



A Siemens Business

---

# Power Aware Simulation User's Manual

Software Version 10.7c

---

© 2010-2018 Mentor Graphics Corporation  
All rights reserved.

This document contains information that is proprietary to Mentor Graphics Corporation. The original recipient of this document may duplicate this document in whole or in part for internal business purposes only, provided that this entire notice appears in all copies. In duplicating any part of this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use and distribution of the proprietary information.

Note - Viewing PDF files within a web browser causes some links not to function (see [MG595892](#)).  
Use HTML for full navigation.

This document is for information and instruction purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

**MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.**

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**U.S. GOVERNMENT LICENSE RIGHTS:** The software and documentation were developed entirely at private expense and are commercial computer software and commercial computer software documentation within the meaning of the applicable acquisition regulations. Accordingly, pursuant to FAR 48 CFR 12.212 and DFARS 48 CFR 227.7202, use, duplication and disclosure by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in the license agreement provided with the software, except for provisions which are contrary to applicable mandatory federal laws.

**TRADEMARKS:** The trademarks, logos and service marks ("Marks") used herein are the property of Mentor Graphics Corporation or other parties. No one is permitted to use these Marks without the prior written consent of Mentor Graphics or the owner of the Mark, as applicable. The use herein of a third-party Mark is not an attempt to indicate Mentor Graphics as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A current list of Mentor Graphics' trademarks may be viewed at: [mentor.com/trademarks](http://mentor.com/trademarks).

The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

**End-User License Agreement:** You can print a copy of the End-User License Agreement from: [mentor.com/eula](http://mentor.com/eula).

Mentor Graphics Corporation  
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777  
Telephone: 503.685.7000  
Toll-Free Telephone: 800.592.2210  
Website: [mentor.com](http://mentor.com)  
Support Center: [support.mentor.com](http://support.mentor.com)

Send Feedback on Documentation: [support.mentor.com/doc\\_feedback\\_form](http://support.mentor.com/doc_feedback_form)

# Table of Contents

---

<b>Chapter 1</b>	
<b>Getting Started With Power Aware Simulation .....</b>	<b>15</b>
Introduction to Power Aware Simulation .....	15
Power Aware in Your Design Flow.....	16
Documentation—Scope and Organization .....	18
How to Use This Manual .....	19
<b>Chapter 2</b>	
<b>Concepts for Using Power Aware Simulation .....</b>	<b>21</b>
Power Aware Models.....	22
Power Aware Model Overview .....	22
Corruption Models .....	23
Isolation Models .....	24
Retention Models .....	25
Bias Models .....	28
<b>Chapter 3</b>	
<b>Power Aware Simulation .....</b>	<b>29</b>
Required Inputs .....	30
Power Aware Simulation Commands .....	30
Power Aware Simulation Flows .....	34
Using the Standard Power Aware Simulation Flow.....	35
Using the Delayed Optimization Flow .....	37
PDU-Based Simulation.....	38
Power Aware in Interactive Mode .....	40
Macro Models Concepts .....	46
Non-Power Aware HDL Models .....	46
Power Aware HDL Models .....	46
Extended Power Aware HDL Models .....	47
Liberty Models of Hard Macros .....	48
Liberty Library Models .....	52
Liberty File Concepts .....	52
Using Liberty Source Files with your Power Aware Simulation .....	53
Specifying Liberty Cells via a Liberty Attribute Library.....	53
Specifying Liberty Source Files Directly.....	53
Liberty Attribute Library Management .....	54
Creating a Liberty Attribute Library .....	54
Updating a Liberty Attribute Library .....	54
Refreshing a Liberty Attribute Library.....	55
Checker Modules and the bind_check Command .....	56
Power Aware Simulation Debug .....	60
Power Intent Application to Mixed RTL and Gate-Level Designs .....	63

Hard Macro Cell Detection . . . . .	63
Gate-level Cell Detection . . . . .	68
Power Management Cell Detection . . . . .	69
Automatic Insertion of Power Management Cells . . . . .	70
Automatic Corruption and Retention of UDPs . . . . .	71
UPF Supply Connections . . . . .	73
Implicit Connections . . . . .	73
Explicit Connections . . . . .	73
Automatic Connections . . . . .	75
Simulation Semantics for UPF Supply Connections . . . . .	78
Value Conversion Tables . . . . .	79
Using VCT Commands . . . . .	79
Connections Using Value Conversion Tables (VCTs) . . . . .	81
Questa-specific Value Conversion Tables . . . . .	83
Defining Isolation . . . . .	83
Defining Retention . . . . .	87
-retention_supply_set . . . . .	87
-no_retention . . . . .	88
-use_retention_as_primary . . . . .	90
<b>Chapter 4</b>	
<b>Power Aware Reports . . . . .</b>	<b>93</b>
Generating Reports for Power Aware . . . . .	93
Power Aware Reports Reference . . . . .	94
Architecture Report . . . . .	95
Ack Port Driver Report . . . . .	105
Connection Report . . . . .	106
Design Element Report . . . . .	109
Dynamic UPF Report . . . . .	113
Macro Cell Report . . . . .	116
Missing Liberty Cell Report . . . . .	118
Multi-Rail Cell Report . . . . .	118
Non-Retention Synchronous Flop Report . . . . .	119
Power Cell Report . . . . .	120
PST Analysis Report . . . . .	122
Source Sink Path Report . . . . .	133
Static Check Report . . . . .	135
Supply Network Initialization Report . . . . .	139
Unassociated Cell Report . . . . .	140
<b>Chapter 5</b>	
<b>Power Aware Checks . . . . .</b>	<b>143</b>
Power Aware Checks Overview . . . . .	144
Isolation Checks . . . . .	146
Level Shifter Checks . . . . .	146
Path Analysis Checks . . . . .	147
Retention Checks . . . . .	147
Back-to-Back Checks . . . . .	147

