accuracy.
low	Does not propagate the switching activities and uses the default settings instead.
medium	Propagates the switching activities in the design.

Note: The higher the power analysis effort, the higher the run time and memory usage are.

Related Information

[Sources of Switching Activity Information in Low Power in Encounter RTL Compiler](#)

Affects these commands: [synthesize](#)

[report power](#)

Affects these attributes: [lp_computed_probability](#) on page 1390

[lp_computed_toggle_rate](#) on page 1391

Affected by these attributes: [lp_asserted_probability](#) on page 1389

[lp_asserted_toggle_rate](#) on page 1390

lp_power_unit

lp_power_unit {nW | mW | pW | uW}

Default: nW

Read-write root attribute. Specifies the power unit to be used when analyzing net power, cell internal power, or cell leakage power. The power units are case sensitive.

Note: To get more precise results, use a smaller unit.

Related Information

Power Analysis in Low Power in Encounter RTL Compiler

Affects this command:	report gates -power report power
Affects these attributes:	(design) lp_internal_power on page 1363 (instance) lp_internal_power on page 1378 (design) lp_leakage_power on page 1364 (instance) lp_leakage_power on page 1378 (design) lp_net_power on page 1379 (instance) lp_net_power on page 1379 (pin) lp_net_power on page 1385 (net) lp_net_power on page 1392 (port) lp_net_power on page 1401 (subport) lp_net_power on page 1414

lp_pso_aware_estimation

`lp_pso_aware_estimation {true | false}`

Default: true

Read-write root attribute. Specifies if the power estimation must be power domain-aware.

true	<p>Specifies to estimate the average power of the design based on the probability of the power domains being powered on.</p> <p>To make the power estimation power domain-aware, you must assert the probability of the enable signals that power down the power domains.</p> <p>The RC-LP engine will consider a power domain to be always on if the probability of the shutoff enable signal is not user-asserted</p>
false	<p>Specifies to estimate the power assuming that all power domains are always powered on.</p>

Related Information

Affects these commands: [report gates -power](#)

[report power](#)

Related attribute: [lp_pso_aware_tcf](#) on page 1365

lp_toggle_rate_unit

`lp_toggle_rate_unit {/ns | /us | /ms | /s}`

Default: /ns

Read-write root attribute. Specifies the time unit used for the toggle rate in RTL Compiler.

Related Information

Affects this command: [report power](#)

Affects these attributes: [lp_asserted_toggle_rate](#) on page 1390

[lp_computed_toggle_rate](#) on page 1391

 (design) [lp_default_toggle_rate](#) on page 1363

 (instance) [lp_default_toggle_rate](#) on page 1377

lp_x_transition_probability_count

`lp_x_transition_probability_count float`

Default: 0.5

Read-write root attribute. Specifies the weight of the probability count for each transition from X. You can specify any value between 0 and 1.

Note: To ignore the X transitions, you can set this attribute or the `lp_x_transition_toggle_count` attribute to -1.

Related Information

Affects these commands: [read_vcd](#)

[report power](#)

Affects these attributes: [lp_asserted_probability](#) on page 1389

[lp_computed_probability](#) on page 1390

lp_x_transition_toggle_count

`lp_x_transition_toggle_count float`

Default: 0.5

Read-write root attribute. Specifies the weight of the toggle count for each transition from and to X. You can specify any value between 0 and 1.

Note: To ignore the X transitions, you can set this attribute or the `lp_x_transition_probability_count` attribute to -1.

Related Information

Affects these commands: [read_vcd](#)

[report power](#)

Affects these attributes: [lp_asserted_toggle_rate](#) on page 1390

[lp_computed_toggle_rate](#) on page 1391

lp_z_transition_probability_count

`lp_z_transition_probability_count float`

Default: 0.25

Read-write root attribute. Specifies the weight of the probability count for each transition from Z. You can specify any value between 0 and 1.

Note: To ignore the Z transitions, you can set this attribute or the `lp_z_transition_toggle_count` attribute to -1.

Related Information

Affects these commands: [read_vcd](#)

[report power](#)

Affects these attributes: [lp_asserted_probability](#) on page 1389

[lp_computed_probability](#) on page 1390

lp_z_transition_toggle_count

`lp_z_transition_toggle_count float`

Default: 0.25

Read-write root attribute. Specifies the weight of the toggle count for each transition from and to Z. You can specify any value between 0 and 1.

Note: To ignore the Z transitions, you can set this attribute or the `lp_z_transition_probability_count` attribute to -1.

Related Information

Affects these commands: [read_vcd](#)

[report power](#)

Affects these attributes: [lp_asserted_toggle_rate](#) on page 1390

[lp_computed_toggle_rate](#) on page 1391

Design Attributes

Contain information about the specified design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design
```

or

```
set_attribute attribute_name attribute_value [find /des* -design name]
```

- To get a design attribute value, type

```
get_attribute attribute_name design
```

lp_clock_gating_add_obs_port

```
lp_clock_gating_add_obs_port {false | true}
```

Default: false

Read-write design attribute. Specifies whether the integrated clock-gating cell must contain observability logic.

- If you set this attribute to `true`, the low power (RC-LP) engine only uses integrated clock-gating cells that have a `clock_gating_integrated_cell` attribute that contains `obs` in the fourth substring.

The RC-LP engine adds an `obs` port to the clock-gating module and connects this port to the pin of the integrated clock-gating cell that has the `clock_gate_obs_pin` attribute.

- If you set this attribute to `false`, the RC-LP engine only uses integrated clock-gating cells that have a `clock_gating_integrated_cell` attribute that has no entry for the fourth substring.

Related Information

[Clock Gating Cell Specification](#) in the *Library Guide for Encounter RTL Compiler*.

[Using Attributes to Select an Integrated Clock-Gating Cell in the Technology Library in Low Power in Encounter RTL Compiler](#)

Affects these commands:

[clock_gating_insert_obs](#)

[synthesize](#)

[report_clock_gating](#)

lp_clock_gating_add_reset

`lp_clock_gating_add_reset {false | true}`

Default: false

Read-write design attribute. Determines whether the integrated clock-gating cell must contain reset logic added to the glitch removing logic.

- If you set this attribute to `true`, and the gated registers have an asynchronous reset or set pin, the low power (RC-LP) engine only uses an integrated clock-gating cell that has a pin with a `clock_gate_reset_pin` attribute defined.

If the RC-LP engine finds such a cell, it adds an `a_rst` port to the clock-gating module and internally connects this port to the reset pin of the integrated clock-gating cell.

If `c1, c2, ..., cn` represent the asynchronous reset signals for each of the registers controlled by this clock-gating instance, and `p1, p2, ..., pn` represent the asynchronous set signals for each of the registers controlled by this clock-gating instance, the RC-LP engine determines the reset signal for the clock-gating instance as:

$(c1+p1) \text{ AND } (c2+p2) \text{ AND } \dots \text{ AND } (cn+pn)$

The RC-LP engine creates the corresponding logic and connects the output of this logic to the `a_rst` pin of the clock-gating instance.

Setting the design attribute to `true`, automatically sets the `lp_clock_gating_add_reset` attribute to `true` on all flops in the design.

Note: If you set this attribute to `true`, but the gated registers do not have an asynchronous reset or set pin, no reset logic will be added.

- If you set this attribute to `false`, the RC-LP engine does not add an `a_rst` port to the clock-gating module, even when the gated registers have asynchronous set or reset pins.

Related Information

[Clock Gating Cell Specification](#) in the *Library Guide for Encounter RTL Compiler*.

[Using Attributes to Select an Integrated Clock-Gating Cell in the Technology Library in Low Power in Encounter RTL Compiler](#)

Affects these commands:	<u>synthesize</u> <u>report clock gating</u>
Affects these attributes:	(instance) <u>lp_clock_gating_add_reset</u> on page 1369 (subdesign) <u>lp_clock_gating_add_reset</u> on page 1404
Related attribute:	<u>polarity</u> on page 223

lp_clock_gating_auto_cost_group_initial_target

`lp_clock_gating_auto_cost_group_initial_target float`

Default:

Read-write design attribute. Specifies the initial target slack for automatic clock gate cost groups in picoseconds. This attribute will set the initial target for all newly generated clock-gating logic.

When no value is specified, the tool will generate the initial target for these cost groups during mapping.

Related Information

[Tightening Timing Constraints on the Clock-Gating Enables in Low Power in Encounter RTL Compiler](#)

Affects these commands: [synthesize](#)
 [report timing](#)

lp_clock_gating_auto_cost_grouping

`lp_clock_gating_auto_cost_grouping {true | false}`

Default: true

Read-write design attribute. Controls whether to automatically create a cost group for the paths going through clock gate enable pins. The cost groups are defined after clock gating is inserted and have the following name:

`cg_enable_group_XXX`

where XXX is the name of the clock.

Related Information

[Tightening Timing Constraints on the Clock-Gating Enables in Low Power in Encounter RTL Compiler](#)

Affects these commands: [synthesize](#)
 [report timing](#)

lp_clock_gating_auto_path_adjust

`lp_clock_gating_auto_path_adjust string`

Read-write design attribute. Controls the automatic timing adjustment on the clock gate enable paths in the design. You can specify any of the following values:

fixed	Specifies to use the user-defined path adjust value.
none	Prevents the automatic timing adjustment.
variable	Specifies to scale the tool-calculated path adjust values.

Related Information

Tightening Timing Constraints on the Clock-Gating Enables in Low Power in Encounter RTL Compiler

Affects these commands: **synthesize**

Affects these attributes:

- (subdesign) lp_clock_gating_auto_path_adjust on page 1405
- (design) lp_clock_gating_auto_path_adjust_fixed_delay on page 1352
- (design) lp_clock_gating_auto_path_adjust_multiplier on page 1406

Ip clock gating auto path adjust fixed delay

lp clock gating auto path adjust fixed delay float

Read-write design attribute. Specifies a user-defined path adjust value (in picoseconds) for the enable pins of all clock-gating instances in the design. This value overrides the tool-calculated path adjust values. You must specify a positive value. A negative or null value does not affect the path constraints.

The path_adjust timing exceptions are stored in the following directory:

/designs/top/timing/exceptions/path_adjusts/

The tool-computed path_adjust timing exceptions are calculated based on information in the libraries and have the following name:

_auto_xxps_cg_path_adjust

where `xx` is the user-provided delay value.

Example

The following command specifies to tighten the path to the enable pins of all clock-gating instances in design `top` with 100ps.

```
set_attribute lp_clock_gating_auto_path_adjust_fixed_delay 100 /designs/top
```

Related Information

[Tightening Timing Constraints on the Clock-Gating Enables in Low Power in Encounter RTL Compiler](#)

Affects these commands: [synthesize](#)
[report timing](#)

Affected by this attribute: (design) [lp_clock_gating_auto_path_adjust](#) on page 1352

Affects this attribute: (subdesign) [lp_clock_gating_auto_path_adjust_fixed_delay](#)
on page 1406

lp_clock_gating_auto_path_adjust_modes

`lp_clock_gating_auto_path_adjust_modes modes`

Read-write [design](#) attribute. Specifies the timing constraint modes in which automatic timing adjustment on the clock gate enable paths will be applied. By default, the path adjust is applied in all modes.

Related Information

[Tightening Timing Constraints on the Clock-Gating Enables in Low Power in Encounter RTL Compiler](#)

Affects these commands: [synthesize](#)
[report timing](#)

lp_clock_gating_auto_path_adjust_multiplier

`lp_clock_gating_auto_path_adjust_multiplier float`

Read-write design attribute. Scales the tool-calculated path adjust values added to the enable pins of all clock-gating instances in the design. Specify a real number between 0.0 and infinity.

The path_adjust timing exceptions are stored in the following directory:

`/designs/top/timing/exceptions/path_adjusts/`

The tool-computed path_adjust timing exceptions are calculated based on information in the libraries and have the following name:

`_auto_XXps_cg_path_adjust`

where XX is the magnitude of the adjustment.

Related Information

[Tightening Timing Constraints on the Clock-Gating Enables in Low Power in Encounter RTL Compiler](#)

Affects these commands: [synthesize](#)

[report timing](#)

Affected by this attribute: [\(design\) lp_clock_gating_auto_path_adjust](#) on page 1352

Affects this attribute: [\(subdesign\) lp_clock_gating_auto_path_adjust_multiplier](#) on page 1406

lp_clock_gating_cell

`lp_clock_gating_cell path_name_for_cell`

Read-write design attribute. Specifies the path name of the cell to be used for clock-gating insertion. The path name can contain the wildcard character (*). This attribute overrides the following attributes: `lp_clock_gating_add_obs_port`, `lp_clock_gating_add_reset`, `lp_clock_gating_control_point`, and `lp_clock_gating_style`.

If the specified cell does not exist in any of the libraries, the auto clock-gating insertion will fail. If multiple cells are found, the attribute is not set and the tool reports an error.

Examples

The following examples sets CGIC_LPP as the clock-gating cell to use:

```
rc:/> set_att lp_clock_gating_cell /libr*/cg/libcells/CGIC_LPP /designs/top
      Setting attribute of design top: 'lp_clock_gating_cell' =
/libraries/cg/libcells/CGIC_LPP
rc:/designs/top> set_att lp_clock_gating_cell [find / -libcell CGIC_LPP]
      Setting attribute of design top: 'lp_clock_gating_cell' =
/libraries/cg/libcells/CGIC_LPP
```

Related Information

[Specifying the Library Cell Name of the Integrated Clock-Gating Cell in Low Power in Encounter RTL Compiler.](#)

Affects these commands: [synthesize](#)
[report clock gating](#)

Related attribute: (subdesign) [lp_clock_gating_cell](#) on page 1407

lp_clock_gating_control_point

`lp_clock_gating_control_point {precontrol | postcontrol | none }`

Default: precontrol

Read-write [design](#) attribute. If a user-defined clock-gating module is used, specifies whether the module contains test logic. If no user-defined clock gating module or special library cell is specified for clock gating, the attribute controls whether the integrated clock-gating cell (to be selected) must contain test control logic, and specifies where the test logic must be located—before or after the latch or flip-flop in the integrated clock-gating cell.

The attribute can have the following values:

none	If a user-defined clock-gating module is used, specifies that the module has no test logic. If an integrated clock-gating cell must be selected, specifies that only cells that have a <code>clock_gating_integrated_cell</code> attribute that has no entry for the third substring can be used.
------	--

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Design Attributes

precontrol	If a user-defined clock-gating module is used, specifies that the module has test logic. If an integrated clock-gating cell must be selected, specifies to only use cells whose <code>clock_gating_integrated_cell</code> (.lib) attribute contains <code>precontrol</code> in the third substring. The <code>test</code> port of the clock-gating cell will be connected to the pin with the <code>clock_gate_test_pin</code> attribute.
postcontrol	If a user-defined clock-gating module is used, specifies that the the module has a test port. If an integrated clock-gating cell must be selected, specifies to only use cells whose <code>clock_gating_integrated_cell</code> (.lib) attribute contains <code>postcontrol</code> in the third substring. The <code>test</code> port of the clock-gating cell will be connected to the pin with the <code>clock_gate_test_pin</code> attribute.

Note: To select a clock-gating cell that contains test control logic without a latch or flip-flop, set the `lp_clock_gating_control_point` attribute to either `precontrol` or `postcontrol`, and set the `lp_clock_gating_style` attribute to `none`. The RC-LP engine then looks for a clock-gating cell in the library with the value `none_posedge_control` or `none_needge_control` for the `clock_gating_integrated_cell` attribute.

Related Information

[Clock Gating Cell Specification in the Library Guide for Encounter RTL Compiler](#).

[Using Attributes to Select an Integrated Clock-Gating Cell in the Technology Library in Low Power in Encounter RTL Compiler](#)

Affects these commands: [report clock gating](#)
 [synthesize](#)

lp_clock_gating_exclude

`lp_clock_gating_exclude {false | true}`

Default: false

Read-write design attribute. Controls whether to insert clock-gating logic for this design. If you set this attribute to `true`, no clock-gating logic is added to the design.

Related Information

Controlling Insertion of Clock-Gating Logic in Low Power in Encounter RTL Compiler

Affects this command: [synthesize](#)

Related attributes: (instance) [lp_clock_gating_exclude](#) on page 1371

 (subdesign) [lp_clock_gating_exclude](#) on page 1408

lp_clock_gating_extract_common_enable

`lp_clock_gating_extract_common_enable {true | false}`

Default: true

Read-write design attribute. Controls whether to extract the common enable from the enable logic of the registers to be clock gated.

Register banks can sometimes not be considered for clock-gating insertion because their bit width is smaller than the minimum number of registers required for clock-gating insertion. If the flops that were not gated—because of this minimum bit width requirement—have a complex enable logic, common enable extraction can extract a common function in the enable logic of those registers. By considering this common function as the enable signal, the minimum bit width requirement can be satisfied and the registers can be gated.

Related Information

Controlling Insertion of Clock-Gating Logic in Low Power in Encounter RTL Compiler

Affects this command: [synthesize](#)

Related attribute: [lp_clock_gating_min_flops](#) on page 1358

lp_clock_gating_max_flops

`lp_clock_gating_max_flops integer`

Default: infinity

Read-write design attribute. Determines the maximum number of registers that can be driven by each clock-gating element. If a register bank has a bit width larger than the specified size, the low power (RC-LP) engine will duplicate the clock-gating cells and distribute the registers evenly over the clock-gating cells. Specify an integer larger than 1.

Note: This attribute applies to any clock-gating logic type: user-defined clock-gating module, selected clock-gating integrated cell, or discrete clock-gating logic created by the tool.

Example

If the register bank width is 32, and the maximum number is 20, each clock-gating cell will be driving 16 registers.

Related Information

[Controlling Insertion of Clock-Gating Logic in Low Power in Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

lp_clock_gating_min_flops

`lp_clock_gating_min_flops integer`

Default: 3

Read-write design attribute. Enables clock-gating insertion for any register bank with a bit width larger than or equal to the specified size. You can specify a value from 1 to 1000.

Note: This attribute applies to any clock-gating logic type: user-defined clock-gating module, selected clock-gating integrated cell, or discrete clock-gating logic created by the tool.

Related Information

[Controlling Insertion of Clock-Gating Logic in Low Power in Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

lp_clock_gating_module

`lp_clock_gating_module path_name_for_module`

Read-write design attribute. Specifies the path to the module that defines the customized clock-gating logic.

Related Information

Defining a Clock-Gating Module in Low Power in Encounter RTL Compiler

Affected by these attributes: (instance) [Ip_clock_gating_module](#) on page 1374
(subdesign) [Ip_clock_gating_module](#) on page 1409

lp_clock_gating_style

`lp_clock_gating_style {latch | ff | none}`

Default: latch

Read-write design attribute. Controls whether the integrated clock-gating cells must contain logic to prevent glitches on the enable signal and specifies whether to use a latch or a flip-flop for the logic.

The low power (RC-LP) engine only uses integrated clock-gating cells that have a `clock_gating_integrated_cell` attribute of which the first substring matches the value of this attribute.

Related Information

Clock Gating Cell Specification in the *Library Guide* for *Encounter RTL Compiler*.

Using Attributes to Select an Integrated Clock-Gating Cell in the Technology Library in Low Power in Encounter RTL Compiler

Affects these commands: report clock gating
synthesize

lp_clock_gating_test_signal

`lp_clock_gating_test_signal string`

Read-write design attribute. Indicates which test signal to connect to the test pins of all the clock-gating instances in the design. If the clock-gating instance contains test logic and you did not specify a test signal, RTL Compiler ties the test pin of all clock gating instances to the inactive value. For example, for active high (low) test pin, the pin is tied to constant 0 (1). You can specify the following values:

<code>use_shift_enable</code>	Indicates to use the shift-enable signal of the scan chain to which the gated flip-flops belong. If the gated flip-flops belong to different scan chains with different shift enable signals, the shift enable signals are OR-ed and the output of the OR-tree is used as the clock-gating test signal.
<code>test_signal_object</code>	Indicates to use the specified test signal object.

Note: Test signals are defined using the `define_dft test_mode` or `define_dft shift_enable` commands. The corresponding object is stored at `/designs/design/dft/test_signals`.

Example

The following command instructs to connect the test pins of all clock-gating instances in the design to test signal `tm`.