## Table of Contents

---

Power State Composition . . . . .	148
Power Aware Checks Command Reference . . . . .	154
Static RTL Checks . . . . .	155
Static RTL Isolation Checks. . . . .	155
Static RTL Level Shifter Checks . . . . .	157
Static RTL Path Analysis Checks. . . . .	159
Static RTL Back-to-Back Checks . . . . .	161
Static GLS Checks . . . . .	163
Static GLS Isolation Checks. . . . .	164
Static GLS Level Shifter Checks . . . . .	168
Static GLS Retention Checks . . . . .	173
Static GLS Back-to-Back Checks . . . . .	175
Static GLS Miscellaneous Checks . . . . .	176
Static GLS Switch Checks . . . . .	177
Dynamic Checks . . . . .	179
Dynamic Isolation Checks . . . . .	179
Dynamic Level Shifter Checks. . . . .	183
Dynamic Retention Checks . . . . .	185
Dynamic Miscellaneous Checks. . . . .	190
Quick Reference of Static RTL and Dynamic Checks . . . . .	193
Power Aware Checks Control . . . . .	196
Controlling Checks With the pa_checks Command . . . . .	197
Controlling Checks With the pa msg Command . . . . .	199
Precedence Order of Arguments . . . . .	201
Pathname Convention in Arguments . . . . .	202
 <b>Chapter 6</b>	
<b>Power Aware Coverage . . . . .</b>	<b>205</b>
Power Aware Coverage Overview. . . . .	205
Power Aware Coverage Flow . . . . .	206
Power Aware Coverage Collection. . . . .	208
Collecting Coverage Data of Dynamic Checks . . . . .	208
Collecting Coverage Data of Power States and Transitions. . . . .	209
Generating UCDB and Test Plan Files . . . . .	210
Power Aware Coverage Analysis . . . . .	212
Generating the Power Aware Coverage Report . . . . .	212
Analyzing Coverage Using the Power Aware Test Plan . . . . .	214
Generating the Basic Combined Coverage Report. . . . .	220
Generating the Advanced Combined Coverage Report . . . . .	221
Coverage Support of UPF Objects. . . . .	223
 <b>Chapter 7</b>	
<b>Graphical Display of Power Aware Operations and Results . . . . .</b>	<b>237</b>
UPF Object Display . . . . .	238
UPF Object Concepts . . . . .	238
Displaying UPF Objects in the GUI . . . . .	238
Power Aware Source Window . . . . .	241
Finding Power Aware Drivers . . . . .	241

UPF Color Scheme in the Source Window .....	243
UPF-specific Context Menu in the Source Window .....	244
Show Drivers Control Bar .....	245
Power Aware Schematic Display .....	246
Top-Down Debugging (From the Test Bench).....	246
Bottom-Up Debugging (From the Design Under Test) .....	247
Schematic Window Features for Debugging .....	248
Power Aware Waveform Display .....	254
Power Aware Waveform Concepts.....	254
Using Power Aware Highlighting.....	255
Power State and Transition Display.....	256
Power State and Transition Concepts.....	256
Displaying Power Aware State Machines .....	257
Power Aware State Machine List Window.....	258
Power Aware State Machine Viewer Window.....	260
Power Aware Static Check Display.....	265
Viewing Static Checks in the Results Analysis Window .....	265
GUI Elements of the Results Analysis Window for Static Checks .....	266
<b>Chapter 8</b>	
<b>Power Aware Information Model .....</b>	<b>269</b>
Creating and Loading Power Aware Information Model Database.....	269
Tcl Commands.....	270
Running Tcl Commands in the Interactive Mode.....	270
Running Tcl Commands From the VSIM Prompt .....	270
Supported Tcl Commands .....	272
upf_object_in_class .....	272
upf_query_object_pathname .....	273
upf_query_object_properties .....	273
upf_query_object_type .....	274
HDL Package Functions .....	275
Supported HDL Package Functions .....	275
Using the HDL Package Functions for Power Aware Coverage .....	277
<b>Appendix A</b>	
<b>Power Aware-related Commands .....</b>	<b>281</b>
Questa SIM Commands and Arguments .....	282
Power Aware-specific Arguments to vopt and vsim .....	282
Usage of vopt -pa_enable and -pa_disable .....	285
Argument to Exclude Design Element .....	297
Command to Control Dynamic Checks .....	300
Arguments to Control Power Aware Message Severity .....	308
Supported UPF Extensions .....	309
Tcl Commands.....	317
describe_state_cross_coverage .....	318
getenv .....	319
pa_checks .....	320
save_checks_config .....	325