```
set_attr lp_clock_gating_test_signal /designs/alu/dft/test_signals/tm /des*/alu
```

Related Information

[Clock Gating with DFT in Low Power in Encounter RTL Compiler](#).

[Clock-Gating Insertion in a Scan-Connected Netlist in Low Power in Encounter RTL Compiler](#).

Affects this command:	<u>synthesize</u>
Related attribute:	(instance) <u>lp_clock_gating_test_signal</u> on page 1375 (subdesign) <u>lp_clock_gating_test_signal</u> on page 1409

lp_clock_tree_buffers

`lp_clock_tree_buffers libcell_list`

Read-write design attribute. Specifies a list of valid libcell buffers and inverters that can be used for clock tree synthesis.

Note: If the design has multiple library domains, specify the buffers and inverters that belong to the default library domain.

Related Information

Affects this command: [report power](#)

Related attribute: [lp_clock_tree_leaf_max_fanout](#) on page 1361

lp_clock_tree_leaf_max_fanout

`lp_clock_tree_leaf_max_fanout integer`

Default: 0

Read-write design attribute. Specifies the maximum number of flip-flops that can be driven by a leaf clock buffer. Specify a positive integer.

Related Information

Affects this command: [report power](#)

Related attribute: [lp_clock_tree_buffers](#) on page 1361

lp_default_probability

`lp_default_probability float`

Default: no_value

Read-write design attribute. Sets the default probability value (probability of a net being high) for power estimation. RTL Compiler uses this value for those nets that have no user-specified probability value. Specify any value between zero and one.

Note: This attribute is not hierarchical. It only applies to the nets at the top-level of the design.

Related Information

Sources of Switching Activity Information in Low Power in Encounter RTL Compiler

Affects this command:	report power
Affected by this attribute:	(root) lp_default_probability on page 1341
Related attributes:	(instance) lp_default_probability on page 1376
	(pin) lp_default_probability on page 1383
	(port) lp_default_probability on page 1399

lp_default_toggle_percentage

`lp_default_toggle_percentage float`

Default: 0.2

Read-write [design](#) attribute. Specifies the multiplication factor to be used with all clocks in the design to modify the *default* toggle rate of any activity propagation starting point associated with clock information. The toggle rate of an activity propagation starting point is derived by multiplying this scaling factor with the toggle rate of the associated clock information. If multiple clocks are related to the pin, the fastest clock is used. Specify a value between 0 and 1. You can use a value greater than 1.0 to model cases with high glitching power which could result in more than one value change per clock on average. A warning is printed if you specify a value larger than 1.

Related Information

Using Default Switching Activities in Low Power in Encounter RTL Compiler

Affects this command:	report power
Affects this attribute:	(clock) lp_default_toggle_percentage on page 1417
Affected by this attribute:	lp_toggle_rate_unit on page 1346

lp_default_toggle_rate

`lp_default_toggle_rate float`

Default: no_value

Read-write design attribute. Specifies the default toggle rate (toggle count per toggle rate unit). RTL Compiler uses this value for those nets that have no user-specified toggle rate. You can specify any positive value, including zero. The unit is determined by the `lp_toggle_rate_unit` attribute.

Note: This attribute is not hierarchical. It only applies to the nets at the top-level of the design.

Related Information

Sources of Switching Activity Information in Low Power in Encounter RTL Compiler

Affects this command: [report power](#)

Affected by this attribute: (root) [lp_default_toggle_rate](#) on page 1342

[lp_toggle_rate_unit](#) on page 1346

Related attributes: (instance) [lp_default_toggle_rate](#) on page 1377

(pin) [lp_default_toggle_rate](#) on page 1384

(port) [lp_default_toggle_rate](#) on page 1400

lp_internal_power

`lp_internal_power float`

Read-only design attribute. Computes the total internal cell power of all instances in the design. The unit of the power value is determined by the value of the `lp_power_unit` attribute.

Related Information

Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler

Related commands: [report gates -power](#)

[report power](#)

Affected by this attribute: [lp_power_unit](#) on page 1345

Related attributes: (instance) [lp_internal_power](#) on page 1378

lp_leakage_power

`lp_leakage_power float`

Read-only design attribute. Computes the total leakage power of all instances in the design. The unit of the power value is determined by the value of the `lp_power_unit` attribute.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

[Reporting Leakage Power in Low Power in Encounter RTL Compiler](#)

Related commands: [report gates -power](#)
[report power](#)

Affected by this attribute: [lp_power_unit](#) on page 1345

Related attribute: (instance) [lp_leakage_power](#) on page 1378

lp_net_power

`lp_net_power float`

Read-only design attribute. Computes the total switching power of all nets in the design. The unit of the power value is determined by the value of the `lp_power_unit` attribute.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands: [report gates -power](#)
[report power](#)

Affected by this attribute: [lp_power_unit](#) on page 1345

Related attributes: (instance) [lp_net_power](#) on page 1379
(pin) [lp_net_power](#) on page 1385
(net) [lp_net_power](#) on page 1392
(port) [lp_net_power](#) on page 1401
(subport) [lp_net_power](#) on page 1414

lp_power_optimization_weight

`lp_power_optimization_weight float`

Default: no_value

Read-write design attribute. Controls the weight factors to be used when optimizing leakage power and dynamic power simultaneously during global mapping, mapping, and incremental optimization. Specify a value between zero and one. Assuming the attribute is set to w, the RC-LP engine optimizes for total power. Total power is computed as follows:

$$\text{Total power} = w \times \text{leakage_power} + (1-w) \times \text{dynamic_power}$$

Note: The weight factor will only be taken into account if both the `max_dynamic_power` and `max_leakage_power` attributes are set.

If you do not set this attribute and both the `max_dynamic_power` and `max_leakage_power` attributes are set, the tool issues warning POPT-501 and will ignore dynamic power optimization.

Related Information

[Controlling Power Optimization in Low Power in Encounter RTL Compiler](#)

Affects these commands: [synthesize](#)

Related commands: [report gates -power](#)

[report power](#)

Related attributes: [max_dynamic_power](#) on page 1366

[max_leakage_power](#) on page 1367

[leakage_power_effort](#) on page 1338

lp_pso_aware_tcf

`lp_pso_aware_tcf {false | true}`

Default: false

Read-write design attribute. Specifies whether the switching activities were generated by simulating the design while taking into account when the power domains are shut off.

Related Information

Affects these commands: [report gates -power](#)

[report power](#)

Related attribute: [lp_pso_aware_estimation](#) on page 1346

max_dynamic_power

`max_dynamic_power float`

Default: no_value

Read-write [design](#) attribute. Specifies the maximum dynamic-power constraint of the design. The dynamic power is the sum of the internal power dissipated in all instances and the switching power dissipated in all nets.

You must specify a constraint value to enable RTL Compiler to perform timing and dynamic power optimization simultaneously during mapping (without taking the constraint value into account). During global and incremental optimization, the constraint value is taken into account.

Related Information

[Enabling Dynamic Power Optimization in Low Power in Encounter RTL Compiler](#)

Affected by this attribute: [lp_power_unit](#) on page 1345

Related attributes: [lp_power_optimization_weight](#) on page 1365

[max_leakage_power](#) on page 1367

[leakage_power_effort](#) on page 1338

max_leakage_power

`max_leakage_power float`

Default: no_value

Read-write `design` attribute. Specifies the maximum leakage-power constraint of the design. The constraint enables RTL Compiler to perform timing and leakage power optimization simultaneously during mapping (without taking the constraint value into account). During global and incremental optimization, the constraint value is taken into account.

Related Information

[Enabling Leakage Power Optimization in Low Power in Encounter RTL Compiler](#)

Affects this command: [synthesize](#)

Affected by this attribute: [lp_power_unit](#) on page 1345

Related attributes: [lp_power_optimization_weight](#) on page 1365

[max_dynamic_power](#) on page 1366

[leakage_power_effort](#) on page 1338

Instance Attributes

Contain information about a combinational, sequential, or hierarchical instance in the specified design. Instance objects are found in the `instances_comb`, `instances_seq`, and `instances_hier` directories.

- To set an instance attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -instance name]
```

- To get an instance attribute value, type

```
get_attribute attribute_name [find /des*/design -instance name]
```

instance_internal_power

`instance_internal_power float`

Read-write `instance` attribute. Specifies the internal power of this instance. The unit of the power value is determined by the value of the `lp_power_unit` attribute. Use this attribute to specify the internal power of a sequential cell, combinational cell, blackbox, abstract model, or timing model.

By default, an instance inherits the internal power of the corresponding libcell.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands:

[report gates -power](#)

[report power](#)

Affected by this attribute:

[lp_power_unit on page 1345](#)

Related attributes:

[\(libcell\) cell_internal_power on page 192](#)

[\(libcell\) cell_leakage_power on page 192](#)

[instance_leakage_power on page 1369](#)

instance_leakage_power

`instance_leakage_power float`

Read-write `instance` attribute. Specifies the leakage power of this instance. The unit of the power value is determined by the value of the `lp_power_unit` attribute. Use this attribute to specify the internal power of a sequential cell, combinational cell, blackbox, abstract model, or timing model.

By default, an instance inherits the leakage power of the corresponding libcell.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands:	report gates -power report power
Affected by this attribute:	lp_power_unit on page 1345
Related attributes:	(libcell) cell_internal_power on page 192 (libcell) cell_leakage_power on page 192 instance_internal_power on page 1368

lp_clock_gating_add_reset

`lp_clock_gating_add_reset {false | true}`

Default: false

Read-write `instance` attribute. Determines whether the integrated clock-gating cell must contain reset logic added to the glitch removing logic. This attribute can be set on flops and hierarchical instances.

- If you set this attribute to `true`, and the gated registers have an asynchronous reset or set pin, the RC-LP engine only uses an integrated clock-gating cell that has a pin with a `clock_gate_reset_pin` attribute defined.

If the RC-LP engine finds such a cell, it adds an `a_RST` port to the clock-gating module and internally connects this port to the reset pin of the integrated clock-gating cell.

If `c1, c2, ..., cn` represent the asynchronous reset signals for each of the registers controlled by this clock-gating instance, and `p1, p2, ..., pn` represent the asynchronous

set signals for each of the registers controlled by this clock-gating instance, the RC-LP engine determines the reset signal for the clock-gating instance as:

$$(c_1+p_1) \text{ AND } (c_2+p_2) \text{ AND } \dots \text{ AND } (c_n+p_n)$$

The RC-LP engine creates the corresponding logic and connects the output of this logic to the `a_rst` pin of the clock-gating instance.

Setting this attribute to `true` on a hierarchical instance, automatically sets the `lp_clock_gating_add_reset` attribute to `true` on all flops in the hierarchical instance.

Note: If you set this attribute to `true`, but the gated registers do not have an asynchronous reset or set pin, no reset logic will be added.

- If you set this attribute to `false`, the RC-LP engine does not add an `a_rst` port to the clock-gating module, even when the gated registers have asynchronous set or reset pins.

Related Information

[Clock Gating Cell Specification](#) in the *Library Guide for Encounter RTL Compiler*.

[Using Attributes to Select an Integrated Clock-Gating Cell in the Technology Library in Low Power in Encounter RTL Compiler](#)

Affects these commands:	synthesize report clock gating
Affected by these attributes:	(design) lp_clock_gating_add_reset on page 1350 (subdesign) lp_clock_gating_add_reset on page 1404
Related attribute:	polarity on page 223

lp_clock_gating_cell

`lp_clock_gating_cell path_name_for_cell`

Read-only `instance` attribute. Returns the path name of the cell (to be) used for clock-gating insertion. This attribute can be queried on hierarchical instances and leaf sequential instances. This attribute will return no value if the attribute was not set either on the subdesign corresponding to this instance (or its parent) or on the design.

Examples

- The following command queries the clock-gating cell used for hierarchical instance `i_bot`:

```
rc:/> set_attr lp_clock_gating_cell TLATNCAX12 [find / -subdesign bot]
...
rc:/designs/top/instances_hier/i_mid/instances_hier> get_att
lp_clock_gating_cell i_bot
/libraries/library_domains/ld1/typical/libcells/TLATNCAX12
```

This command returns the path to the cell used for inserting clock-gating logic in this instance, because the attribute was set on the corresponding subdesign.

- The following command queries the clock-gating cell used for leaf instance `out_date_reg` which belongs to hierarchical instance `i_bot`:

```
rc:/designs/top> get_att lp_clock_gating_cell [find -instance
*i_mid/i_bot/out*]
/libraries/library_domains/ld1/typical/libcells/TLATNCAX12
```

Related Information

Affected by these attributes: (design) [lp_clock_gating_cell](#) on page 1354
(subdesign) [lp_clock_gating_cell](#) on page 1407

lp_clock_gating_exclude

`lp_clock_gating_exclude {false | true}`

Default: `false`

Read-write `instance` attribute. Determines whether to insert clock-gating logic for this instance. If you set this attribute to `true` on a register, no clock-gating logic is added to this register even when it possible. If you set this attribute to `true` on a hierarchical instance, no clock-gating logic is added to any of the registers in that instance and the hierarchy below that instance.

Note: You can only set this attribute on a unique hierarchical or flat instance. Otherwise, the tool issues an error message and the attribute is ignored. If a hierarchical instance is instantiated multiple times, first use the `edit_netlist dedicate_subdesign` command to uniquify the instance that you want to exclude from clock-gating.

Example

Assume that design `top` contains a subdesign (module) `sub` that has been instantiated five times in the design. The instance names are `i1`, `i2`, `i3`, `i4`, and `i5`. Now you want to exclude clock-gating from instance `i1`:

```
set_attribute lp_clock_gating_exclude true /designs/top/instances_hier/i1
```

This attribute will be ignored because instance `i1` is not a unique instance of subdesign `sub`. To ensure that instance `i1` is excluded from clock gating, you first need to make it a unique instance before setting the attribute:

```
edit_netlist dedicate_subdesign{/designs/top/instances_hier/i1}
```

Related Information

[Controlling Insertion of Clock-Gating Logic in Low Power in Encounter RTL Compiler](#)

Affects this command:	<u>synthesize</u>
Related command:	<u>edit netlist dedicate subdesign</u>
Related attributes:	(design) <u>lp_clock_gating_exclude</u> on page 1357 (subdesign) <u>lp_clock_gating_exclude</u> on page 1408

lp_clock_gating_gated_clock_gates

`lp_clock_gating_gated_clock_gates string`

Read-only [instance](#) attribute. Returns the clock gates gated by this clock-gating instance.

Note: This attribute applies only to clock-gating instances.

Related Information

Affected by this command:	<u>synthesize</u>
---------------------------	-----------------------------------

lp_clock_gating_gated_flops

`lp_clock_gating_gated_flops string`

Read-only [instance](#) attribute. Returns the flops gated by this clock-gating instance.

Note: This attribute applies only to clock-gating instances.

Related Information

Affected by this command: [synthesize](#)

lp_clock_gating_is_flop_rc_gated

`lp_clock_gating_is_flop_rc_gated {false | true}`

Read-only [instance](#) attribute. Indicates whether the flop is gated by a clock-gating instance inserted by RTL Compiler.

Note: This attribute applies only to sequential instances.

lp_clock_gating_is_flop_user_gated

`lp_clock_gating_is_flop_user_gated {false | true}`

Read-only [instance](#) attribute. Indicates whether the flop is gated by a user-instantiated clock-gating instance.

Note: This attribute applies only to sequential instances.

lp_clock_gating_is_leaf_clock_gate

`lp_clock_gating_is_leaf_clock_gate {false | true}`

Read-only [instance](#) attribute. Specifies whether this clock-gating instance is a leaf clock gate, that is, a clock-gating instance driving the flops.

Note: This attribute applies only to clock-gating instances.

Related Information

[Multi-Stage Clock Gating in Low Power in Encounter RTL Compiler](#)

Affected by this command: [synthesize](#)

Ip_clock_gating_module

`lp_clock_gating_module path_name_for_module`

Read-only instance attribute. Returns the path name of the module used for clock-gating insertion. This attribute can be queried on hierarchical instances and leaf sequential instances. This attribute will return no value if the attribute was not set either on the subdesign corresponding to this instance (or its parent) or on the design.

Example

The following command queries the clock-gating module used for hierarchical instance `mul_13_16`:

```
rc:/designs> get_att lp_clock_gating_module [find -instance mul_13_16]  
/designs/my_CG_MOD
```

Related Information

Affected by these attributes: (design) [lp_clock_gating_module](#) on page 1359
(subdesign) [lp_clock_gating_module](#) on page 1409

Ip_clock_gating_rc_inserted

```
lp_clock_gating_rc_inserted {false | true }
```

Read-only instance attribute. Indicates whether this clock-gating instance was inserted by RTL Compiler.

Note: This attribute will have no value for instances that are not clock-gating instances.

Related Information

Affected by these commands: [synthesize](#)
 [clock_gating import](#)

lp_clock_gating_stage

`lp_clock_gating_stage integer`

Read-only instance attribute. Returns the stage to which the clock-gating instance belongs.

Note: This attribute will have no value for instances that are not clock-gating instances.

Example

Assume clock-gating instance CG1 is gating clock-gating instance CG2, which in turn gates a flop. The stage of CG1 is 1 and the stage of CG2 is 2.

Related Information

Affected by this command: [synthesize](#)
[clock_gating share](#)

lp_clock_gating_test_signal

`lp_clock_gating_test_signal string`

Read-write instance attribute. Indicates which test signal to connect to the test pin of the (specified) clock-gating instance(s).

Note: The RC-LP engine creates a separate subdesign for the clock-gating logic in each clock-gating domain, consequently the clock-gating instance names are similar to

`/designs/design/instances_hier/*/RC(CG)_HIER_INSTx`

You can specify one of the following values:

`use_shift_enable` Indicates to use the shift-enable signal of the scan chain to which the gated flip-flops belong.
If the gated flip-flops belong to different scan chains with different shift enable signals, the shift enable signals are OR-ed and the output of the OR-tree is used as the clock-gating test signal.

Note: You can only use this value when you insert clock-gating in a mapped netlist *after* you have inserted the scan chains.

`test_signal_object` Indicates to use the specified test signal object.

Note: Test signals are defined using the `define_dft test_mode` or `define_dft shift_enable` commands. The corresponding object is stored at `/designs/design/dft/test_signals`.

Example

The following command instructs to connect the test pin of clock-gating instance RC(CG)_HIER_INST1 to the shift-enable signal that drives the scan flops gated by this clock-gating instance.

```
set_attribute lp_clock_gating_test_signal use_shift_enable [find . -inst  
RC(CG)_HIER_INST1]
```

Related Information

[Clock Gating with DFT in Low Power in Encounter RTL Compiler](#).

Affects this command:	<u>synthesize</u>
Related attributes:	(design) <u>lp_clock_gating_test_signal</u> on page 1360
	(subdesign) <u>lp_clock_gating_test_signal</u> on page 1409

lp_default_probability

`lp_default_probability float`

Default: no_value

Read-write [instance](#) attribute. Sets the default probability value (probability of a net being high) for power estimation. RTL Compiler uses this value for those nets that have no user-specified probability value. Specify any value between zero and one. You can only set this attribute on a hierarchical instance.

Note: This attribute is not hierarchical. It only applies to the nets of this hierarchical instance.

Related Information

[Sources of Switching Activity Information in Low Power in Encounter RTL Compiler](#)

Affects this command:	<u>report power</u>
Affects this attribute	(pin) <u>lp_default_probability</u> on page 1383
Affected by this attribute	(root) <u>lp_default_probability</u> on page 1341
Related attribute:	(design) <u>lp_default_probability</u> on page 1361

lp_default_toggle_rate

`lp_default_toggle_rate float`

Default: no_value

Read-write instance attribute. Specifies the default toggle rate (toggle count per toggle rate unit). RTL Compiler uses this value for those nets that have no user-specified toggle count. You can specify any positive value, including zero. You can only set this attribute on a hierarchical instance.

Note: This attribute is not hierarchical. It only applies to the nets of this hierarchical instance.

Related Information

Sources of Switching Activity Information in Low Power in Encounter RTL Compiler

Affects this command:

report power

Affects this attribute

(pin) lp_default_toggle_rate on page 1384

Affected by these attributes:

(root) lp_default_toggle_rate on page 1342

lp_toggle_rate_unit on page 1346

Related attribute:

(design) lp_default_toggle_rate on page 1363

lp_dynamic_analysis_scope

`lp_dynamic_analysis_scope {false | true}`

Default: false

Read-write instance attribute. Specifies whether the instance should be added to the scope that impacts activity profiling.

Example

The following command adds instances `i_mid` and `i_bot` to the scope used for activity profiling.

```
set_attribute lp_dynamic_analysis_scope true {i_mid i_bot}
```

Related Information

Affects this command:

read_vcd

lp_internal_power

`lp_internal_power float`

Read-only instance attribute. Computes the internal cell power of the instance. The unit of the power value is determined by the value of the `lp_power_unit` attribute.

Note: You can only get this attribute value on a unique hierarchical or leaf instance.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands: [report gates -power](#)

[report power](#)

Affected by this attribute: [lp_power_unit](#) on page 1345

Related attribute: (design) [lp_internal_power](#) on page 1363

lp_leakage_power

`lp_leakage_power float`

Read-only instance attribute. Computes the leakage power of the instance. The unit of the power value is determined by the value of the `lp_power_unit` attribute.

Note: You can only get this attribute value on a unique hierarchical or leaf instance.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

[Reporting Leakage Power in Low Power in Encounter RTL Compiler](#)

Related commands: [report gates -power](#)

[report power](#)

Affected by this attribute: [lp_power_unit](#) on page 1345

Related attribute: (design) [lp_leakage_power](#) on page 1364

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Instance Attributes

lp_net_power

`lp_net_power float`

Read-only instance attribute. Computes the sum of the net power of all output nets for a flat instance and the total output net power within a hierarchical instance excluding the nets whose drivers are not part of the hierarchical instance. The unit of the power value is determined by the value of the `lp_power_unit` attribute.

Note: You can only get this attribute value on a unique hierarchical or leaf instance.

Related Information

Reporting on Specific Power Components for Specific Objects in *Low Power in Encounter RTL Compiler*

Related commands:	report gates -power report power
Affected by this attribute:	lp_power_unit on page 1345
Related attributes:	(design) lp_net_power on page 1364 (pin) lp_net_power on page 1385 (net) lp_net_power on page 1392 (port) lp_net_power on page 1401 (subport) lp_net_power on page 1414

Pin Attributes

Contain information about the specified instance pin. A pin is a single bit connection point on an instance. Pin objects are stored in `pins_in` and `pins_out` directories.

- To set a pin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pin instance/pin]
```

- To get a pin attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pin instance/pin]
```

Note: You may need to specify more than one instance to serve as the search path that will uniquely identify the relevant pin.

lp_asserted_probability

`lp_asserted_probability float`

Read-write `pin` attribute. Specifies the probability value of a signal on this pin being high for power estimation. You can specify any value between 0 and 1.

Related Information

Sources of Switching Activity Information in Low Power in Encounter RTL Compiler

Set by one of these commands: [read_saif](#)

[read_tcf](#)

Affects these commands: [report_gates -power](#)

[report_power](#)

Related attributes: (net) [lp_asserted_probability](#) on page 1389

 (port) [lp_asserted_probability](#) on page 1396

 (subport) [lp_asserted_probability](#) on page 1411

lp_asserted_toggle_rate

`lp_asserted_toggle_rate float`

Read-write [pin](#) attribute. Specifies the toggle rate (toggle count per toggle rate unit) of this pin for the purpose of power estimation. You can specify any positive value, including zero.

Related Information

[Sources of Switching Activity Information in Low Power in Encounter RTL Compiler](#)

Set by one of these commands: [read_saif](#)

[read_tcf](#)

Affects these commands: [report_gates -power](#)

[report_power](#)

Affected by this attribute: [lp_toggle_rate_unit](#) on page 1346

Related attributes: (net) [lp_asserted_toggle_rate](#) on page 1390

(port) [lp_asserted_toggle_rate](#) on page 1396

(subport) [lp_asserted_toggle_rate](#) on page 1412

lp_computed_probability

`lp_computed_probability float`

Read-only [pin](#) attribute. Retrieves the probability of the pin. The probability can be asserted or computed.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)

[read_tcf](#)

Related commands: [report_gates -power](#)

[report_power](#)

Affected by these attributes:	(net) lp_asserted_probability on page 1389 (pin) lp_asserted_probability on page 1380 (port) lp_asserted_probability on page 1396 (subport) lp_asserted_probability on page 1411 (design) lp_default_probability on page 1361 (instance) lp_default_probability on page 1376
Related attributes:	(net) lp_computed_probability on page 1390 (port) lp_computed_probability on page 1397 (subport) lp_computed_probability on page 1412

lp_computed_toggle_rate

`lp_computed_toggle_rate float`

Read-only pin attribute. Retrieves the toggle rate (toggle count per toggle rate unit) of this pin for power estimation. The toggle rate can be asserted or computed.

Example

The following example associates a clock with port `clock`, finds the pins that the clock is driving and checks the computed toggle rate on one of those pins.

```
rc:/> get_att loads [get_att net */ports_in/clock]
{/designs/alu/instances_seq/aluout_reg[7]/pins_in/clk}
...
{/designs/alu/instances_seq/aluout_reg[0]/pins_in/clk}
/designs/alu/instances_seq/zero_reg/pins_in/clk
rc:/> get_att lp_computed_toggle_rate \
/designs/alu/instances_seq/aluout_reg[2]/pins_in/clk
0.571429
```

The toggle rate can be derived from the clock cycle. Because the default toggle rate unit is /ns and the period is expressed in ps, the toggle rate is determined by the reverse of the clock period times 2 (because the signal toggles two times per period) or 2/3.5ns.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)

[read_tcf](#)

Related commands: [report_gates -power](#)

[report_power](#)

Affected by these attributes: (net) [lp_asserted_toggle_rate](#) on page 1390
(pin) [lp_asserted_toggle_rate](#) on page 1381
(port) [lp_asserted_toggle_rate](#) on page 1396
(subport) [lp_asserted_toggle_rate](#) on page 1412
(design) [lp_default_toggle_rate](#) on page 1363
(instance) [lp_default_toggle_rate](#) on page 1377

Related attributes: (net) [lp_computed_toggle_rate](#) on page 1391
(port) [lp_computed_toggle_rate](#) on page 1398
(subport) [lp_computed_toggle_rate](#) on page 1413

lp_default_probability

`lp_default_probability float`

Read-only pin attribute. Returns the default probability of the pin. The value is determined as follows:

1. If the pin is asserted, it returns the value of the `lp_asserted_probability` pin attribute.
2. If the pin is not asserted, it returns the value of the `lp_default_probability` attribute of the instance the pin belongs to.
3. If the `lp_default_probability` instance attribute was not set, it returns the value of the `lp_default_probability` root attribute.

Note: This attribute can only have a value for output pins of sequential cells, blackboxes, and macros.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands:	report gates -power report power
Affected by these attributes:	(pin) lp_asserted_probability on page 1380 (instance) lp_default_probability on page 1376 (root) lp_default_probability on page 1341 lp_power_unit on page 1345

lp_default_toggle_rate

`lp_default_toggle_rate float`

Read-only [pin](#) attribute. Returns the default toggle rate of the pin. The value is determined as follows:

1. If the pin is asserted, it returns the value of the `lp_asserted_toggle_rate` pin attribute.
2. If the pin is not asserted, it returns the value of the `lp_default_toggle_rate` attribute of the instance the pin belongs to.
3. If the pin is not asserted and the `lp_default_toggle_rate` instance attribute is not set, the value is computed based on the `lp_default_toggle_percentage` and the associated clock information.
4. If the pin is not asserted, the `lp_default_toggle_rate` instance attribute is not set, and the `lp_default_toggle_percentage` attribute was not set, it returns the value of the `lp_default_toggle_rate` root attribute.

Note: This attribute can only have a value for output pins of sequential cells, blackboxes, and macros.

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Pin Attributes

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands:	report gates -power report power
Affected by these attributes:	(pin) lp_asserted_toggle_rate on page 1381 (design) lp_default_toggle_percentage on page 1362 (instance) lp_default_toggle_rate on page 1377 (root) lp_default_toggle_rate on page 1342 lp_power_unit on page 1345

lp_net_power

`lp_net_power float`

Read-only [pin](#) attribute. Computes the switching power dissipated on the net connected to this pin. The unit of the power value is determined by the value of the [lp_power_unit](#) attribute.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands:	report gates -power report power
Affected by this attribute:	lp_power_unit on page 1345
Related attributes:	(design) lp_net_power on page 1364 (instance) lp_net_power on page 1379 (net) lp_net_power on page 1392 (port) lp_net_power on page 1401 (subport) lp_net_power on page 1414

lp_probability_type

```
lp_probability_type {asserted | clock | computed | constant | default}
```

Read-only [pin](#) attribute. Computed attribute. Gives an indication of the source of the value of the probability value. This attribute can have the following values:

asserted	Indicates that the pin value is user-specified. You either specified the value through the lp_asserted_probability attribute or in a switching activity file (TCF or SAIF file).
clock	Indicates that the pin is a clock pin and that the value is derived from the clock waveform.
computed	Indicates that the pin value is computed by propagating the internal switching activities.
constant	Indicates that the pin is driven by a constant value. In this case, the value of the lp_asserted_probability attribute is set to 1 if it is driven by a logic 1, or 0 if it is driven by a logic 0.
default	Indicates that the pin value is not user-specified, but determined by the value of the lp_default_probability attribute.

Example

The following example checks the source of the probability value of a clock pin on a register.

```
rc:/> get_att lp_probability_type /des*/alu/*/aluout_reg[2]/pins_in/clk  
clock
```

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)
[read_tcf](#)

Affected by these attributes: (net) [lp_asserted_probability](#) on page 1389
 (pin) [lp_asserted_probability](#) on page 1380
 (port) [lp_asserted_probability](#) on page 1396
 (subport) [lp_asserted_probability](#) on page 1411
 (design) [lp_default_probability](#) on page 1361

(instance) [lp_default_probability](#) on page 1376

Related attributes: (net) [lp_probability_type](#) on page 1393

(port) [lp_probability_type](#) on page 1402

(subport) [lp_probability_type](#) on page 1415

lp_toggle_rate_type

`lp_toggle_rate_type {asserted | clock | computed | constant | default}`

Read-only [pin](#) attribute. Computed attribute. Gives an indication of the source of the value of the toggle rate value. This attribute can have the following values:

asserted	Indicates that the pin value is user-specified. You either specified the value through the <code>lp_asserted_toggle_rate</code> attribute or in a switching activity file (TCF or SAIF file).
clock	Indicates that the pin is a clock pin and that the value is derived from the clock waveform.
computed	Indicates that the pin value is computed by propagating the internal switching activities.
constant	Indicates that the pin is driven by a constant value. In this case, the value of the <code>lp_asserted_toggle_rate</code> attribute is set to 0.
default	Indicates that the pin value is not user-specified, but determined by the value of the <code>lp_default_toggle_rate</code> attribute.

Example

The following example checks the source of the probability value of a clock pin on a register.

```
rc:/> get_att lp_toggle_rate_type /des*/alu/*/aluout_reg[2]/pins_in/clk  
clock
```

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)
 [read_tcf](#)

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Pin Attributes

Affected by these attributes:	(net) lp asserted toggle rate on page 1390 (pin) lp asserted toggle rate on page 1381 (port) lp asserted toggle rate on page 1396 (subport) lp asserted toggle rate on page 1412 (design) lp default toggle rate on page 1363 (instance) lp default toggle rate on page 1377 lp toggle rate unit on page 1346
Related attributes:	(net) lp toggle rate type on page 1394 (port) lp toggle rate type on page 1403 (subport) lp toggle rate type on page 1416

Net Attributes

Contain information about a net in the specified design. A net can be a top-level net or can belong to a hierarchical instance.

- To set a net attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -net instance/net]
```

- To get a net attribute value, type

```
get_attribute attribute_name [find /des*/design -net instance/net]
```

Note: You may need to specify several instances to uniquely identify the net.

lp_asserted_probability

`lp_asserted_probability float`

Read-write net attribute. Specifies the probability value of this net being high for power estimation. You can specify any value between 0 and 1.

Related Information

Sources of Switching Activity Information in Low Power in Encounter RTL Compiler

Set by one of these commands: [read_saif](#)

[read_tcf](#)

Affects these commands: [report_gates](#) -power

[report_power](#)

Related attributes: (pin) [lp_asserted_probability](#) on page 1380

(port) [lp_asserted_probability](#) on page 1396

(subport) [lp_asserted_probability](#) on page 1411

lp_asserted_toggle_rate

`lp_asserted_toggle_rate float`

Read-write net attribute. Specifies the toggle rate (toggle count per toggle rate unit) of this net for the purpose of power estimation. You can specify any positive value, including zero.

Related Information

Sources of Switching Activity Information in Low Power in Encounter RTL Compiler

Set by one of these commands: [read_saif](#)

[read_tcf](#)

Affects these commands: [report_gates -power](#)

[report_power](#)

Affected by this attribute: [lp_toggle_rate_unit](#) on page 1346

Related attributes: (pin) [lp_asserted_toggle_rate](#) on page 1381

(port) [lp_asserted_toggle_rate](#) on page 1396

(subport) [lp_asserted_toggle_rate](#) on page 1412

lp_computed_probability

`lp_computed_probability float`

Read-only net attribute. Retrieves the probability of the net. The probability can be asserted or computed.

Example

In the following example the net is driven by a clock port. The probability therefore can be derived from the duty cycle of the clock (which is 50 percent by default).

```
rc:/> define_clock -period 3500 -name clock -design /designs/alu
...
rc:/> get_att lp_computed_probability */nets/clock
0.50000
```

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Net Attributes

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)

[read_tcf](#)

Related commands: [report_gates -power](#)

[report_power](#)

Affected by these attributes: (net) [lp_asserted_probability](#) on page 1389
(pin) [lp_asserted_probability](#) on page 1380
(port) [lp_asserted_probability](#) on page 1396
(subport) [lp_asserted_probability](#) on page 1411
(design) [lp_default_probability](#) on page 1361
(instance) [lp_default_probability](#) on page 1376

Related attributes: (net) [lp_computed_probability](#) on page 1390
(port) [lp_computed_probability](#) on page 1397
(subport) [lp_computed_probability](#) on page 1412

lp_computed_toggle_rate

`lp_computed_toggle_rate float`

Read-only `net` attribute. Retrieves the toggle rate (toggle count per toggle rate unit) of this net for power estimation. The toggle rate can be asserted or computed.

Example

In the following example the net is driven by a clock port. The toggle rate can therefore be derived from the clock cycle.

```
rc:/> define_clock -period 3500 -name clock -design /designs/alu
...
rc:/> get_att lp_computed_toggle_rate */nets/clock
0.571429
```

Because the default toggle rate unit is /ns and the period is expressed in ps, the toggle rate is determined by the reverse of the clock period times 2 (because the signal toggles two times per period) or 2/3.5ns.

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Net Attributes

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)

[read_tcf](#)

Related commands: [report_gates -power](#)

[report_power](#)

Affected by these attributes:

- (net) [lp_asserted_toggle_rate](#) on page 1390
- (pin) [lp_asserted_toggle_rate](#) on page 1381
- (port) [lp_asserted_toggle_rate](#) on page 1396
- (subport) [lp_asserted_toggle_rate](#) on page 1412
- (design) [lp_default_toggle_rate](#) on page 1363
- (instance) [lp_default_toggle_rate](#) on page 1377

Related attributes:

- (pin) [lp_computed_toggle_rate](#) on page 1382
- (port) [lp_computed_toggle_rate](#) on page 1398
- (subport) [lp_computed_toggle_rate](#) on page 1413

lp_net_power

`lp_net_power float`

Read-only `net` attribute. Computes the switching power dissipated on the net. The unit of the power value is determined by the value of the `lp_power_unit` attribute.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands: [report_gates -power](#)

[report_power](#)

Affected by this attribute: [lp_power_unit](#) on page 1345

Related attributes:	(design) lp_net_power on page 1364
	(instance) lp_net_power on page 1379
	(net) lp_net_power on page 1392
	(port) lp_net_power on page 1401
	(subport) lp_net_power on page 1414

lp_probability_type

`lp_probability_type {asserted | clock | computed | constant | default}`

Read-only net attribute. Computed attribute. Gives an indication of the source of the value of the probability value. This attribute can have the following values:

asserted	Indicates that the net value is user-specified. You either specified the value through the <code>lp_asserted_probability</code> attribute or in a switching activity file (TCF or SAIF file).
clock	Indicates that the net is a clock net and that the value is derived from the clock waveform.
computed	Indicates that the net value is computed by propagating the internal switching activities.
constant	Indicates that the net is driven by a constant value. In this case, the value of the <code>lp_asserted_probability</code> attribute is set to 1 if it is driven by a logic 1, or 0 if it is driven by a logic 0.
default	Indicates that the net value is not user-specified, but determined by the value of the <code>lp_default_probability</code> attribute.

Example

The following example checks the source of the probability value of a net driven by a clock.

```
rc:/> get_att lp_probability_type */nets/clock  
clock
```

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Net Attributes

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)

[read_tcf](#)

Affected by these attributes: (net) [lp_asserted_probability](#) on page 1389

(pin) [lp_asserted_probability](#) on page 1380

(port) [lp_asserted_probability](#) on page 1396

(subport) [lp_asserted_probability](#) on page 1411

(design) [lp_default_probability](#) on page 1361

(instance) [lp_default_probability](#) on page 1376

Related attributes: (pin) [lp_probability_type](#) on page 1386

(port) [lp_probability_type](#) on page 1402

(subport) [lp_probability_type](#) on page 1415

lp_toggle_rate_type

`lp_toggle_rate_type {asserted | clock | computed | constant | default}`

Read-only [net](#) attribute. Computed attribute. Gives an indication of the source of the value of the toggle rate value. This attribute can have the following values:

asserted	Indicates that the net value is user-specified. You either specified the value through the <code>lp_asserted_toggle_rate</code> attribute or in a switching activity file (TCF or SAIF file).
clock	Indicates that the net is a clock net and that the value is derived from the clock waveform.
computed	Indicates that the net value is computed by propagating the internal switching activities.
constant	Indicates that the net is driven by a constant value. In this case, the value of the <code>lp_asserted_toggle_rate</code> attribute is set to 0.
default	Indicates that the net value is not user-specified, but determined by the value of the <code>lp_default_toggle_rate</code> attribute.

Example

The following example checks the source of the toggle rate value of a net driven by a clock.

```
rc:/> get_att lp_toggle_rate_type */nets/clock  
clock
```

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)

[read_tcf](#)

Affected by these attributes:

(net) [lp_asserted_toggle_rate](#) on page 1390
(pin) [lp_asserted_toggle_rate](#) on page 1381
(port) [lp_asserted_toggle_rate](#) on page 1396
(subport) [lp_asserted_toggle_rate](#) on page 1412
(design) [lp_default_toggle_rate](#) on page 1363
(instance) [lp_default_toggle_rate](#) on page 1377
[lp_toggle_rate_unit](#) on page 1346

Related attributes:

(pin) [lp_toggle_rate_type](#) on page 1387
(port) [lp_toggle_rate_type](#) on page 1403
(subport) [lp_probability_type](#) on page 1415

Port Attributes

Contain information about single bit connection points in the specified top-level design. Port objects are stored in `ports_in` and `ports_out` directories.

- To set a port attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -port name]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -port name]
```

lp_asserted_probability

`lp_asserted_probability float`

Read-write `port` attribute. Specifies the probability value of a signal on this port being high for power estimation. You can specify any value between 0 and 1.

Related Information

Sources of Switching Activity Information in Low Power in Encounter RTL Compiler

Set by one of these commands: [read_saif](#)

[read_tcf](#)

Affects these commands: [report_gates -power](#)

[report_power](#)

Related attributes: (net) [lp_asserted_probability](#) on page 1389

(pin) [lp_asserted_probability](#) on page 1380

(subport) [lp_asserted_probability](#) on page 1411

lp_asserted_toggle_rate

`lp_asserted_toggle_rate float`

Read-write `port` attribute. Specifies the toggle rate (toggle count per toggle rate unit) of this port for the purpose of power estimation. You can specify any positive value, including zero.

Related Information

Sources of Switching Activity Information in Low Power in Encounter RTL Compiler

Set by one of these commands: [read_saif](#)

[read_tcf](#)

Affects these commands: [report_gates -power](#)

[report_power](#)

Affected by this attribute: [lp_toggle_rate unit](#) on page 1346

Related attributes: (net) [lp_asserted_toggle_rate](#) on page 1390

((pin) [lp_asserted_toggle_rate](#) on page 1381

(subport) [lp_asserted_toggle_rate](#) on page 1412

lp_computed_probability

`lp_computed_probability float`

Read-only port attribute. Retrieves the probability of the port. The probability can be asserted or computed.

Example

In the following example a clock is associated with the port. The probability therefore can be derived from the duty cycle of the clock (which is 50 percent by default).

```
rc:/> define_clock -period 3500 clock  
/designs/alu/timing/clock_domains/domain_1/clock  
rc:/> get_att lp_computed_probability */ports_in/clock  
0.50000
```

Related Information

Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler

Affected by these commands: [read_saif](#)

[read_tcf](#)

Related commands: [report_gates -power](#)

[report_power](#)

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Port Attributes

Affected by these attributes:

- (net) [lp_asserted_probability](#) on page 1389
- (pin) [lp_asserted_probability](#) on page 1380
- (port) [lp_asserted_probability](#) on page 1396
- (subport) [lp_asserted_probability](#) on page 1411
- (design) [lp_default_probability](#) on page 1361
- (instance) [lp_default_probability](#) on page 1376

Related attributes:

- (net) [lp_computed_probability](#) on page 1390
- (pin) [lp_computed_probability](#) on page 1381
- (subport) [lp_computed_probability](#) on page 1412

lp_computed_toggle_rate

`lp_computed_toggle_rate float`

Read-only port attribute. Retrieves the toggle rate (toggle count per toggle rate unit) of this port for power estimation. The toggle rate can be asserted or computed.

Example

In the following example the net is driven by a clock port. The toggle rate can therefore be derived from the clock cycle.

```
rc:/> define_clock -period 3500 clock  
/designs/alu/timing/clock_domains/domain_1/clock  
rc:/> get_att lp_computed_toggle_rate */ports_in/clock  
0.571429
```

Because the default toggle rate unit is /ns and the period is expressed in ps, the toggle rate is determined by the reverse of the clock period times 2 (because the signal toggles two times per period) or 2/3.5ns.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)
 [read_tcf](#)

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Port Attributes

Related commands:	report gates -power report power
Affected by these attributes:	(net) lp asserted toggle rate on page 1390 (pin) lp asserted toggle rate on page 1381 (port) lp asserted toggle rate on page 1396 (subport) lp asserted toggle rate on page 1412 (design) lp default toggle rate on page 1363 (instance) lp default toggle rate on page 1377
Related attributes:	(net) lp computed toggle rate on page 1391 (pin) lp computed toggle rate on page 1382 (subport) lp computed toggle rate on page 1413

lp_default_probability

`lp_default_probability float`

Read-only port attribute. Returns the default probability of the port. The value is determined as follows:

1. If the port is asserted, it returns the value of the `lp asserted probability` port attribute.
2. If the port is not asserted, it returns the value of the `lp default probability` root attribute.

Note: This attribute can only have a value for primary input ports.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands:	report gates -power report power
Affected by these attributes:	(port) lp asserted probability on page 1396 (root) lp default probability on page 1341 lp power unit on page 1345

lp_default_toggle_rate

`lp_default_toggle_rate float`

Read-only port attribute. Returns the default toggle rate of the port. The value is determined as follows:

1. If the port is asserted, it returns the value of the `lp_asserted_toggle_rate` port attribute.
2. If the port is not asserted, the default value will be computed based on the `lp_default_toggle_percentage` attribute and the associated clock information.
3. If the port is not asserted and the `lp_default_toggle_percentage` attribute is not set, it returns the value of the `lp_default_toggle_rate` root attribute.

Note: This attribute can only have a value for primary input ports.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands: [report gates -power](#)

[report power](#)

Affected by these attributes: (port) [lp_asserted_toggle_rate](#) on page 1396

 (root) [lp_default_toggle_rate](#) on page 1342

[lp_power_unit](#) on page 1345

lp_net_power

`lp_net_power float`

Read-only port attribute. Computes the switching power dissipated on the net connected to this port. The unit of the power value is determined by the value of the `lp_power_unit` attribute.

Related Information

Reporting on Specific Power Components for Specific Objects in *Low Power in Encounter RTL Compiler*

Related commands:	report gates -power report power
Affected by this attribute:	lp_power_unit on page 1345
Related attributes:	(design) lp_net_power on page 1364 (instance) lp_net_power on page 1379 (net) lp_net_power on page 1392 (pin) lp_net_power on page 1385 (subport) lp_net_power on page 1414

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Port Attributes

lp_probability_type

lp_probability_type {asserted | clock | computed | constant | default}

Read-only port attribute. Computed attribute. Gives an indication of the source of the value of the probability value. This attribute can have the following values:

asserted	Indicates that the port value is user-specified. You either specified the value through the lp_asserted_probability attribute or in a switching activity file (TCF or SAIF file).
clock	Indicates that the port is connected to a clock net and that the value is derived from the clock waveform.
computed	Indicates that the port value is computed by propagating the internal switching activities.
constant	Indicates that the port is driven by a constant value. In this case, the value of the lp_asserted_probability attribute is set to 1 if it is driven by a logic 1, or 0 if it is driven by a logic 0.
default	Indicates that the port value is not user-specified, but determined by the value of the lp_default_probability attribute.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)

[read_tcf](#)

Affected by these attributes: (net) [lp_asserted_probability](#) on page 1389

(pin) [lp_asserted_probability](#) on page 1380

(port) [lp_asserted_probability](#) on page 1396

(subport) [lp_asserted_probability](#) on page 1411

(design) [lp_default_probability](#) on page 1361

(instance) [lp_default_probability](#) on page 1376

Related attributes: (net) [lp_probability_type](#) on page 1393

(pin) [lp_probability_type](#) on page 1386

(subport) [lp_probability_type](#) on page 1415

lp_toggle_rate_type

`lp_toggle_rate_type {asserted | clock | computed | constant | default}`

Read-only port attribute. Computed attribute. Gives an indication of the source of the value of the toggle rate value. This attribute can have the following values:

asserted	Indicates that the port value is user-specified. You either specified the value through the <code>lp_asserted_toggle_rate</code> attribute or in a switching activity file (TCF or SAIF file).
clock	Indicates that the port is connected to a clock net and that the value is derived from the clock waveform.
computed	Indicates that the port value is computed by propagating the internal switching activities.
constant	Indicates that the port is driven by a constant value. In this case, the value of the <code>lp_asserted_toggle_rate</code> attribute is set to 0.
default	Indicates that the port value is not user-specified, but determined by the value of the <code>lp_default_toggle_rate</code> attribute.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)

[read_tcf](#)

Affected by these attributes:

(net) [lp_asserted_toggle_rate](#) on page 1390

(pin) [lp_asserted_toggle_rate](#) on page 1381

(port) [lp_asserted_toggle_rate](#) on page 1396

(subport) [lp_asserted_toggle_rate](#) on page 1412

(design) [lp_default_toggle_rate](#) on page 1363

(instance) [lp_default_toggle_rate](#) on page 1377

[lp_toggle_rate_unit](#) on page 1346

Related attributes:

(net) [lp_toggle_rate_type](#) on page 1394

(pin) [lp_toggle_rate_type](#) on page 1387

(subport) [lp_probability_type](#) on page 1415

Subdesign Attributes

Contain information about the subdesigns in the specified design. Subdesigns correspond to Verilog modules or VHDL entities instantiated in the top-level Verilog module or top-level VHDL entity.

lp_clock_gating_add_reset

`lp_clock_gating_add_reset {false | true}`

Default: false

Read-write subdesign attribute. Determines whether the integrated clock-gating cell must contain reset logic added to the glitch removing logic.

- If you set this attribute to `true`, and the gated registers have an asynchronous reset or set pin, the low power (RC-LP) engine only uses an integrated clock-gating cell that has a pin with a `clock_gate_reset_pin` attribute defined.

If the RC-LP engine finds such a cell, it adds an `a_rst` port to the clock-gating module and internally connects this port to the reset pin of the integrated clock-gating cell.

If `c1, c2, ..., cn` represent the asynchronous reset signals for each of the registers controlled by this clock-gating instance, and `p1, p2, ..., pn` represent the asynchronous set signals for each of the registers controlled by this clock-gating instance, the RC-LP engine determines the reset signal for the clock-gating instance as:

$(c1+p1) \text{ AND } (c2+p2) \text{ AND } \dots \text{ AND } (cn+pn)$

The RC-LP engine creates the corresponding logic and connects the output of this logic to the `a_rst` pin of the clock-gating instance.

Setting the subdesign attribute to `true`, automatically sets the `lp_clock_gating_add_reset` attribute to `true` on all flops in the subdesign.

Note: If you set this attribute to `true`, but the gated registers do not have an asynchronous reset or set pin, no reset logic will be added.

- If you set this attribute to `false`, the RC-LP engine does not add an `a_rst` port to the clock-gating module, even when the gated registers have asynchronous set or reset pins.

Note: This attribute applies to all instances of this subdesign (module).

Related Information

[Clock Gating Cell Specification](#) in the *Library Guide for Encounter RTL Compiler*.

[Using Attributes to Select an Integrated Clock-Gating Cell in the Technology Library in Low Power in Encounter RTL Compiler](#)

Affects these commands:	synthesize report clock gating
Affected by this attribute:	(design) lp_clock_gating_add_reset on page 1350
Affects this attribute:	(instance) lp_clock_gating_add_reset on page 1369
Related attribute:	polarity on page 223

lp_clock_gating_auto_path_adjust

`lp_clock_gating_auto_path_adjust {inherited | none | fixed | variable}`

Read-write [subdesign](#) attribute. Controls the automatic timing adjustment on the clock gate enable paths in this subdesign. You can specify any of the following values:

fixed	Specifies to use the user-defined path adjust value.
inherited	Inherits the value from the design attribute.
none	Prevents the automatic timing adjustment.
variable	Specifies to scale the tool-calculated path adjust values.

Related Information

[Tightening Timing Constraints on the Clock-Gating Enables in Low Power in Encounter RTL Compiler](#)

Affects these commands:	synthesize report timing
Affected by this attribute:	(design) lp_clock_gating_auto_path_adjust on page 1352
Affects these attributes:	(subdesign) lp_clock_gating_auto_path_adjust_fixed_delay on page 1352 (subdesign) lp_clock_gating_auto_path_adjust_multiplier on page 1406

lp_clock_gating_auto_path_adjust_fixed_delay

`lp_clock_gating_auto_path_adjust_fixed_delay float`

Read-write subdesign attribute. Specifies a user-defined path adjust value (in picoseconds) for the enable pins of all clock-gating instances in this subdesign and its subdesigns. This value overrides the tool-computed path adjust values. You must specify a positive value. A negative or null value does not affect the path constraints. If you do not set this attribute, it inherits the value from the design attribute.

The path_adjust timing exceptions are stored in the following directory:

`/designs/top/timing/exceptions/path_adjusts/`

The tool-computed path_adjust timing exceptions are calculated based on information in the libraries and have the following name:

`_auto_xxps_cg_path_adjust`

where `xx` is the user-provided delay value.

Related Information

[Tightening Timing Constraints on the Clock-Gating Enables in Low Power in Encounter RTL Compiler](#)

Affects these commands:

[synthesize](#)

[report timing](#)

Affected by these attributes:

(subdesign) [lp_clock_gating_auto_path_adjust](#) on page 1352

(design) [lp_clock_gating_auto_path_adjust_fixed_delay](#) on page 1352

lp_clock_gating_auto_path_adjust_multiplier

`lp_clock_gating_auto_path_adjust_multiplier float`

Read-write subdesign attribute. Scales the tool-computed path adjust values of the enable pins of the clock-gating instances in this subdesign and its subdesigns. Specify a real number between 0.0 and infinity.

The path_adjust timing exceptions are stored in the following directory:

`/designs/top/timing/exceptions/path_adjusts/`

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Subdesign Attributes

The tool-computed path_adjust timing exceptions are calculated based on information in the libraries and have the following name:

_auto_XXps_cg_path_adjust

where XX is the magnitude of the adjustment.

Related Information

[Tightening Timing Constraints on the Clock-Gating Enables in Low Power in Encounter RTL Compiler](#)

Affects these commands: [synthesize](#)

[report timing](#)

Affected by these attributes: ([subdesign](#)) [lp_clock_gating_auto_path_adjust](#) on page 1352

 ([design](#)) [lp_clock_gating_auto_path_adjust_multiplier](#) on page 1354

lp_clock_gating_cell

`lp_clock_gating_cell path_name_for_cell`

Read-write [subdesign](#) attribute. Specifies the path name of the cell to be used for clock-gating insertion. The path name can contain the wildcard character (*). This attribute overrides the following attributes: [lp_clock_gating_add_obs_port](#), [lp_clock_gating_add_reset](#), [lp_clock_gating_control_point](#), and [lp_clock_gating_style](#).

If the specified cell does not exist in any of the libraries, the auto clock-gating insertion will fail. If multiple cells are found, the attribute is not set and the tool reports an error.

Note: This attribute applies to all instances of this subdesign (module).

Examples

The following examples sets CGIC_LPP as the clock-gating cell to use:

```
rc:/designs/top> set_att lp_clock_gating_cell /libr*/cg/libcells/CGIC_LPP \
[find / -subdesign bot]
Setting attribute of subdesign bot: 'lp_clock_gating_cell' =
/libraries/cg/libcells/CGIC_LPP
rc:/designs/top> set_att lp_clock_gating_cell [find / -libcell CGIC_LPP] \
[find / -subdesign bot]
Setting attribute of subdesign bot: 'lp_clock_gating_cell' =
/libraries/cg/libcells/CGIC_LPP
```

Related Information

[Specifying the Library Cell Name of the Integrated Clock-Gating Cell in Low Power in Encounter RTL Compiler](#).

Affects these commands:	<u>synthesize</u> <u>report clock gating</u>
Affects these attributes:	<u>lp_clock_gating_add_obs_port</u> on page 1349 <u>lp_clock_gating_add_reset</u> on page 1350 (instance) <u>lp_clock_gating_cell</u> on page 1370 <u>lp_clock_gating_control_point</u> on page 1355 <u>lp_clock_gating_style</u> on page 1359
Related attribute:	(design) <u>lp_clock_gating_cell</u> on page 1354

lp_clock_gating_exclude

`lp_clock_gating_exclude {false | true}`

Default: false

Read-write subdesign attribute. Determines whether to insert clock-gating logic for this subdesign (module). If you set this attribute to `true`, no clock-gating logic is added to this module and all its submodules.

Note: This attribute applies to all instances of this subdesign (module).

Example

Assume that design `top` contains a subdesign (module) `sub` that has been instantiated five times in the design. The instance names are `i1, i2, i3, i4, and i5`. Now you want to exclude clock gating from subdesign `sub`:

```
set_attribute lp_clock_gating_exclude true /designs/top/subdesigns/sub
```

This attribute setting excludes all instances of `sub` from clock gating. To prevent instances `i1` and `i2` from being excluded from clock gating, you need to create a separate subdesign for instances `i1` and `i2`:

```
edit_netlist dedicate_subdesign {/designs/top/instances_hier/i1 \
/designs/top/instances_hier/i2}
```

Related Information

[Controlling Insertion of Clock-Gating Logic in Low Power in Encounter RTL Compiler](#)

Affects this command:	<u>synthesize</u>
Related command:	<u>edit netlist dedicate subdesign</u>
Related attributes:	(design) <u>lp_clock_gating_exclude</u> on page 1357 (instance) <u>lp_clock_gating_exclude</u> on page 1371

lp_clock_gating_module

`lp_clock_gating_module path_name_for_module`

Read-write subdesign attribute. Specifies the path to the module that defines the customized clock-gating logic to be used for this subdesign.

Example

The following example specifies to use module `my(CG_MOD1` to gate the flops belonging to all instances corresponding to the subdesigns starting with `mid*`.

```
set_attr lp_clock_gating_module designs/my(CG_MOD1 [find / -subdesign mid*]
```

Related Information

[Defining a Clock-Gating Module in Low Power in Encounter RTL Compiler](#)

Related attributes:	(design) <u>lp_clock_gating_module</u> on page 1359 (instance) <u>lp_clock_gating_module</u> on page 1374
---------------------	--

lp_clock_gating_test_signal

`lp_clock_gating_test_signal string`

Read-write subdesign attribute. Indicates which test signal to connect to the test pin of the clock-gating instances in all instances of the specified subdesign.

Note: The RC-LP engine creates a separate subdesign for the clock-gating logic in each clock-gating domain, consequently the clock-gating instance names are similar to

```
/designs/design/instances_hier/*/RC(CG_HIER_INSTx
```

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Subdesign Attributes

You can specify one of the following values:

<code>use_shift_enable</code>	Indicates to use the shift-enable signal of the scan chain to which the gated flip-flops belong. If the gated flip-flops belong to different scan chains with different shift enable signals, the shift enable signals are OR-ed and the output of the OR-tree is used as the clock-gating test signal.
-------------------------------	--

Note: You can only use this value when you insert clock-gating in a mapped netlist *after* you have inserted the scan chains.

<code>test_signal_object</code>	Indicates to use the specified test signal object.
---------------------------------	--

Note: Test signals are defined using the `define_dft test_mode` or `define_dft shift_enable` commands. The corresponding object is stored at `/designs/design/dft/test_signals`.

Example

The following command instructs to connect the test pins of all clock-gating instances in the subdesign `mid` to test signal `tm`.

```
set_attr lp_clock_gating_test_signal /designs/alu/dft/test_signals/tm \
[find . -subdesign mid]
```

Related Information

[Clock Gating with DFT in Low Power in Encounter RTL Compiler](#).

Affects this command:	<u>synthesize</u>
Related attributes:	(design) <u>lp_clock_gating_test_signal</u> on page 1360 (instance) <u>lp_clock_gating_test_signal</u> on page 1375

Subport Attributes

Contain information about subports in the specified design. A subport is a single bit connection point within a hierarchical instance, that is a module or entity that has been instantiated.

- To set a subport attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subport hier_instance/name]
```

- To get a subport attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subport hier_instance/name]
```

Note: You may need to specify more than one (hierarchical) instance to serve as the search path that will uniquely identify the relevant subport.

lp_asserted_probability

`lp_asserted_probability float`

Read-write subport attribute. Specifies the probability value of a signal on this subport being high for power estimation. You can specify any value between 0 and 1.

Related Information

Sources of Switching Activity Information in Low Power in Encounter RTL Compiler

Set by one of these commands: [read_saif](#)

[read_tcf](#)

Affects these commands: [report_gates_power](#)

[report_power](#)

Affects these commands: [report_gates_power](#)

[report_power](#)

Related attributes: (net) [lp_asserted_probability](#) on page 1389

(pin) [lp_asserted_probability](#) on page 1380

(port) [lp_asserted_probability](#) on page 1396

lp_asserted_toggle_rate

`lp_asserted_toggle_rate float`

Read-write subport attribute. Specifies the toggle rate (toggle count per toggle rate unit) of this subport for the purpose of power estimation. You can specify any positive value, including zero.

Related Information

Sources of Switching Activity Information in Low Power in Encounter RTL Compiler

Set by one of these commands: [read_saif](#)

[read_tcf](#)

Affects these commands: [report_gates-power](#)

[report_power](#)

Affected by this attribute: [lp_toggle_rate_unit](#) on page 1346

Related attributes: (net) [lp_asserted_toggle_rate](#) on page 1390

(pin) [lp_asserted_toggle_rate](#) on page 1381

(port) [lp_asserted_toggle_rate](#) on page 1396

lp_computed_probability

`lp_computed_probability float`

Read-only subport attribute. Retrieves the probability of the subport. The probability can be asserted or computed.

Related Information

Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler

Affected by these commands: [read_saif](#)

[read_tcf](#)

Related commands: [report_gates-power](#)

[report_power](#)

Affected by these attributes:	(net) lp asserted probability on page 1389 (pin) lp asserted probability on page 1380 (port) lp asserted probability on page 1396 (subport) lp asserted probability on page 1411 (design) lp default probability on page 1361 (instance) lp default probability on page 1376
Related attributes:	(net) lp computed probability on page 1390 (pin) lp computed probability on page 1381 (port) lp computed probability on page 1397

lp_computed_toggle_rate

`lp_computed_toggle_rate float`

Read-only subport attribute. Retrieves the toggle rate (toggle count per toggle rate unit) of this subport for power estimation. The toggle rate can be asserted or computed.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands:	read_saif read_tcf
Related commands:	report_gates -power report power
Affected by these attributes:	(net) lp asserted toggle rate on page 1390 (pin) lp asserted toggle rate on page 1381 (port) lp asserted toggle rate on page 1396 (subport) lp asserted toggle rate on page 1412 (design) lp default toggle rate on page 1363 (instance) lp default toggle rate on page 1377

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Subport Attributes

Related attributes:	(net) lp_computed_toggle_rate on page 1391
	(pin) lp_computed_toggle_rate on page 1382
	(port) lp_computed_toggle_rate on page 1398

lp_net_power

`lp_net_power float`

Read-only subport attribute. Computes the switching power dissipated on the net connected to this subport. The unit of the value is determined by the value of the `lp_power_unit` attribute.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Related commands:	report gates -power report power
Affected by this attribute:	lp_power_unit on page 1345
Related attributes:	(design) lp_net_power on page 1364 (instance) lp_net_power on page 1379 (net) lp_net_power on page 1392 (port) lp_net_power on page 1401 (subport) lp_net_power on page 1414

lp_probability_type

`lp_probability_type {asserted | clock | computed | constant | default}`

Read-only subport attribute. Computed attribute. Gives an indication of the source of the value of the probability value. This attribute can have the following values:

asserted	Indicates that the value is user-specified. You either specified the value through the <code>lp_asserted_probability</code> attribute or in a switching activity file (TCF or SAIF file).
clock	Indicates that the subport is connected to a clock net and that the value is derived from the clock waveform.
computed	Indicates that the value is computed by propagating the internal switching activities.
constant	Indicates that the subport is driven by a constant value. In this case, the value of the <code>lp_asserted_probability</code> attribute is set to 1 if it is driven by a logic 1, or 0 if it is driven by a logic 0.
default	Indicates that the value is not user-specified, but determined by the value of the <code>lp_default_probability</code> attribute.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)

[read_tcf](#)

Affected by these attributes: [\(net\) lp_asserted_probability](#) on page 1389

[\(pin\) lp_asserted_probability](#) on page 1380

[\(port\) lp_asserted_probability](#) on page 1396

[\(subport\) lp_asserted_probability](#) on page 1411

[\(design\) lp_default_probability](#) on page 1361

[\(instance\) lp_default_probability](#) on page 1376

Related attributes: [\(net\) lp_probability_type](#) on page 1393

[\(pin\) lp_probability_type](#) on page 1386

[\(port\) lp_probability_type](#) on page 1402

lp_toggle_rate_type

`lp_toggle_rate_type {asserted | clock | computed | constant | default}`

Read-only subport attribute. Computed attribute. Gives an indication of the source of the value of the toggle rate value. This attribute can have the following values:

asserted	Indicates that the value is user-specified. You either specified the value through the <code>lp_asserted_toggle_rate</code> attribute or in a switching activity file (TCF or SAIF file).
clock	Indicates that the subport is connected to a clock net and that the value is derived from the clock waveform.
computed	Indicates that the value is computed by propagating the internal switching activities.
constant	Indicates that the subport is driven by a constant value. In this case, the value of the <code>lp_asserted_toggle_rate</code> attribute is set to 0.
default	Indicates that the value is not user-specified, but determined by the value of the <code>lp_default_toggle_rate</code> attribute.

Related Information

[Reporting on Specific Power Components for Specific Objects in Low Power in Encounter RTL Compiler](#)

Affected by these commands: [read_saif](#)
[read_tcf](#)

Affected by these attributes: (net) [lp_asserted_toggle_rate](#) on page 1390
(pin) [lp_asserted_toggle_rate](#) on page 1381
(port) [lp_asserted_toggle_rate](#) on page 1396
(subport) [lp_asserted_toggle_rate](#) on page 1412
(design) [lp_default_toggle_rate](#) on page 1363
(instance) [lp_default_toggle_rate](#) on page 1377
[lp_toggle_rate_unit](#) on page 1346

Related attributes: (net) [lp_toggle_rate_type](#) on page 1394
(pin) [lp_toggle_rate_type](#) on page 1387
(port) [lp_toggle_rate_type](#) on page 1403

Clock Attributes

Contain information about clock objects in the specified design.

- To set a `clock` attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -clock myclock]
```

- To get a `clock` attribute value, type

```
get_attribute attribute_name [find /des*/design -clock myclock]
```

lp_default_toggle_percentage

```
lp_default_toggle_percentage float
```

Default: inherited

Read-write `clock` attribute. Specifies the multiplication factor to be used with this clock to modify the `default` toggle rate of any activity propagation starting point associated with clock information. The toggle rate of an activity propagation starting point is derived by multiplying this scaling factor with the toggle rate of the associated clock information. If multiple clocks are related to the output pin, the fastest clock is used. Specify a value between 0 and 1. You can use a value greater than 1.0 to model cases with high glitching power which could result in more than one value change per clock on average. A warning is printed if you specify a value larger than 1.

By default, this attribute inherits the value from the corresponding design attribute.

Example

The following example compares the toggle rate of the output of a flip-flop before and after the multiplication factor was specified for the clock triggering the flip-flop.

```
rc:/> set_attr lp_power_analysis_effort low /
Setting attribute of root '/': 'lp_power_analysis_effort' = low
rc:/> define_clock -period 1000 [clock_ports]
/designs/top/timing/clock_domains/domain_1/c
rc:/> get_att lp_computed_toggle_rate ports_in/c
2.000000
rc:/> get_att lp_computed_toggle_rate /designs/top/instances_seq/g1/pins_out/Q
0.020000
rc:/> set_attr lp_default_toggle_percentage 0.2 [find / -clock *]
Setting attribute of clock 'c': 'lp_default_toggle_percentage' = 0.200000
rc:/> get_att lp_computed_toggle_rate /designs/top/instances_seq/g1/pins_out/Q
0.400000
rc:/> get_att lp_toggle_rate_type /designs/top/instances_seq/g1/pins_out/Q
default
```

Attribute Reference for Encounter RTL Compiler

Low Power Synthesis—Clock Attributes

Related Information

[Using Default Switching Activities in Low Power in Encounter RTL Compiler](#)

Affects this command: [report power](#)

Affected by these attributes: (design) [lp_default_toggle_percentage](#) on page 1362
[lp_toggle_rate_unit](#) on page 1346

Advanced Low Power

Root Attributes

- [clp_ignore_ls_high_to_low](#) on page 1424
- [clp_treat_errors_as_warnings](#) on page 1424

Library Domain Attributes

- [active_operating_conditions](#) on page 1426
- [default](#) on page 1427
- [library](#) on page 1427
- [operating_conditions](#) on page 1429
- [power_library](#) on page 1429
- [wireload_selection](#) on page 1430

Level Shifter Group Attributes

- [direction](#) on page 1431
- [ground_direction](#) on page 1431
- [level_shifter_cells](#) on page 1432
- [max_input_voltage](#) on page 1433
- [max_output_voltage](#) on page 1433
- [min_input_voltage](#) on page 1435
- [min_output_voltage](#) on page 1435
- [valid_location](#) on page 1435

Libcell Attributes

- [aux_enables](#) on page 1436
- [valid_location](#) on page 1436

Design Attributes

- [base_mode](#) on page 1437
- [library_domain](#) on page 1437
- [lp_isolate_domain_crossing_constants](#) on page 1438
- [pi_relax_map_iso_cell_checks](#) on page 1438
- [pi_relax_map_ls_cell_checks](#) on page 1439
- [power_domain](#) on page 1439

Constant Attributes

- [power_domain](#) on page 1440

Port Attributes

- [isolation_rule](#) on page 1441
- [level_shifter_rule](#) on page 1441
- [power_domain](#) on page 1442

hdl_arch Attributes

- [library_domain](#) on page 1443

Subdesign Attributes

- [library_domain](#) on page 1444

Support Attributes

- [isolation_rule](#) on page 1446
- [level_shifter_rule](#) on page 1447

- [power_domain](#) on page 1447

Instance Attributes

- [library_domain](#) on page 1448
- [power_domain](#) on page 1448
- [secondary_domain](#) on page 1449
- [state_retention_rule](#) on page 1449

Pin Attributes

- [isolation_rule](#) on page 1450
- [level_shifter_rule](#) on page 1450
- [power_domain](#) on page 1451

Isolation Rule Attributes

- [cells](#) on page 1452
- [cpf_pins](#) on page 1452
- [enable_driver](#) on page 1453
- [enable_polarity](#) on page 1453
- [exclude_pins](#) on page 1453
- [location](#) on page 1454
- [off_domain](#) on page 1454
- [output_value](#) on page 1455
- [pins](#) on page 1455
- [prefix](#) on page 1455
- [secondary_domain](#) on page 1456
- [to_power_domain](#) on page 1456
- [within_hierarchy](#) on page 1456

Level Shifter Rule Attributes

- [cells](#) on page 1457
- [cpf_pins](#) on page 1457
- [direction](#) on page 1457
- [exclude_pins](#) on page 1458
- [from_power_domain](#) on page 1458
- [location](#) on page 1458
- [pins](#) on page 1459
- [prefix](#) on page 1459
- [to_power_domain](#) on page 1459

Nominal Condition Attributes

- [ground_voltage](#) on page 1461
- [library_set](#) on page 1462
- [voltage](#) on page 1462

Power Domain Attributes

- [base_domains](#) on page 1463
- [default](#) on page 1463
- [dft_iso_rule](#) on page 1464
- [has_external_shutoff](#) on page 1464
- [library_domain](#) on page 1465
- [library_domain_by_mode](#) on page 1465
- [shutoff_condition](#) on page 1466

Power Mode Attributes

- [constraint_mode](#) on page 1467
- [default](#) on page 1467

- [domain_conditions](#) on page 1467

State Retention Rule Attributes

- [cell_type](#) on page 1468
- [cells](#) on page 1468
- [restore](#) on page 1469
- [restore_phase](#) on page 1469
- [save](#) on page 1469
- [save_phase](#) on page 1469
- [secondary_domain](#) on page 1470

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

clp_ignore_ls_high_to_low

```
clp_ignore_ls_high_to_low {false | true}
```

Default: false

Read-write `root` attribute. Determines whether Conformal Low Power (CLP) should check for high to low voltage level shifting.

Related Information

[Enabling High to Low Voltage Level Shifting Checking in *Interfacing between Encounter RTL Compiler and Encounter Conformal*](#)

Affects this command: [verify_power_structure](#)

clp_treat_errors_as_warnings

```
clp_treat_errors_as_warnings error_message_list
```

Read-write `root` attribute. Forces Conformal Low Power (CLP) to treat the specified error message IDs as warnings.

Example

The following example specifies that CLP should consider all ISO error messages to be "Warning" messages.

```
rc:/> set_attribute clp_treat_errors_as_warning ISO*
```

This attribute setting forces the `write_do_clp` command to generate the following command in the dofile:

```
set rule handling ISO* -Warning
```

Attribute Reference for Encounter RTL Compiler

Advanced Low Power—Root Attributes

Related Information

[Controlling Severity of Messages](#) in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects these commands:

[check_cpf](#)

[verify_power_structure](#)

[write_do_clp](#)

Library Domain Attributes

Contain information about the libraries associated with a library domain. These attributes are read-write attributes.

- To set a `library_domain` attribute value, type

```
set_attribute attribute_name attribute_value \
[find /libraries -library_domain domain]
```

- To get a `library_domain` attribute value, type

```
get_attribute attribute_name \
[find /libraries -library_domain domain]
```

Note: These attributes are located at `/libraries/library_domains/domain`.

active_operating_conditions

`active_operating_conditions string`

Read-only `library_domain` attribute. Returns the complete path to the operating conditions that were set with the `operating_conditions` attribute and forces the tool to load the operating conditions if they were not loaded yet or if they became invalid. Any problems found while loading are reported.

If the operating conditions cannot be loaded, then the attribute returns an empty string. This is a computed attribute. Computed attributes are potentially very time consuming to process and not listed by the `ls` command by default.

Related Information

Affects these commands:	synthesize report
Affected by these attributes:	(<code>library_domain</code>) library on page 1427 (<code>library_domain</code>) operating_conditions on page 1429
Relate attribute:	(root) active_operating_conditions on page 928

default

```
default {false | true}
```

Default: false

Read-write library_domain attribute. Indicates whether the library domain is the default library domain. By default, the first library domain for which you specify the libraries, becomes the default library domain. Set this attribute to true for the desired domain. The tool automatically changes the value of the previous default library domain to false.

Example

```
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
rc:/> set_att library tutorial.lbr dom_2
Info      : Set default library domain. [LBR-109]
          : The default library domain is '/libraries/library_domains/dom_2'
Setting attribute of library_domain dom_2: 'library' = tutorial.lbr
rc:/> get_attribute default [find /libraries -library_domain dom_2]
true
rc:/> set_attribute library typical.lib dom_1
Setting attribute of library_domain dom_1: 'library' = typical.lib
rc:/> set_attribute default true [find /libraries -library_domain dom_1]
Info      : Set default library domain. [LBR-109]
          : The default domain changed from '/libraries/library_domains/dom_2' to
'/libraries/library_domains/dom_1'.

Setting attribute of library_domain dom_1: 'default' = true
rc:/> get_attribute default [find /libraries -library_domain dom_2]
false
```

If you read in a structural netlist, during elaboration RTL Compiler will link the cells used in the netlist to library cells of the *default* library domain.

Related Information

Related attributes: (mode) default on page 561
 (power_domain) default on page 1463

library

```
library {{lib [lib]...} [{lib [lib]...}]...}
```

Read-write library_domain attribute. Sets the target library for technology mapping for the specified library domain.

You must specify this attribute on the appropriate library domain when using a multi voltage design.

You can specify a single library or a Tcl list of library lists. Each library list is also a Tcl list. Each library must be found in the library search path (specified through the `lib_search_path` attribute). The first library in each list is considered the master library to which the content of the other libraries in that list is appended.

The `library` attribute also supports libraries that have been compressed with GNU Zip (`.gz` extension).

Note: The information in the appended libraries overwrites the corresponding information in the master library. However, RTL Compiler fails on loading the libraries if the delay models in the appended libraries differ from the delay models in the master library.

Examples

- To read independent libraries, type the following command:

```
set_attribute library {a.lib b.lib c.lib x.lbr y.lib} /lib*/lib*/dom_1
```

This command creates five libraries in the `/libraries/library_domains/dom_1` directory.

- To load one library and append some others, type the following command:

```
set_attribute library {{a.lib b.lib c.lib x.lbr y.lib}} /lib*/lib*/dom_1
```

This command creates one library in the `/libraries/library_domains/dom_1` directory; it loads the first library (`a.lib`) and appends the contents of the next libraries to the first one.

- To load several libraries and append some, type the following command:

```
set_attribute library {{a.lib b.lib} {c.lib} {x.lbr y.lib}}
```

This command creates three libraries in the `/libraries/library_domains/dom_1` directory: `a.lib`, `c.lib`, and `x.lbr`; it loads `a.lib`, appends `b.lib` to `a.lib`, then loads `c.lib`, followed by `x.lbr` and finally appends `y.lib` to `x.lbr`.

- To load a GZIP compressed library, type the following command:

```
set_attribute library big_15_lv.lib.gz /lib*/lib*/dom_1
```

Related Information

Affected by this command: [create_library_domain](#)

Affected by this attribute: [lib_search_path](#) on page 273

Related attribute: (root) [library](#) on page 274

Attribute Reference for Encounter RTL Compiler

Advanced Low Power—Library Domain Attributes

operating_conditions

`operating_conditions string`

Read-write `library_domain` attribute. Specifies the operating conditions to use for timing for the specified library domain. You must specify the path to an `operating_conditions` object in a library of this library domain. The `operating_conditions` attribute does not need to have a value if you want to use the default operating conditions.

Example

The following command sets the operating condition for library domain srpg to 1p08 .

```
set_attribute operating_condition 1p08 library_domains/srpg
```

Related Information

[Set Timing and Design Constraints in *Encounter RTL Compiler Synthesis Flows*](#)

Related attribute: (root) [operating_conditions](#) on page 537

power_library

`power_library domain`

Read-write `library_domain` attribute. Specifies the name of the library domain with which the libraries to be used for power analysis are associated.

Examples

In the following example two library domains are created for the design. The libraries associated with domain dom1 are the timing libraries of the design. The libraries associated with domain dom2 are the power libraries of the design. The last command indicates that the power libraries for domain dom1 are associated with domain dom2.

```
create_library_domain {dom1 dom2}
set_attribute library {tim_lib1 tim_lib2} [find /libraries -library_domain dom1]
set_attribute library {pow_lib1 ...} [find /libraries -library_domain dom2]
set_attribute power_library dom2 [find /libraries -library_domain dom1]
```

Attribute Reference for Encounter RTL Compiler

Advanced Low Power—Library Domain Attributes

Related Information

[Using Dedicated Libraries for Power Analysis in Low Power in Encounter RTL Compiler](#)

- Related attributes:
- (design) [library_domain](#) on page 1437
 - (subdesign) [library_domain](#) on page 1444
 - (instance) [library_domain](#) on page 1448

wireload_selection

`wireload_selection {default | table | none}`

Default: default

Read-write [library_domain](#) attribute. Indicates whether to use a wire-load selection table to choose default wire-load models for blocks based on their cell areas.

<code>default</code>	RTL Compiler behaves as if the attribute had never been set. The environment reverts to the default settings.
<code>none</code>	No automatic wire-load selection by area will be performed. The only wire-load models that will be used are the ones that are set with the <code>force_wireload</code> attribute on individual modules or the default wireload model specified in the library.
<code>table</code>	Indicates the wire-load selection table to use. Specify the hierarchical path to the wire-load selection table to be used. You can obtain the path using the find command.

Example

Some libraries contain multiple selection tables (for example, for different numbers of metal layers), and in such cases you can indicate which `wireload_selection` table should be used.

```
set_att wireload_selection [find /*/my_dom -wireload_selection ALUMINUM] [find /libraries/ -library_domain my_dom]
```

Related Information

- Related attributes:
- [force_wireload](#) on page 547
 - [wireload_mode](#) on page 542
 - (root) [wireload_selection](#) on page 543

Level Shifter Group Attributes

Contain information about the level shifter groups.

- To get a `level_shifter_group` attribute value, type

```
get_attribute attribute_name \
[find /libraries -level_shifter_group group]
```

Note: These attributes are located at `/libraries/level_shifter_groups/group`.

direction

```
direction {up | down | bidir}
```

Read-only `level_shifter_group` attribute. Specifies the direction in which this group of level shifters can shift the voltage.

- `up` indicates that the level shifters connect from a lower voltage domain to a higher voltage domain.
- `down` indicates that the level shifters connect from a higher voltage domain to a lower voltage domain.
- `bidir` indicates that the direction of the level shifters can be up or down.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

ground_direction

```
ground_direction {up | down | bidir}
```

Read-only `level_shifter_group` attribute. Specifies the direction in which this group of ground level shifters can shift the ground voltage.

- `up` indicates that the level shifters connect from a lower voltage domain to a higher voltage domain.
- `down` indicates that the level shifters connect from a higher voltage domain to a lower voltage domain.
- `bidir` indicates that the direction of the level shifters can be up or down.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

level_shifter_cells

`level_shifter_cells string`

Read-only [level_shifter_group](#) attribute. Returns the list of library cells in this group that can be used as level shifters between the voltages specified in the `min_input_voltage` (and `max_input_voltage`) and the `min_output_voltage` (and `max_output_voltage`) attributes.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

max_ground_input_voltage

`max_ground_input_voltage float`

Read-only [level_shifter_group](#) attribute. Returns the maximum voltage for the input (source) ground supply voltage that can be handled by this level shifter.

Note: This attribute only applies to ground level shifters.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (libcell) [max_ground_input_voltage](#) on page 1432

(level_shifter_group) [min_ground_input_voltage](#) on page 1434

(libcell) [min_ground_input_voltage](#) on page 1434

Attribute Reference for Encounter RTL Compiler

Advanced Low Power—Level Shifter Group Attributes

max_ground_output_voltage

max_ground_output_voltage *float*

Read-only level_shifter_group attribute. Returns the maximum voltage for the output (destination) ground supply voltage that can be handled by this level shifter.

Note: This attribute only applies to ground level shifters.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (libcell) [max_ground_output_voltage](#) on page 1433

(level_shifter_group) [min_ground_output_voltage](#) on page 1434

(libcell) [min_ground_output_voltage](#) on page 1434

max_input_voltage

max_input_voltage *float*

Read-only level_shifter_group attribute. Specifies the upper bound of the voltage range (in volt) that can be handled by the level shifter for the source domain.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

Related attribute: (libcell) [max_input_voltage](#) on page 1433

max_output_voltage

max_output_voltage *float*

Read-only level_shifter_group attribute. Specifies the upper bound of the voltage range (in volt) that can be handled by the level shifter for the destination domain.

Related Information

- Affected by this command: [read_power_intent](#)
Affected by this attribute: (library_domain) [library](#) on page 1427
Related attribute: (libcell) [max_output_voltage](#) on page 1433

min_ground_input_voltage

`min_ground_input_voltage float`

Read-only [level_shifter_group](#) attribute. Returns the minimum voltage for the input (source) ground supply voltage that can be handled by this level shifter.

Note: This attribute only applies to ground level shifters.

Related Information

- Affected by this command: [read_power_intent](#)
Related attributes: (level_shifter_group) [min_ground_input_voltage](#) on page 1434
(level_shifter_group) [max_ground_input_voltage](#) on page 1432
(libcell) [max_ground_input_voltage](#) on page 200

min_ground_output_voltage

`min_ground_output_voltage float`

Read-only [level_shifter_group](#) attribute. Returns the minimum voltage for the output (destination) ground supply voltage that can be handled by this level shifter.

Note: This attribute only applies to ground level shifters.

Related Information

- Affected by this command: [read_power_intent](#)
Related attributes: (level_shifter_group) [min_ground_output_voltage](#) on page 1434
(level_shifter_group) [max_ground_output_voltage](#) on page 1433
(libcell) [max_ground_output_voltage](#) on page 201

Attribute Reference for Encounter RTL Compiler

Advanced Low Power—Level Shifter Group Attributes

min_input_voltage

`min_input_voltage float`

Read-only level_shifter_group attribute. Specifies the lower bound of the voltage range (in volt) that can be handled by the level shifter for the source domain.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

Related attribute: (libcell) [min_input_voltage](#) on page 1435

min_output_voltage

`min_output_voltage float`

Read-only level_shifter_group attribute. Specifies the lower bound of the voltage range (in volt) that can be handled by the level shifter for the destination domain.

Related Information

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

Related attribute: (libcell) [min_output_voltage](#) on page 1435

valid_location

`valid_location {from|to|on|off|either|any}`

Read-only level_shifter_group attribute. Specifies where the level shifters must be placed.

Related Information

[define_level_shifter_cell](#) in the *Common Power Format Language Reference*

Affected by this command: [read_power_intent](#)

Affected by this attribute: (library_domain) [library](#) on page 1427

Libcell Attributes

Contain information about a cell in the specified technology library.

- To set a libcell attribute, type

```
set_attribute attribute_name attribute_value \
[find /lib*/library -libcell cell]
```

- To get a libcell attribute value, type

```
get_attribute attribute_name [find /lib*/library -libcell cell]
```

aux_enables

`aux_enables libpin_list`

Read-write [libcell](#) attribute. Specifies a list of additional or auxiliary enable pins for the isolation cell.

Related Information

[Modeling Isolation Cells](#) in the *Common Power Format User Guide*

[define_isolation_cell](#) in the *Common Power Format Language Reference*

Affected by this command: [read_power_intent](#)

valid_location

`valid_location {from|to|on|off|either|any}`

Read-write [libcell](#) attribute. Specifies where the level shifter cell can be placed.

Related Information

[define_level_shifter_cell](#) in the *Common Power Format Language Reference*

Affected by this command: [read_power_intent](#)

Design Attributes

Contain information about a multiple supply voltage design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design
```

or

```
set_attribute attribute_name attribute_value [find /des* -design name]
```

- To get a design attribute value, type

```
get_attribute attribute_name design
```

base_mode

`base_mode mode`

Read-write design attribute. Specifies the design mode for which you want to report the timing and power. Setting this attribute, causes the netlist to be relinked to the library domains specified for this mode.

Note: This attribute applies only for dynamic voltage frequency scaling (DVFS).

Related Information

Related attribute: [library_domain_by_mode](#) on page 1465

library_domain

`library_domain domain`

Read-write design attribute. Sets the target library domain for technology mapping. During mapping only library cells from libraries in the target domain can be used.

If you do not set this attribute explicitly on the design, the design is linked to the default library domain.

Note: The order in which you set the `library_domain` attributes on the design and subdesigns matters! You must set the library domain of the design first. The design attribute applies hierarchically to all instances and subdesigns of this design.



You cannot set this attribute, if the design is marked preserved.

Related Information

Related attributes:

- [default](#) on page 1427
- (hdl_arch) [library_domain](#) on page 1443
- (instance) [library_domain](#) on page 1448
- (power_domain) [library_domain](#) on page 1465
- (subdesign) [library_domain](#) on page 1444

lp_isolate_domain_crossing_constants

`lp_isolate_domain_crossing_constants {true | false}`

Default: true

Read-write [design](#) attribute. Controls level-shifter and isolation cell insertion on constant nets in the CPF flow. By default, insertion occurs on constant nets.

Note: This attribute applies for CPF version 1.0e and above.

Related Information

Affects this command: [commit_power_intent](#)

pi_relax_map_iso_cell_checks

`pi_relax_map_iso_cell_checks {false | true}`

Default: false

Read-write [design](#) attribute. Controls whether to relax sanity checks on isolation cells listed with the `map_isolation_cell` command specified in the power intent file using the format described in the IEEE 1801-2009 standard.

If this attribute is enabled, the tool will skip checks on the isolation type during isolation cell insertion.

However, the tool will still check if the library domain of the cell matches with the library domain of the insertion location.

If the isolation cells specified with the `map_isolation_cell` command are not referenced in the isolation rule, the attribute setting has no effect.

Related Information

[IEEE-1801 Support in RTL Compiler in Low Power in Encounter RTL Compiler](#)

Affects this command: [commit_power_intent](#)

pi_relax_map_ls_cell_checks

`pi_relax_map_ls_cell_checks {false | true}`

Default: false

Read-write [design](#) attribute. Controls whether to relax sanity checks on level shifter cells listed with the `map_level_shifter_cell` command specified in the power intent file using the format described in the IEEE 1801-2009 standard.

If this attribute is enabled, the tool will skip checks during level shifter insertion on

- the input and output range of the cell
- the direction of the level shifter cell

However, the tool will still check if the library domain of the cell matches with the library domain of the insertion location.

If the level shifters specified with the `map_level_shifter_cell` command are not referenced in the level shifter rule, the attribute setting has no effect.

Related Information

[IEEE-1801 Support in RTL Compiler in Low Power in Encounter RTL Compiler](#)

Affects this command: [commit_power_intent](#)

power_domain

`power_domain domain`

Read-only [design](#) attribute. Returns the default power domain for the design.

Related Information

Affected by this command: [read_power_intent](#)

Constant Attributes

Contain information about the specified constant. Each level of hierarchy has its own dedicated logic constants that can only be connected to other objects within that level of hierarchy, such as `logic0` and `logic1` pins. They are in the `constants` directory and are called 1 and 0.

- To get a constant attribute value, type

```
get_attribute attribute_name [find /des*/design_name/instances_hier/  
instance_name/constants -constant 0]
```

or

```
get_attribute attribute_name [find /des*/design_name/instances_hier/  
instance_name/constants -constant 1]
```

power_domain

`power_domain domain`

Read-write constant attribute. Specifies the power domain to which this constant belongs.

Related Information

Related attributes:	(instance) power_domain on page 1448
	(pin) power_domain on page 1451
	(port) power_domain on page 1442
	(subport) power_domain on page 1447

Port Attributes

Contain information about single bit connection points in the specified top-level design. Port objects are stored in `ports_in` and `ports_out` directories.

- To set a port attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -port name]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -port name]
```

isolation_rule

`isolation_rule rules`

Read-only `port` attribute. Returns the isolation rules specified in the CPF file that apply to this port.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (pin) [isolation_rule](#) on page 1450

(subport) [isolation_rule](#) on page 1446

level_shifter_rule

`level_shifter_rule rules`

Read-only `port` attribute. Returns the level shifter rules specified in the CPF file that apply to this port.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (pin) [level_shifter_rule](#) on page 1450

(subport) [level_shifter_rule](#) on page 1447

power_domain

`power_domain domain`

Read-write `port` attribute. Specifies the power domain to which this port belongs.

Related Information

Related attributes:	(constant) power_domain on page 1440
	(instance) power_domain on page 1448
	(pin) power_domain on page 1451
	(subport) power_domain on page 1447

hdl_arch Attributes

Contain information about user-defined modules (or entities).

- To set an hdl_arch attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -hdl_arch name]
```

- To get a an hdl_arch attribute value, type

```
get_attribute attribute_name \
[find /des*/design -hdl_arch name]
```

library_domain

```
library_domain domain
```

Read-write hdl_arch attribute. Sets the target library domain for technology mapping of the specified architecture. During mapping only library cells from libraries in the target domain can be used.

Note: This attribute is not hierarchical. It only applies to the specified architecture.

Related Information

Related attributes:

(design) [library_domain](#) on page 1437

(instance) [library_domain](#) on page 1448

(power_domain) [library_domain](#) on page 1465

(subdesign) [library_domain](#) on page 1444

Subdesign Attributes

Contain information about the subdesigns in the specified design. Subdesigns correspond to Verilog modules or VHDL entities instantiated in the top-level Verilog module or top-level VHDL entity.

- To set a subdesign attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subdesign name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subdesign name]
```

Note: These attributes are located at `/designs/design/subdesigns/subdesign`.

library_domain

```
library_domain domain
```

Read-write `subdesign` attribute. Sets the target library domain for technology mapping of the specified subdesign. During mapping only library cells from libraries in the target domain can be used.

By default, a subdesign inherits the library domain setting from its parent subdesign or design.

Note: The order in which you set the `library_domain` attributes on the design and subdesigns matters! Set the library domain on the design before you set it on a subdesign. The library domain setting for a subdesign applies to all instantiations of that subdesign in the design. The subdesign attribute also applies hierarchically to all instances and subdesigns of this subdesign.



You cannot set this attribute, if the subdesign is marked preserved.

Examples

The following command sets the library domain to `dom_1` on the subdesign corresponding to the specified hierarchical instance `g2`.

```
set_attribute library_domain [find / -library_domains/dom_1] \
[get_attribute subdesign [find / -instance inst*hier/g2]]
```

Attribute Reference for Encounter RTL Compiler

Advanced Low Power—Subdesign Attributes

Related Information

Related attributes:

- (design) [library_domain](#) on page 1437
- (hdl_arch) [library_domain](#) on page 1443
- (instance) [library_domain](#) on page 1448
- (power_domain) [library_domain](#) on page 1465

Subport Attributes

Contain information about subports in the specified design. A subport is a single bit connection point within a hierarchical instance, that is a module or entity that has been instantiated.

- To set a subport attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subport hier_instance/name]
```

- To get a subport attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subport hier_instance/name]
```

Note: You may need to specify more than one (hierarchical) instance to serve as the search path that will uniquely identify the relevant subport.

isolation_rule

isolation_rule rules

Read-only subport attribute. Returns the isolation rules specified in the CPF file that apply to this subport.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (pin) [isolation_rule](#) on page 1450

 (port) [isolation_rule](#) on page 1441

level_shifter_rule

`level_shifter_rule rules`

Read-only subport attribute. Returns the level shifter rules specified in the CPF file that apply to this subport.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (pin) [level_shifter_rule](#) on page 1450
(port) [level_shifter_rule](#) on page 1441

power_domain

`power_domain domain`

Read-write subport attribute. Specifies the power domain to which this subport belongs.

Related Information

Related attributes: (constant) [power_domain](#) on page 1440
(instance) [power_domain](#) on page 1448
(pin) [power_domain](#) on page 1451
(port) [power_domain](#) on page 1442

Instance Attributes

Contain information about a combinational, sequential, or hierarchical instance in the specified design. Instance objects are found in the `instances_comb`, `instances_seq`, and `instances_hier` directories.

- To set an instance attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -instance name]
```

- To get an instance attribute value, type

```
get_attribute attribute_name [find /des*/design -instance name]
```

library_domain

`library_domain domain`

Read-write `instance` attribute. Identifies the library domain to which this instance belongs. You can only set this attribute on a timing model instance.

Related Information

Related attributes:

(design) [library_domain](#) on page 1437

(hdl_arch) [library_domain](#) on page 1443

(power_domain) [library_domain](#) on page 1465

(subdesign) [library_domain](#) on page 1444

power_domain

`power_domain domain`

Read-write `instance` attribute. Identifies the primary power domain to which this instance belongs. You can only set this attribute on a hierarchical instance or a timing model instance.

Related Information

Related attributes:

(constant) [power_domain](#) on page 1440

(pin) [power_domain](#) on page 1451

(port) [power_domain](#) on page 1442

(subport) [power_domain](#) on page 1447

secondary_domain

`secondary_domain power_domain`

Read-write instance attribute. Returns the secondary domain of the instance of a special low power cell (level shifter cell, isolation cell, state retention cell, always-on cell, or power switch cell).

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (isolation_rule) [secondary_domain](#) on page 1456

(state_retention_rule) [secondary_domain](#) on page 1470

state_retention_rule

`state_retention_rule state_retention_rule`

Read-write instance attribute. Returns the state retention rule that applies to this instance.

Related Information

Affected by this command: [read_power_intent](#)

Pin Attributes

Contain information about the specified instance pin. A pin is a single bit connection point on an instance. Pin objects are stored in `pins_in` and `pins_out` directories.

- To set a pin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pin instance/pin]
```

- To get a pin attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pin instance/pin]
```

Note: You may need to specify more than one instance to serve as the search path that will uniquely identify the relevant pin.

isolation_rule

`isolation_rule rules`

Read-only `pin` attribute. Returns the isolation rules specified in the CPF file that apply to this pin.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (port) [isolation_rule](#) on page 1441

(subport) [isolation_rule](#) on page 1446

level_shifter_rule

`level_shifter_rule rules`

Read-only `pin` attribute. Returns the level shifter rules specified in the CPF file that apply to this pin.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (port) [level_shifter_rule](#) on page 1441

(subport) [level_shifter_rule](#) on page 1447

power_domain

`power_domain domain`

Read-write `pin` attribute. Identifies the power domain to which this pin belongs. You can only change the power domain of the pin of a timing-model instance or of an unresolved reference instance.

Related Information

Related attributes:	(constant) power_domain on page 1440
	(instance) power_domain on page 1448
	(port) power_domain on page 1442
	(subport) power_domain on page 1447

Isolation Rule Attributes

Contain information about the isolation rules associated with a design. These attributes are read-write attributes.

- To set an `isolation_rule` attribute value, type

```
set_attribute attribute_name attribute_value \
[find /*/isolation_rules -isolation_rule rule]
```

- To get an `isolation_rule` attribute value, type

```
get_attribute attribute_name \
[find /*/isolation_rules -isolation_rule rule]
```

Note: These attributes are located at `/designs/design/power/isolation_rules/rule`.

cells

```
cells cell_list
```

Read-write `isolation_rule` attribute. Returns the list of cells that can be used to insert isolation logic according to this isolation rule.

Related Information

Affected by this command: [read_power_intent](#)

cpf_pins

```
cpf_pins pin_list
```

Read-write `isolation_rule` attribute. Returns the list of pins that was specified with the `-pins` option of the `create_isolation_rule` CPF command.

Related Information

Affected by this command: [read_power_intent](#)

enable_driver

```
enable_driver {pin |port}
```

Read-write isolation_rule attribute. Specifies the driver of the enable signal which controls when the isolation cells are in isolation mode.

Related Information

Affected by this command: [read_power_intent](#)

enable_polarity

```
enable_polarity {active_high | active_low}
```

Default: active_high

Read-write isolation_rule attribute. Specifies the polarity of the enable signal which controls when the isolation cells are in isolation mode.

Related Information

Affected by this command: [read_power_intent](#)

exclude_pins

```
exclude_pins pin_list
```

Read-write isolation_rule attribute. Returns the list of pins that was specified with the -exclude option of the `create_isolation_rule` CPF command.

Related Information

Affected by this command: [read_power_intent](#)

from_power_domain

`from_power_domain domain_list`

Read-write isolation_rule attribute. Returns the list of power domains specified with the -from option of the `create_isolation_rule` CPF command.

Related Information

Affected by this command: [read_power_intent](#)

location

`location {from | to}`

Read-only isolation_rule attribute. Specifies the location of the isolation cells.

Note: To change the location, you need to remove the isolation rule.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (`level_shifter_rule`) [location](#) on page 1458

off_domain

`off_domain {from | to}`

Default: from

Read-write isolation_rule attribute. Specifies which power domain is powered down.

Related Information

Affected by this command: [read_power_intent](#)

Attribute Reference for Encounter RTL Compiler

Advanced Low Power—Isolation Rule Attributes

output_value

`output_value {low | high | hold | complex | any}`

Default: low

Read-write `isolation_rule` attribute. Specifies the required output state of the isolation cells.

Related Information

Affected by this command: [read_power_intent](#)

pins

`pins pin_list`

Read-only `isolation_rule` attribute. Returns the list of pins to which this isolation rule applies.

Related Information

Affected by this command: [read_power_intent](#)

prefix

`prefix string`

Default: RC_ISO

Read-write `isolation_rule` attribute. Specifies the prefix used to name the isolation modules and hierarchical instances inserted according to the isolation rule.

Related Information

Affected by this command: [read_power_intent](#)

secondary_domain

`secondary_domain power_domain`

Read-write `isolation_rule` attribute. Returns the secondary domain of the instances of the isolation rule. The secondary domain is the domain whose primary power and ground nets must be connected to the secondary power and ground pins of an isolation cell.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (instance) [secondary_domain](#) on page 1449

(state_retention_rule) [secondary_domain](#) on page 1470

to_power_domain

`to_power_domain domain_list`

Read-write `isolation_rule` attribute. Returns the list of power domains specified with the `-to` option of the `create_isolation_rule` CPF command.

Related Information

Affected by this command: [read_power_intent](#)

within_hierarchy

`within_hierarchy instance`

Read-write `isolation_rule` attribute. Specifies the instance in which the isolation logic (with or without wrapper) must be inserted.

The power domain of the specified instance must match the power domain of the selected location.

Related Information

Affected by this command: [read_power_intent](#)

Level Shifter Rule Attributes

Contain information about the level shifter rules created in the CPF file that was read in. These attributes are read-only attributes.

- To get a `level_shifter_rule` attribute value, type

```
get_attribute attribute_name \
[find /*/level_shifter_rules -level_shifter_rule rule]
```

Note: These attributes are located at
`/designs/design/power/level_shifter_rules/rule`.

cells

```
cells cell_list
```

Read-write `level_shifter_rule` attribute. Returns the list of cells that can be used to insert level shifters according to this level-shifter rule.

Related Information

Affected by this command: [read_power_intent](#)

cpf_pins

```
cpf_pins pin_list
```

Read-write `level_shifter_rule` attribute. Returns the list of pins that was specified with the `-pins` option of the `create_level_shifter_rule` CPF command.

Related Information

Affected by this command: [read_power_intent](#)

direction

```
direction {up|down|both}
```

Read-only `level_shifter_rule` attribute. Specifies the direction of the level shifters for a global level shifter rule.

Related Information

Affected by this command: [read_power_intent](#)

exclude_pins

`exclude_pins pin_list`

Read-write [level_shifter_rule](#) attribute. Returns the list of pins that was specified with the `-exclude` option of the `create_level_shifter_rule` CPF command.

Related Information

Affected by this command: [read_power_intent](#)

from_power_domain

`from_power_domain domain_list`

Read-write [level_shifter_rule](#) attribute. Returns the list of power domains specified with the `-from` option of the `create_level_shifter_rule` CPF command.

Related Information

Affected by this command: [read_power_intent](#)

location

`location {to | from}`

Default: to

Read-only [level_shifter_rule](#) attribute. Returns the location that was specified with the `-location` option of the `update_level_shifter_rules` CPF command.

Related Information

Affected by this command: [read_power_intent](#)

Related attribute: (isolation_rule) [location](#) on page 1454

pins

pins pin_list

Read-only level_shifter_rule attribute. Returns the list of pins to which the rule defined with the `create_level_shifter_rule` CPF command is applicable.

Related Information

Affected by this command: [read_power_intent](#)

Related attribute: [\(isolation_rule\) pins](#) on page 1455

prefix

prefix string

Read-only level_shifter_rule attribute. Returns the prefix that was specified with the `-prefix` option of the `update_level_shifter_rules` CPF command.

Related Information

Affected by this command: [read_power_intent](#)

Related attribute: [\(isolation_rule\) prefix](#) on page 1455

to_power_domain

to_power_domain domain_list

Read-write level_shifter_rule attribute. Returns the list of power domains specified with the `-to` option of the `create_level_shifter_rule` CPF command.

Related Information

Affected by this command: [read_power_intent](#)

within_hierarchy

`within_hierarchy instance`

Read-write level_shifter_rule attribute. Specifies the instance in which the level shifter (with or without wrapper) must be inserted.

The power domain of the specified instance must match the power domain of the selected location.

Related Information

Affected by this command: [read_power_intent](#)

Nominal Condition Attributes

Contain information about the nominal conditions specified in the CPF file that was read in. These attributes are read-only attributes.

- To get a nominal_condition attribute value, type

```
get_attribute attribute_name \
[find /*/nominal_conditions -nominal_condition nc]
```

Note: These attributes are located at
`/designs/design/power/nominal_conditions/nc`

ground_voltage

`ground_voltage voltage_list`

Read-write `nominal_condition` attribute. Returns the ground supply voltage(s) for this nominal condition. This value corresponds to the value specified for the `-ground_voltage` option of the `create_nominal_condition` CPF command specified for this nominal condition. The list can contain up to three voltages: minimum, nominal, and maximum voltages. The list must contain increasing values.

Example

The following command checks the ground voltage for the `med` nominal condition.

```
rc:/designs/top> get_attr ground_voltage \
[find /*/nominal_conditions -nominal_condition med]
0.0
```

Related Information

[create_nominal_condition](#) in the *Common Power Format Language Reference*

Affected by this command: [read_power_intent](#)

library_set

`library_set library_domain`

Read-write nominal_condition attribute. Returns the library set associated with the specified condition. The value corresponds to the value specified for the `-library_set` option of the `update_nominal_condition` CPF command specified for this nominal condition.

Note: In RTL Compiler, the library sets correspond to library domains.

Example

The following command returns the library set used for the `med` nominal condition.

```
rc:/designs/top> get_attr library_set \
[find /*/nominal_conditions -nominal_condition med]
/libraries/library_domains/umc_08v
```

Related Information

Affected by this command: [read_power_intent](#)

voltage

`voltage voltage_list`

Read-write nominal_condition attribute. Returns the supply voltage(s) for this nominal condition. This value corresponds to the value specified for `-voltage` option of the `create_nominal_condition` CPF command specified for this nominal condition. The list can contain up to three voltages: minimum, nominal, and maximum voltages. The list must contain increasing values.

Note: You cannot assign any other voltage to a nominal condition with voltage 0.

Related Information

[create_nominal_condition](#) in the *Common Power Format Language Reference*

Affected by this command: [read_power_intent](#)

Power_Domain Attributes

Contain information about the power domains associated with a design. These attributes are read-write attributes.

- To set an `power_domain` attribute value, type

```
set_attribute attribute_name attribute_value \
[find /designs/design/power -power_domain domain]
```

- To get an `power_domain` attribute value, type

```
get_attribute attribute_name \
[find /designs/design/power -power_domain domain]
```

Note: These attributes are located at `/designs/design/power/power_domains/domain`.

base_domains

```
base_domains domain_list
```

Read-only `power_domain` attribute. Returns the list of base domains associated with this power domain. These power (base) domains supply external power to the primary domain through some power switch network.

Related Information

Set by this command: [read_power_intent](#)

default

```
default {false | true}
```

Default: false

Read-write `power_domain` attribute. Indicates whether the power domain is the default power domain. By default, the first created power domain becomes the default power domain. Set this attribute to `true` for the desired domain.

Note: Only one power domain can be the default domain.

Example

```
rc:/> create_power_domain -design top -name pd1
/designs/top/power/power_domains/pd1
```

Attribute Reference for Encounter RTL Compiler

Advanced Low Power—Power_Domain Attributes

```
rc:/> create_power_domain -design top -name pd2  
/designs/top/power/power_domains/pd2  
rc:/> get_att default [find /*/power -power_domain pd1]  
true
```

Related Information

Affected by this command:

[read_power_intent](#)

Related attributes:

(library_domain) [default](#) on page 1427

(mode) [default](#) on page 561

(hdl_arch) [library_domain](#) on page 1443

(instance) [library_domain](#) on page 1448

(subdesign) [library_domain](#) on page 1444

dft_iso_rule

```
dft_iso_rule iso_rule
```

Read-write [power_domain](#) attribute. Specifies the isolation rule to be associated with any pins or ports created by DFT in this power domain.

Related Information

[PSO with DFT Flow in Design for Test in Encounter RTL Compiler](#)

has_external_shutoff

```
has_external_shutoff {false | true}
```

Default: false

Read-write [power_domain](#) attribute. Specifies whether the power domain can be powered down by an external power switch (outside of the chip).

Related Information

[PSO with DFT Flow in Design for Test in Encounter RTL Compiler](#)

Affected by this command:

[read_power_intent](#)

library_domain

```
library_domain domain
```

Read-write power_domain attribute. Specifies the library domain to be used to optimize or analyze this power domain.

Note: In a CPF flow, this attribute is automatically set for the *default* power mode.

Related Information

Affected by this command: [read_power_intent](#)

Related attributes: (design) [library_domain](#) on page 1437

(hdl_arch) [library_domain](#) on page 1443

(instance) [library_domain](#) on page 1448

(subdesign) [library_domain](#) on page 1444

library_domain_by_mode

```
library_domain_by_mode {{mode library_domain}...}
```

Read-write power_domain attribute. Specifies for each mode the library domain to be associated with this power domain. Specify a Tcl list of Tcl lists. There can be as many Tcl lists as there are modes. Every Tcl list contains the mode name followed by the library domain name.

Note: In the native flow, this attribute applies only for dynamic voltage frequency scaling (DVFS). In the CPF flow, this attribute is automatically set and contains the library domain information for all modes except the default power mode.

Example

```
set_attribute library_domain_by_mode {{active_mode 1.08v} {sleep_mode 0.75v}} pd1
```

Related Information

Related attribute: [base_mode](#) on page 1437

shutoff_condition

`shutoff_condition string`

Read-only `power_domain` attribute. Returns the condition when a power domain is shut off

Power_Mode Attributes

Contain information about the power modes for the design in the CPF file. These attributes are read-write attributes.

- To set a power_mode attribute value, type

```
set_attribute attribute_name attribute_value \
[find /*/power -power_mode power_mode]
```

- To get an power_mode attribute value, type

```
get_attribute attribute_name \
[find /*/power -power_mode power_mode]
```

Note: These attributes are located at

```
/designs/design/power/power_modes/power_mode
```

constraint_mode

```
constraint_mode mode
```

Read-write power_mode attribute. Specifies the timing constraint mode for this power mode.

default

```
default {false | true}
```

Default: false

Read-write power_mode attribute. Indicates whether the power mode was identified as the default power mode in the CPF file.

domain_conditions

```
domain_conditions domain_conditions
```

Read-write power_mode attribute. Specifies the domain conditions of the power mode. The value contains a Tcl list for each power domain in the power mode. Each list contains the path to the power domain name and to the nominal condition for the power domain in this power mode. The information corresponds to the value for the -domain_conditions option of the create_power_mode command for this power mode.

State Retention Rule Attributes

Contain information about the state retention rules for the design in the CPF file. Most attributes are read-write attributes.

- To set a `state_retention_rule` attribute value, type

```
set_attribute attribute_name attribute_value \
[find /*/power -state_retention_rule state_retention_rule]
```

- To get an `state_retention_rule` attribute value, type

```
get_attribute attribute_name \
[find /*/power -state_retention_rule state_retention_rule]
```

Note: These attributes are located at

```
/designs/design/power/state_retention_rules/state_retention_rule
```

cell_type

`cell_type string`

Read-write `state_retention_rule` attribute. Specifies the type of library cells that can be used to map the sequential cells of this rule.

Note: The specified cell type must correspond to a cell type specified in a `define_state_retention_cell` command in the CPF file.

Related Information

Affected by this command: [read_power_intent](#)

cells

`cells cell_list`

Read-write `state_retention_rule` attribute. Specifies a list of library cells that can be used to map the sequential cells of this rule.

Related Information

Affected by this command: [read_power_intent](#)

restore

`restore {pin|port|bus}`

Read-write state retention rule attribute. Specifies the restore signal for the state retention registers of this rule.

Related Information

Affected by this command: [read_power_intent](#)

restore_phase

`restore_phase string`

Read-only state retention rule attribute. Returns the phase of the restore signal for the state retention registers of this rule.

Related Information

Affected by this command: [read_power_intent](#)

save

`save {pin|port|bus}`

Read-write state retention rule attribute. Specifies the save signal for the state retention registers of this rule.

Related Information

Affected by this command: [read_power_intent](#)

save_phase

`save_phase string`

Read-only state retention rule attribute. Returns the phase of the save signal for the state retention registers of this rule.

Related Information

Affected by this command: [read_power_intent](#)

secondary_domain

`secondary_domain power_domain`

Read-write [state_retention_rule](#) attribute. Specifies the name of the power domain that provides the continuous power when the state retention registers of this rule are in retention mode.

.Related Information

Affected by this command: [read_power_intent](#)

Customization

Root Attributes

- [ui_respects_preserve](#) on page 1473

Design Attributes

- [user_defined](#) on page 1474

Instance Attributes

- [user_defined](#) on page 1475

Pin Attributes

- [user_defined](#) on page 1476

Pgpin Attributes

- [user_defined](#) on page 1477

Net Attributes

- [user_defined](#) on page 1478

Port Attributes

- [user_defined](#) on page 1479

Subdesign Attributes

- [user_defined](#) on page 1480

Attribute Reference for Encounter RTL Compiler

Customization—List

Subport Attributes

- [user_defined](#) on page 1481

Message Group Attributes

- [is_user](#) on page 1482

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

ui_respects_preserve

```
ui_respects_preserve {true | false}
```

Default: true

Read-write `root` attribute. Specifies whether netlist editing commands should fail if they attempt to modify preserved logic. Set this attribute to `false` to allow netlist editing commands to modify preserved instances.

Related Information

Affects these commands: [edit_netlist](#)
[rm](#)

Design Attributes

Contain information about the specified design.

- To set a design attribute, type

```
set_attribute attribute_name attribute_value /designs/design
```

or

```
set_attribute attribute_name attribute_value [find /des* -design name]
```

- To get a design attribute value, type

```
get_attribute attribute_name /designs/design
```

user_defined

user_defined string

Read-write design attribute. Provided to make Tcl scripting easier. The specified string can contain any provided value for the attribute.

Related Information

Related attributes:	(instance) user_defined on page 1475
	(libcell) user_defined on page 210
	(libpin) user_defined on page 227
	(pin) user_defined on page 1476
	(pgpin) user_defined on page 1477
	(net) user_defined on page 1478
	(port) user_defined on page 1479
	(subdesign) user_defined on page 1480
	(subport) user_defined on page 1481

Instance Attributes

Contain information about a combinational, sequential, or hierarchical instance in the specified design. Instance objects are found in the `instances_comb`, `instances_seq`, and `instances_hier` directories.

- To set an instance attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -instance name]
```

- To get an instance attribute value, type

```
get_attribute attribute_name [find /des*/design -instance name]
```

user_defined

`user_defined string`

Read-write `instance` attribute. Provided to make Tcl scripting easier. The specified string can contain any provided value for the attribute.

Related Information

Related attributes:	(design) user_defined on page 1474
	(libcell) user_defined on page 210
	(libpin) user_defined on page 227
	(pin) user_defined on page 1476
	(pgpin) user_defined on page 1477
	(net) user_defined on page 1478
	(port) user_defined on page 1479
	(subdesign) user_defined on page 1480
	(subport) user_defined on page 1481

Pin Attributes

Contain information about the specified instance pin. A pin is a single bit connection point on an instance. Pin objects are stored in `pins_in` and `pins_out` directories.

- To set a pin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pin instance/pin]
```

- To get a pin attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pin instance/pin]
```

Note: You may need to specify more than one instance to serve as the search path that will uniquely identify the relevant pin.

user_defined

`user_defined string`

Read-write `pin` attribute. Provided to make Tcl scripting easier. The specified string can contain any provided value for the attribute.

Related Information

Related attributes:	(design) user_defined on page 1474
	(instance) user_defined on page 1475
	(libcell) user_defined on page 210
	(libpin) user_defined on page 227
	(pgpin) user_defined on page 1477
	(net) user_defined on page 1478
	(port) user_defined on page 1479
	(subdesign) user_defined on page 1480
	(subport) user_defined on page 1481

Pgpin Attributes

Contain information about power and ground instance pins. Pgpin objects are stored in `pgpins_in` and `pgpins_out` directories.

- To set a pgpin attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -pgpin instance/pin]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -pgpin instance/pin]
```

user_defined

`user_defined string`

Read-write `pgpin` attribute. Provided to make Tcl scripting easier. The specified string can contain any provided value for the attribute.

Related Information

Related attributes:	(design) user_defined on page 1474
	(instance) user_defined on page 1475
	(libcell) user_defined on page 210
	(libpin) user_defined on page 227
	(pin) user_defined on page 1476
	(pgpin) user_defined on page 1477
	(net) user_defined on page 1478
	(port) user_defined on page 1479
	(subdesign) user_defined on page 1480
	(subport) user_defined on page 1481

Net Attributes

Contain information about a net in the specified design. A net can be a top-level net or can belong to a hierarchical instance.

- To set a net attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -net instance/net]
```

- To get a net attribute value, type

```
get_attribute attribute_name [find /des*/design -net instance/net]
```

Note: You may need to specify several instances to uniquely identify the net.

user_defined

user_defined string

Read-write net attribute. Provided to make Tcl scripting easier. The specified string can contain any provided value for the attribute.

Related Information

Related attributes:	(design) user_defined on page 1474
Related attributes:	(instance) user_defined on page 1475
	(libcell) user_defined on page 210
	(libpin) user_defined on page 227
	(pin) user_defined on page 1476
	(pgpin) user_defined on page 1477
	(port) user_defined on page 1479
	(subdesign) user_defined on page 1480
	(subport) user_defined on page 1481

Port Attributes

Contain information about single bit connection points in the specified top-level design. Port objects are stored in `ports_in` and `ports_out` directories.

- To set a port attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -port name]
```

- To get a port attribute value, type

```
get_attribute attribute_name \
[find /des*/design -port name]
```

user_defined

`user_defined string`

Read-write `port` attribute. Provided to make Tcl scripting easier. The specified string can contain any provided value for the attribute.

Related Information

Related attributes:	(design) user_defined on page 1474
Related attributes:	(instance) user_defined on page 1475
	(libcell) user_defined on page 210
	(libpin) user_defined on page 227
	(pin) user_defined on page 1476
	(pgpin) user_defined on page 1477
	(net) user_defined on page 1478
	(subdesign) user_defined on page 1480
	(subport) user_defined on page 1481

Subdesign Attributes

Contain information about the subdesigns in the specified design. Subdesigns correspond to Verilog modules or VHDL entities instantiated in the top-level Verilog module or top-level VHDL entity.

- To set a subdesign attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subdesign name]
```

- To get a subdesign attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subdesign name]
```

user_defined

user_defined string

Read-write subdesign attribute. Provided to make Tcl scripting easier. The specified string can contain any provided value for the attribute.

Related Information

Related attributes: (design) [user_defined](#) on page 1474

Related attributes: (instance) [user_defined](#) on page 1475

(libcell) [user_defined](#) on page 210

(libpin) [user_defined](#) on page 227

(pin) [user_defined](#) on page 1476

(pgpin) [user_defined](#) on page 1477

(net) [user_defined](#) on page 1478

(port) [user_defined](#) on page 1479

(subport) [user_defined](#) on page 1481

Subport Attributes

Contain information about subports in the specified design. A subport is a single bit connection point within a hierarchical instance, that is a module or entity that has been instantiated.

- To set a subport attribute, type

```
set_attribute attribute_name attribute_value \
[find /des*/design -subport hier_instance/name]
```

- To get a subport attribute value, type

```
get_attribute attribute_name \
[find /des*/design -subport hier_instance/name]
```

Note: You may need to specify more than one (hierarchical) instance to serve as the search path that will uniquely identify the relevant subport.

user_defined

`user_defined string`

Read-write subport attribute. Provided to make Tcl scripting easier. The specified string can contain any provided value for the attribute.

Related Information

Related attributes:	(design) user_defined on page 1474
Related attributes:	(instance) user_defined on page 1475
	(libcell) user_defined on page 210
	(libpin) user_defined on page 227
	(pin) user_defined on page 1476
	(pgpin) user_defined on page 1477
	(net) user_defined on page 1478
	(port) user_defined on page 1479
	(subdesign) user_defined on page 1480

Message Group Attributes

Indicate if a group of messages is created by the user.

is_user

```
is_user {true | false}
```

Read-only message_group attribute. Indicates if a message is user-defined or is a member of an internal group (created by RTL Compiler). Messages created by the user have the value true while internal group messages have the value false.

Example

To get the `is_user` attribute value for the DFT message group, type

```
rc:/> get_attribute is_user /messages/DFT  
false
```

A

Applets

Root Attributes

- [applet_mode](#) on page 1484
- [applet_replay](#) on page 1485
- [applet_search_path](#) on page 1485
- [applet_server](#) on page 1486
- [applet_server_pass](#) on page 1487
- [applet_server_user](#) on page 1487

Root Attributes

Contain information about all loaded designs. The `root` object is identified by a forward slash (/).

- To set a `root` attribute, type

```
set_attribute attribute_name attribute_value /
```

- To get a `root` attribute value, type

```
get_attribute attribute_name /
```

applet_mode

```
applet_mode {local | auto | remote}
```

Default: local

Read-write `root` attribute. Specifies how the applet server must be accessed.

- In `remote` mode, the applet server is accessed over the network and could be in a remote location. In this case, you need a working network connection, and you must set the `applet_server_user` and `applet_server_pass` attributes appropriately
- In `local` mode, the applet server is on the same file system that the application is running.
- In `auto` mode, the tool defaults to the local mode unless an applet server is available over the network.

Example

The following command enables remote server mode:

```
rc:/> set_attr applet_mode remote /
Setting attribute of root '/': 'applet_mode' = remote
```

Related Information

Affects these commands:

[applet avail](#)

[applet install](#)

[applet update](#)

[applet whatis](#)

Related attributes:

[applet_server_pass](#) on page 1487

[applet_server_user](#) on page 1487

applet_replay

`applet_replay string`

Read-write root attribute. Records the applets that were successfully loaded for restoration purposes.

applet_search_path

`applet_search_path path_list`

Default: `user_home/.localApps/rc/major_version/install_path/tools.platform/lib/applets`

Read-write root attribute. Specifies one or more directories in the file system from where applets will be loaded and installation information be reported.

By default, applet commands check for a valid applet directory to load applets from in the following order:

1. `user_home/.localApps/rc/`
2. `installation_path/tools.platform/lib/applets`

If no valid applet directories are found, no applets can be loaded. Even though no applets will be available for loading, server information will continue to be available.

Example

The following example reports the current paths and then sets it to a user-specified value.

```
rc:/> get_attr applet_search_path /  
/home/joe/.localApps/rc/12.2/cdns/rc121/tools.lnx86/lib/applets  
rc:/> set_attr applet_search_path /my/local/applets /  
Setting attribute of root '/': 'applet_search_path' = /my/local/applets
```

applet_server

```
applet_server {auto |server_path}
```

Default: auto

Read-write root attribute. Specifies the location of the applet server whether on the network or on the file system.

- `auto` causes RTL Compiler to look for an applet server in the default location based on the value of the `applet_mode` attribute. If `applet_mode` is set to `auto` and no server is detected over the network, the system defaults to `local` or over the file system and looks for the applet server in the RTL Compiler installation.
- You can specify an explicit server path.
 - For a local server, the syntax is the directory path of the server root.
 - For a remote server, the syntax is `server_name/directory`

Examples

- The following example shows how to set an alternate location available on the file system.

```
rc:/> set_attr applet_mode local /  
      Setting attribute of root '': 'applet_mode' = local  
rc:/> set_attr applet_server /my/local/applets /  
      Setting attribute of root '': 'applet_server' = my/local/applets
```
- The following example shows how to set an alternate location available over the network.

```
rc:/> set_attr applet_mode remote /  
      Setting attribute of root '': 'applet_mode' = remote  
rc:/> set_attr applet_server cadence.com/apps/rc /  
      Setting attribute of root '': 'applet_server' = cadence.com/apps/rc
```

applet_server_pass

`applet_server_pass user_password`

Read-write root attribute. User password when applet server is in remote mode.

In remote mode you connect to the server through the network using FTP, which requires a user name and password. This attribute stores the password.

Example

```
rc:/> set_attr applet_server_pass VerySafe /  
Setting attribute of root '/': 'applet_server_pass' = VerySafe
```

applet_server_user

`applet_server_user user_name`

Default: anonymous

Read-write root attribute. User name when applet server is in remote mode.

In remote mode you connect to the server through the network using FTP, which requires a user name and password. This attribute stores the user name.

Examples

```
rc:/> set_attr applet_mode JohnDoe /  
Setting attribute of root '/': 'applet_server_user' = JohnDoe
```

Attribute Reference for Encounter RTL Compiler

Applets—Root Attributes

Index

Note: See [Alphabetical List of Attributes](#) for an index of the attribute names.

A

abstract scan segment
 active clock edge
 head [1299](#)
 tail [1274](#), [1306](#), [1322](#)
clock object
 head [1266](#), [1299](#)
 tail [1265](#), [1298](#)
clock object, active edge
 head [1299](#)
 tail [1266](#), [1267](#), [1298](#)
clock port
 head [1263](#), [1292](#)
 tail [1274](#), [1305](#)
clock port, active edge
 head [1263](#), [1293](#)
DFT violation type [1300](#)
shift enable, active value [1262](#), [1279](#),
 [1292](#)
shift-enable port [1273](#), [1305](#)
activity profiling
 setting scope [1377](#)
adder
 library cell defined as [188](#)
area
 cell area of design, reporting [932](#)
 cell area of instance, reporting [946](#)
 cell area of subdesign, reporting [1031](#)
 library cell [188](#)
 net area of design, reporting [940](#), [1037](#)
 total area of design, reporting [931](#),
 [1030](#)
arrival time
 uncertainty of clock capturing edge
 early-mode timing analysis
 clock [578](#), [579](#), [619](#), [620](#), [668](#)
 pin [598](#), [648](#)
 late-mode timing analysis
 clock [672](#)
 pin [588](#), [589](#), [607](#), [631](#), [657](#)

port [629](#)
arrival time, checking if derived [971](#), [1009](#)
asynchronous clear pin
 active phase [212](#)
 path to libpin [189](#)
asynchronous preset pin
 active phase [212](#)
 path to libpin [189](#)
asynchronous reset pin
 force implementation [723](#)
 pin defined on CG cell [214](#)
asynchronous set pin
 force implementation [723](#)
attributes (liberty)
 MSV
 input_signal_level [225](#)
 level_shifter_enable_pin [219](#)
 output_signal_level [225](#)
attributes, alphabetical list of [69](#)

B

bit-blasting
 constants [294](#), [298](#)
 mapped ports [296](#)
 naming style for ports [243](#)
blackbox
 checking if instance is [955](#)
 library cell defined as [208](#)
 making DFT-controllable [1142](#)
body segment, name [1285](#)
buffer
 checking if instance is [946](#)
 library cell defined as [190](#)

C

capacitance
 net, reporting

pin [991](#)
 port [1026](#)
 subport [1042](#)
 nets, maximum
 design [220](#), [221](#), [557](#)
 port [651](#)
 of pin in library [213](#)
 per fanout in wire-load model [234](#)
 pin, reporting [974](#)
 port, reporting [1012](#)
 scaling factor [180](#)
 subport, reporting [1041](#)
cell
 area in design, reporting [932](#)
 area of library cell [188](#)
 count, reporting
 design [932](#)
 subdesign [1031](#)
 defined in library as
 adder [188](#)
 always-on cell [198](#)
 buffer [190](#)
 clock-gating cell [198](#)
 combinational [194](#)
 flip-flop [196](#)
 inverter [197](#)
 latch [199](#)
 level shifter [198](#)
 master-slave flop [200](#)
 master-slave LSSD flop [200](#)
 pad [204](#)
 sequential [207](#)
 SRPG cell [205](#)
 timing model [208](#)
 tristate [209](#)
 function [216](#)
 integrated CG cell functionality [194](#)
 internal power, overwriting [192](#)
 leakage power, overwriting [192](#)
 Liberty attributes [199](#), [205](#)
 path to cell with higher drive [217](#)
 path to cell with lower drive [219](#)
 use during mapping
 tool-controlled [210](#)
 user-controlled [190](#)
CGIC cell
 See integrated clock-gating cell
clock
 inverted sources [677](#)
 propagated information, reporting [974](#),
[976](#), [1013](#), [1015](#)
 sources [677](#)
 value of option
 divide_fall in define_clock [1053](#)
 divide_period in define_clock [1053](#)
 divide_rise in define_clock [1054](#)
 fall in define_clock [1056](#)
 period option in define_clock [1056](#)
 rise in define_clock [1056](#)
 clock delay
 network delay
 early-mode timing analysis
 clock [669](#)
 pin [580](#), [582](#), [604](#), [623](#), [654](#)
 port [621](#)
 late-mode timing analysis
 clock [670](#)
 pin [584](#), [586](#), [605](#), [627](#), [655](#)
 port [625](#)
 source delay
 early-mode timing analysis
 clock [673](#)
 pin [590](#), [592](#), [609](#), [634](#), [659](#)
 port [632](#)
 late-mode timing analysis
 clock [675](#)
 pin [594](#), [596](#), [610](#), [638](#), [660](#)
 port [636](#)
 clock edge, abstract segment
 head [1263](#), [1293](#)
 tail [1274](#), [1306](#)
clock gating
 enabling [1343](#)
 preventing on
 design [1357](#)
 instance [1371](#)
 subdesign [1408](#)
 clock nets, prefix for gated [1340](#)
 clock object edge, abstract segment
 head [1267](#), [1299](#)
 tail [1266](#), [1298](#)
 clock object, abstract segment
 head [1266](#), [1299](#)
 tail [1265](#), [1298](#)
 clock pin
 active phase [214](#)
 clock-gating cell [213](#)
 path to libpin [193](#)
 clock port, abstract segment
 head [1263](#), [1292](#)
 tail [1274](#), [1305](#)
 clock-gating (CG) logic

functionality [194](#)
 library cell defined as [198](#)
 library cell to use [1354](#), [1370](#), [1374](#),
[1407](#)
 maximum fanout [1358](#)
 minimum fanout [1358](#)
 prefix for modules, nets, ports [1340](#)
 user-defined module, specifying [1359](#),
[1409](#)

clock-gating cell
 asynchronous reset pin [214](#)
 clock pin [213](#)
 comb cell, defined in library as [198](#)
 enable pin [213](#)
 observable pin [213](#)
 output pin [214](#)

combinational cell
 library cell defined as [194](#)
 number of cells in library [186](#)

commands
 identify_shift_register_scan_segments
[1111](#)
 log file, specifying [244](#)

constants
 output, controlling bit-blasting [294](#)
 propagation through sequential
 cells [528](#)

constraint
 dynamic power [1366](#)
 leakage power [1367](#)

control logic
 naming, controlling [1108](#)

cost group
 slack, reporting [1060](#)
 weight [679](#)
 worst slacks of all endpoints [1062](#)

CSA transformation, controlling
 design [833](#)
 root [711](#)
 subdesign [901](#)

D

design
 cell area, reporting [932](#)
 cell count, reporting [932](#)
 clock gating, preventing [1357](#)
 CSA transformation, controlling [833](#)
 internal power, reporting [1363](#)
 leakage power, reporting [1364](#)

mapping to scan flops, preventing [1121](#)
 optimization, controlling [838](#)
 slack, reporting [942](#)
 switching power, reporting [1364](#)
 worst slacks of all endpoints [944](#)

design rule constraints
 controlling use of
 defined on external driver [650](#)
 defined on technology library [549](#)

maximum capacitance
 design [220](#), [221](#), [557](#)
 port [651](#)

maximum fanout
 design [220](#), [221](#), [558](#)
 port [652](#)

maximum transition
 design [559](#)
 port [653](#)

timing constraints, priority [529](#), [530](#),
[531](#), [533](#)

DFT clock domain, See test clock domain
 DFT clock domains, See test-clock domains
 DFT rule violation type, of abstract scan
 segment [1300](#)

DFT rule violation type, of flip-flop [1138](#)

drive strength
 path to cell with higher drive [217](#)
 path to cell with lower drive [219](#)

dynamic power
 design constraint [1366](#)

E

enable pin, defined on
 clock-gating cell [213](#)
 latch [218](#)

exception
 command used to create,
 returning [1064](#)

cost group of path group [1063](#)
 created for MAX delay analysis [681](#)

delay constraint
 path adjust [1063](#)
 path delay [1063](#)

endpoints, list of [1068](#)

multi-cycle
 capture clock shift value [1067](#)
 launch clock shift value [1067](#)

paths applied to [1066](#)

points to traverse, list of [1067](#)

p
 priority
 tool-defined [1066](#)
 user-defined [681](#)
 start points, list of [1065](#)
e
 external delay
 fall delay [1069](#)
 input or output delay, checking [1071](#)
 list of pins and ports, applying to [1070](#)
 reference clock [1069](#)
 reference clock edge, checking [1069](#)
x
 external driver
 fall transition time of input pin of [642](#)
 resistance for rise and fall
 transition [644](#)
ext
 external resistance time
 transition
 port [644](#)

F

f
 fanout
 maximum for all nets
 design [220, 221, 558](#)
 port [652](#)
 maximum, for clock-gating logic [1358](#)
 minimum, for clock-gating logic [1358](#)
 number of, outside design [646](#)
 of a libcell input pin [216](#)
feed
 feedthrough connection
 preserving [892](#)
feedt
 feedthrough pins
 controlling optimization [701](#)
fil
 files
 command file, specifying [244](#)
 hdl, search path [256](#)
 script files, search path [279](#)
flip-flop
 checking if instance is [948](#)
 DFT violation type [1138](#)
 library cell defined as [196](#)
 moving, controlling for optimization [842](#)
 scannable status [1136](#)
flip-flops
 constant 0 propagation, allowing [810](#)
 constant 1 propagation, allowing [810,](#)
 [852, 853](#)
 mapping to scan
 controlling [1126](#)
 preventing
 on design [1121](#)

 on instance [1133](#)
 on subdesign [1153](#)
 naming style if part of array [749, 750,](#)
 [752, 754, 755, 757, 759, 762, 763,](#)
 [764, 765, 766, 773, 775, 777, 782,](#)
 [783, 864, 865](#)
 preventing use of Qbar output [710, 843](#)
 stable states, implementing feedback
 path [736, 863](#)

H

hard
 hard region, marking
 instance [567](#)
 subdesign [667](#)
hdl
 hdl language, default version [252](#)
head
 head segment, name [1287](#)

I

input
 input delay [1069](#)
 checking if external delay is [1071](#)
 list of pins and ports, applying to [1070](#)
 reference clock [1069](#)
 reference clock edge, checking [1069](#)
input pin
 internal fanout [216](#)
 pin defined in library as [217](#)
 timing arcs [222](#)
input pragmas
 equivalent for
 asynch_set_reset [722](#)
 asynchro_reset [266](#)
 asynchro_reset_blk, in specified
 block [265](#)
 case_logic cover [267](#)
 case_logic no_priority [267](#)
 synchro_reset [270](#)
non synthesizable constructs
 indication of beginning of [281](#)
 indication of end of [281](#)
instance
 cell area, reporting [946](#)
 checking if
 black-box [955](#)
 buffer [946, 947](#)
 flip-flop [948](#)
 hierarchical [948](#)
 inverter [949](#)

latch [949](#)
 sequential [952](#)
 tristate [955](#)
 unresolved [859](#)
 clock gating, preventing [1371](#)
 hard region, marking as [567](#)
 internal power, reporting [1378](#)
 leakage power, reporting [1378](#)
 library cell mapped to [949](#)
 making DFT-controllable [1142](#)
 optimization, controlling [845](#), [855](#)
 primitive function, reporting [951](#)
 slack, reporting [952](#)
 subdesign name, reporting [953](#)
instances
 customizing names of generated instances [271](#)
 optimization, preventing [206](#)
 unloaded
 deletion, controlling [708](#), [900](#)
integrated clock-gating (CG) cell
 library cell to use [1354](#), [1370](#), [1374](#), [1407](#)
 library cell, functionality [194](#)
 selecting
 with glitch control [1359](#)
 with observability logic [1349](#)
 with reset logic [1350](#), [1369](#), [1404](#)
 with test control logic [1355](#)
internal pin
 pin defined in library as [217](#)
internal power
 of design, reporting [1363](#)
 of instance, reporting [1378](#)
 of libcell, overwriting [192](#)
inverter
 checking if instance is [949](#)
 library cell defined as [197](#)
isolation cells
 controlling insertion on constant nets [1438](#)

L

latch
 borrowed time, specifying
 design [551](#)
 instance [568](#)
 checking if instance is [949](#)
 library cell defined as [199](#)

pin mapping, controlling [723](#)
 stable states, implementing feedback path [748](#), [863](#)
latch enable pin
 active phase [218](#)
 path to libpin [199](#)
leakage power
 cell [192](#)
 design constraint [1367](#)
 of design, reporting [1364](#)
 of instance, reporting [1378](#)
 scaling factor [182](#)
level shifters
 controlling insertion on constant nets [1438](#)
libraries
 appending [274](#), [1428](#)
 for technology mapping,
 specifying [274](#), [1427](#)
 search path [273](#)
library
 Liberty attributes [182](#)
 version [187](#)
lockup elements, controlling type
 at end of scan chain [1291](#)
 between scan segments [1122](#)
log file, specifying [244](#)
logic, removing of unused logic [887](#)
loop breaker
 preventing addition [564](#)
loop, unfolding
 determining number of iterations [749](#)

M

mapping
 libraries used for [274](#), [1427](#)
 to scan
 controlling [1126](#)
 preventing
 for an instance [1133](#)
 for design [1121](#)
 for flip-flop [1132](#)
 for subdesign [1153](#)
 unmapping and remapping,
 controlling [1114](#)
 use of cell during mapping
 tool-controlled [210](#)
 user-controlled [190](#)
 master-slave flop

Attribute Reference for Encounter RTL Compiler

library cell defined as [200](#)
master-slave LSSD flop
 library cell defined as [200](#)
mbist clock
 period [1239](#)
 source of test clock waveform [1239](#)
memory, consumption [108](#)
message, indicating if user-defined [1482](#)
messages
 detailed explanation [131](#)
 explanation [134](#)
 identification number [131](#)
 number of occurrence [131](#)
 printing
 actual number printed [132, 133](#)
 controlling when to print [132](#)
 limiting number printed [132, 133](#)
 severity [133](#)
 verbosity, controlling [105](#)
modules, naming
 generated modules [720](#)
 parameterized modules [750](#)
multiplexers
 control generation of [750](#)

N

net
 area in design, reporting [940, 1037](#)
 capacitance, reporting
 pin [991](#)
 port [1026](#)
 subport [1042](#)
 drivers, number of [1000](#)
 ideal, checking [962, 1003, 1039](#)
 loads, number of [1000](#)
 pgpin connected to, reporting [994](#)
 pin connected to, reporting [973](#)
 pin driving, reporting [999](#)
 pins loading, reporting [1000](#)
 power, See also switching power
 probability
 computing [1390](#)
 source of value [1393, 1402, 1415](#)
 specifying [1389, 1396](#)
 resistance, reporting
 pin [992](#)
 port [1027](#)
 subport [1043](#)
 supply0 driven, reporting [997](#)

supply1 driven, reporting [998](#)
toggle rate
 computing [1391](#)
 source of value [1394, 1403, 1416](#)
 specifying [1390](#)

O

observability logic
 selecting CGIC cell with [1349](#)
observable pin, defined on CG cell [213](#)
operating conditions
 Liberty attributes [232](#)
 loading and reporting [928, 1426](#)
 use for timing [537, 1429](#)
optimization
 boundary, controlling [892](#)
 constant 0 propagation, controlling [810](#)
 constant 1 propagation,
 controlling [810, 852, 853](#)
 design, controlling [838](#)
 instance, controlling [845, 855](#)
 preventing on instances of libcell [206](#)
 subdesign, controlling [909](#)
 timing slack, of non-critical paths [822](#)
output delay [1069](#)
 checking if external delay is [1071](#)
 list of pins and ports, applying to [1070](#)
 reference clock [1069](#)
 reference clock edge, checking [1069](#)
output pin
 function of input pins [216](#)
 pin defined as [222](#)
 pin defined on clock-gating cell [214](#)
 timing arcs [217](#)
output, controlling verbosity [105](#)

P

pgpin
 net connected to, reporting [994](#)
phase inversion
 enabling [790](#)
Physical Library
 Importing [272, 364](#)
pin
 arrival time, checking if derived [971, 1009](#)
 capacitive load [213](#)

connected delay, reporting [966](#)
 defined in library as
 clock pin
 on clock-gating cell [213](#)
 enable pin
 on clock-gating cell [213](#)
 input pin [217](#)
 internal pin [217](#)
 IQ pin [218](#)
 IQN pin [218](#)
 observable pin [213](#)
 output pin [222](#)
 pad [222](#)
 tristate pin [226](#)
 direction [967](#), [993](#)
 endpoint of timing path [967](#)
 ideal driver, marking as [599](#)
 Liberty attributes [216](#), [219](#)
 library pin of mapped instance pin [972](#),
 [994](#)
 logic value, forcing for timing
 analysis [612](#)
 net capacitance, reporting [991](#)
 net connected to, reporting [973](#)
 net resistance, reporting [992](#)
 output pin, defined on clock-gating
 cell [214](#)
 output value, checking if
 computed [983](#), [985](#)
 phase of
 asynchronous clear [212](#)
 asynchronous preset [212](#)
 clock [214](#)
 latch enable [218](#)
 scan data input [225](#)
 scan enable [225](#)
 synchronous clear [226](#)
 synchronous enable [226](#)
 synchronous preset [226](#)
 probability
 computing [1381](#)
 source of value [1386](#)
 specifying [1380](#)
 propagated clock information,
 reporting [974](#)
 required time, checking if derived [962](#),
 [1002](#)
 slack, reporting [980](#)
 startpoint of timing path [983](#)
 timing arcs to pin, reporting [983](#)
 toggle rate

computing [1382](#)
 source of value [1387](#)
 specifying [1381](#)
 wire-load model, reporting [992](#)
 pin bus, paths to pins (bits) [996](#)
 pins, implementing asynchronous set and
 reset [723](#)
 port
 bus name, returning [1002](#)
 capacitance, reporting [1012](#)
 connect delay, reporting [1005](#)
 direction [1005](#)
 endpoint of timing path [1006](#)
 external capacitance [644](#)
 external delay [968](#), [1007](#)
 external resistance time
 transition [644](#)
 external wire capacitance [645](#)
 external wire resistance [645](#)
 fanouts outside design [646](#)
 ideal driver, identifying as [649](#)
 input driven by
 in case of fall transition [640](#)
 in case of rise transition [640](#)
 logic value, forcing for timing
 analysis [663](#)
 net capacitance, reporting [1026](#)
 net connected to, reporting [1011](#)
 net resistance, reporting [1027](#)
 number of parallel driving pins [643](#)
 output value, checking if
 computed [1020](#)
 probability
 computing [1397](#)
 specifying [1396](#)
 rise and fall delay, reporting [1012](#)
 slack, reporting [1017](#)
 slew time
 transition [647](#)
 startpoint of timing path [1020](#)
 toggle rate
 computing [1398](#)
 specifying [1396](#)
 wire-load model [646](#)
 port bus
 direction, reporting [1028](#)
 order of ports, reporting [1029](#)
 path to ports (bits), reporting [1028](#)
 ports
 complex, representation in netlist [737](#)
 controlling bit-blasting of mapped [296](#)

power

See also internal power, leakage power, and switching power
unit for reporting [1345](#)

power gating pin

active phase [224](#)

preserved logic

netlist editing, controlling [1473](#)

preserving

drivers of unresolved instance [859](#)
I/O pins of subdesign [892](#)
instances of libcell from optimization [206](#)
mapped instances in subdesign [833](#), [901](#)
order of elements in scan segment [1135](#), [1275](#), [1307](#)
pre-existing mapped instances [708](#), [900](#)
type of mapped flip-flops [1126](#)
unloaded instances [708](#), [900](#)

preset flop needed, only reset available [790](#)

probability

computing value [1390](#)
pin [1381](#)
port [1397](#)
subport [1412](#)

default value

for nets in design [1341](#), [1361](#)
for nets in hierarchical instance [1376](#)
definition [1341](#), [1361](#), [1380](#)
source of value [1386](#), [1393](#), [1402](#), [1415](#)
specifying user value [1389](#)
pin [1380](#)
port [1396](#)
subport [1411](#)

process, operating condition in library [232](#)

program

name [110](#)
short name [110](#)
version [110](#)
wordsize [108](#)

R

RC component, implementation (speed)
choosing [172](#)
reporting [171](#)

recursive instantiations

setting number of elaborations [750](#)

registers

optimizing X state [812](#)

unused, preserving [759](#), [865](#)

required time, checking if derived [962](#), [1002](#)

reset flop needed, only preset available [790](#)

reset latch, force mapping to [723](#)

resistance

of constant net [959](#)

resistance of net, reporting

pin [992](#)

port [1027](#)

subport [1043](#)

RTL speculation, controlling

root [714](#)

S

scaling factor

capacitance [180](#)

leakage power [182](#)

timing [186](#)

scan chain

tool-created

control test signal for mux [1260](#)

elements, number of [1257](#)

elements, path to [1255](#)

scan-data input [1258](#)

scan-data output [1259](#)

scan-data output, shared [1260](#)

shift-enable port [1260](#)

terminal lockup element [1261](#)

test clock domain associated with edge [1254](#)

name [1254](#), [1268](#)

user-specified

body segment name [1285](#)

completely [1285](#)

control test signal for mux [1290](#)

head segment name [1287](#)

maximum length [1287](#)

scan-data input [1289](#)

scan-data output [1289](#)

scan-data output, shared [1290](#)

shift-enable port [1290](#)

tail segment name [1291](#)

terminal lockup element, type [1291](#)

Attribute Reference for Encounter RTL Compiler

test clock domain associated with
edge [1287](#)
name [1286](#)

scan chain inversion, preventing [1127](#)

scan chains (all)
lockup elements, controlling type [1122](#)
maximum length of, specifying [1124](#)
minimum number of, specifying [1124](#)
mixing edges of same clock on [1125](#)

scan clock pins, controlling
connection [1120](#)

scan data input pin
active phase [225](#)
path to libpin [207](#)

scan enable pin
active phase [225](#)
path to libpin [206](#)

scan flip-flops
mixing edges of same clock on
chain [1125](#)
output connection, controlling [1127](#)

scan segment
tool-created
elements
number of [1270](#)
path to [1268](#)

scan data input [1271](#)

scan data output [1272](#)

shift enable
confirming if connected [1264](#)
port [1273](#)

type [1275](#)

user-specified
elements
number of [1302](#)
path to [1300](#)

scan data input [1304](#)

scan data output [1304](#)

shift enable
confirming if connected [1294](#)

type [1307](#)

scan style, controlling [1110](#)

scan-data input
connection, controlling [1119](#)
name, controlling [1108](#)

of tool-created scan chain [1258](#)

of tool-created scan segment [1271](#)

of user-specified scan chain [1289](#)

of user-specified scan segment [1304](#)

scan-data output
name, controlling [1108](#)

of tool-created scan chain [1259](#)

of tool-created scan segment [1272](#)

of user-specified scan chain [1289](#)

of user-specified scan segment [1304](#)

shared, of tool-created chain [1260](#)

shared, of user-specified chain [1290](#)

script
begin, keyword [278](#)
end, keyword [278](#)

search path
hdl files [256](#)
implicit finds [275](#)
script files [279](#)
technology libraries [273](#)

sequential cell
checking if instance is [952](#)
integrated clock cell functionality [194](#)
library cell defined as [207](#)
number of cells in library [187](#)

sequential instances
deletion, controlling
design [709](#)
subdesign [833, 901](#)

logic constant propagation,
controlling [528](#)

set and reset signals, preserving [725](#)

set latch, force mapping to [723](#)

shift-enable pins, controlling
connection [1120](#)

shift-enable port
active value
tool-created segment [1262](#)
user-specified segment [1292](#)

associated with
tool-created chain [1260](#)

tool-created segment [1273](#)

user-specified abstract
segment [1305](#)

user-specified chain [1290](#)

confirming if connected
tool-created scan segment [1253, 1264](#)

user-specified scan segment [1294](#)

name, controlling [1108](#)

slack
cost group, reporting [1060](#)

design, reporting [942](#)

instance, reporting [952](#)

non-critical paths, optimization [822](#)

pin, reporting [980](#)

port, reporting [1017](#)

- worst slacks of all endpoints
 - cost group [1062](#)
 - design [944](#)
- slew rate
 - controlling
 - design [559](#)
 - port [653](#)
 - controlling
 - library pin [220](#)
- slew time
 - transition
 - port [647](#)
- SRPG cell
 - library cell defined as [205](#)
- state retention registers
 - cells for mapping [1468](#)
 - restore signal [1469](#)
 - save signal [1469](#)
 - secondary domain [1470](#)
- subdesign
 - boundary optimization [892](#)
 - cell area, reporting [1031](#)
 - cell count, reporting [1031](#)
 - checking if user module [1037](#)
 - clock gating, preventing [1408](#)
 - CSA transformation, controlling [901](#)
 - hard region, marking as [667](#)
 - implementation, choosing [172](#)
 - instance of, reporting [953](#)
 - instances of, list [1035](#)
 - mapping to scan flop, preventing [1153](#)
 - optimization, controlling [909](#)
 - speed implementation, reporting [171](#)
- subport
 - bus name, returning [1039](#)
 - capacitance, reporting [1041](#)
 - direction [1040](#)
 - net capacitance, reporting [1042](#)
 - net connected to, reporting [1041](#)
 - net resistance, reporting [1043](#)
 - probability
 - computing [1412](#)
 - specifying [1411](#)
 - toggle rate
 - computing [1413](#)
 - specifying [1412](#)
- subport bus
 - direction [1044](#)
 - order of subports [1045](#)
 - path to subports (bits) [1044](#)
- switching activities, controlling
 - simulation [1344](#)
 - switching power, reporting
 - of all nets in design [1364](#)
 - of net [1392](#)
 - connected to pin [1385](#)
 - connected to port [1401](#)
 - connected to subport [1414](#)
 - of output nets of instance [1379](#)
 - synchronous clear pin
 - active phase [226](#)
 - path to libpin [207](#)
 - synchronous enable pin
 - active phase [226](#)
 - path to libpin [207](#)
 - synchronous preset pin
 - active phase [226](#)
 - path to libpin [208](#)

T

- tail segment, name [1291](#)
- TAP signal
 - type, identifying [1208](#)
- temperature, operating condition in library [233](#)
- test clock
 - associated with flip-flop
 - active edge [1137, 1138](#)
 - name [1137](#)
 - controllability in test mode,
 - specifying [1313](#)
 - fall value [1316, 1325](#)
 - mixing edges of same clock [1125](#)
 - period [1317, 1328](#)
 - rise value [1317, 1330](#)
 - source of creation [1319](#)
 - source of test clock waveform [1318](#)
- test clock domain
 - edge
 - tool-created chain [1254](#)
 - user-specified chain [1287](#)
 - name
 - tool-created chain [1254](#)
 - user-specified chain [1286](#)
- test clocks
 - assigned by DFT [1099](#)
 - internal, identified by DFT [1098](#)
- test control logic
 - location in CGIC cell [1355](#)
- test signal

-
- active value, specifying [1320](#)
 - associated port [1328](#)
 - connecting test pins of CGIC cells [1360, 1375, 1409](#)
 - designating as default shift-enable [1322](#)
 - marking as ideal [1326](#)
 - source of creation [1331](#)
 - type, identifying [1331](#)
 - test-clock domains
 - assigned by DFT [1099](#)
 - timing analysis
 - constant value, forcing on pin [612](#)
 - static, disabling timing arcs [564](#)
 - timing arc
 - Liberty attributes [229](#)
 - originating pin [229](#)
 - type [230](#)
 - timing arcs
 - disabled, reporting [953](#)
 - disabling on instance [564](#)
 - disabling on the library cell
 - tool-controlled [230](#)
 - from input pin [222](#)
 - to output pin [217](#)
 - to pin, reporting [983](#)
 - timing case analysis [528](#)
 - timing constraints
 - design rule constraints, priority [529, 530, 531, 533](#)
 - timing model
 - library cell defined as [208](#)
 - timing model, number of models in library [187](#)
 - timing, scaling factor [186](#)
 - toggle rate
 - computing value [1391](#)
 - pin [1382](#)
 - port [1398](#)
 - subport [1413](#)
 - default value
 - for nets in design [1342, 1363](#)
 - for nets in hierarchical instance [1377](#)
 - source of net value [1387, 1394, 1403, 1416](#)
 - specifying user value [1390](#)
 - pin [1381](#)
 - port [1396](#)
 - subport [1412](#)
 - unit [1346](#)
 - transformations
 - phase inversion [790](#)
 - pin phase mapping [796, 849, 907](#)
 - transition
 - maximum for all nets
 - design [559](#)
 - port [653](#)
 - tristate cell
 - checking if instance is [955](#)
 - library cell defined as [209](#)
 - tristate pin, defined in library as [226](#)

U

- units
 - power [1345](#)
 - toggle rate [1346](#)

V

- verbosity, messages [105](#)
- version
 - library [187](#)
 - program [110](#)
- VHDL
 - enforcing Irm compliance [262](#)
 - specifying arithmetic library [261](#)
 - specifying case [261](#)
 - specifying preferred architecture [262](#)
 - specifying read version [264](#)
- voltage, operating condition in library [233](#)

W

- wire delay, estimation method used in library [233](#)
- wire-load mode, selecting [542](#)
- wire-load model
 - current model for design [945](#)
 - current model for subdesign [1038](#)
 - fanout capacitance, specifying [234](#)
 - Liberty attributes [235](#)
 - library default
 - returning or specifying [180](#)
 - pin, reporting [992](#)
 - port [646](#)
 - selection, controlling design [547](#)

Attribute Reference for Encounter RTL Compiler

subdesign [666](#)
wire-load selection table, controlling
use [543](#), [1430](#)
wordsize, program [108](#)