## Table of Contents

---

set_corruption_extent . . . . .	326
set_feedthrough_object . . . . .	327
set_pgpin_vct . . . . .	328
set_related_supply_net . . . . .	329
Voltage Level-Shifting (Multi-Voltage Analysis) . . . . .	332
Power State Tables . . . . .	332
Level Shifter Specification . . . . .	334
Isolation and Level Shifting Restriction on a Port . . . . .	336
Simulation of Designs Containing Macromodels . . . . .	341
<b>Appendix B</b>	
<b>Model Construction for Power Aware Simulation . . . . .</b>	<b>347</b>
Basic Model Structure . . . . .	347
Named Events in Power Aware . . . . .	349
Attributes . . . . .	350
Model Interface Ports . . . . .	351
Model Construction Examples . . . . .	353
<b>Appendix C</b>	
<b>UPF Commands and Reference . . . . .</b>	<b>357</b>
Supported UPF Versions . . . . .	357
Supported UPF Commands . . . . .	359
add_domain_elements . . . . .	362
add_port_state . . . . .	362
add_power_state . . . . .	363
add_pst_state . . . . .	364
add_state_transition . . . . .	364
add_supply_state . . . . .	365
apply_power_model . . . . .	366
associate_supply_set . . . . .	366
begin_power_model . . . . .	367
bind_checker . . . . .	367
connect_logic_net . . . . .	369
connect_supply_net . . . . .	370
connect_supply_set . . . . .	370
create_composite_domain . . . . .	371
create_hdl2upf_vct . . . . .	375
create_logic_net . . . . .	375
create_logic_port . . . . .	376
create_power_domain . . . . .	376
create_power_state_group . . . . .	377
create_power_switch . . . . .	378
create_pst . . . . .	379
create_supply_net . . . . .	380
create_supply_port . . . . .	383
create_supply_set . . . . .	383
create_upf2hdl_vct . . . . .	384
describe_state_transition . . . . .	384

end_power_model . . . . .	385
load_simstate_behavior . . . . .	386
load_upf . . . . .	386
load_upf_protected . . . . .	387
map_isolation_cell . . . . .	388
map_level_shifter_cell . . . . .	389
map_retention_cell . . . . .	390
name_format . . . . .	390
save_upf . . . . .	391
set_design_attributes . . . . .	392
set_design_top . . . . .	393
set_domain_supply_net . . . . .	394
set_equivalent . . . . .	394
set_isolation . . . . .	395
set_isolation_control . . . . .	397
set_level_shifter . . . . .	397
set_partial_on_translation . . . . .	400
set_pin_related_supply . . . . .	402
set_port_attributes . . . . .	403
set_power_switch . . . . .	406
set_repeater . . . . .	407
set_retention . . . . .	408
set_retention_control . . . . .	410
set_retention_elements . . . . .	410
set_scope . . . . .	411
set_simstate_behavior . . . . .	413
set_variation . . . . .	414
upf_version . . . . .	415
use_interface_cell . . . . .	415
Supported Query Commands . . . . .	417
find_objects . . . . .	417
query_cell_instances . . . . .	419
query_cell_mapped . . . . .	420
query_design_attributes . . . . .	420
query_isolation . . . . .	421
query_port_state . . . . .	421
query_power_domain . . . . .	422
query_power_domain_element . . . . .	422
query_power_state . . . . .	423
query_power_switch . . . . .	423
query_pst . . . . .	424
query_pst_state . . . . .	424
query_retention . . . . .	425
query_retention_control . . . . .	425
query_supply_net . . . . .	426
query_supply_port . . . . .	426
Supported UPF Attributes . . . . .	427

## Table of Contents

---

<b>Appendix D</b>	
<b>Supplemental Information.....</b>	<b>433</b>
Power Aware Verification of ARM-Based Designs .....	434
Abstract.....	434
Introduction.....	435
Active Power Management.....	435
Power Management Techniques.....	435
Power Management Specification .....	436
Power Management Architecture .....	437
Power Managed Behavior.....	444
Power Control Logic.....	444
Power Aware Verification Flow.....	445
Summary.....	448

## Index

## End-User License Agreement



# List of Figures

---

Figure 1-1. Typical Location of Power Aware Simulation in Design Flow .....	18
Figure 3-1. Simulation Flow for Multiple PDUs .....	40
Figure 3-2. Viewing VCT Information in the Wave Window.....	79
Figure 5-1. Power Aware Checks .....	145
Figure 6-1. Power Aware Coverage Flow .....	207
Figure 6-2. Power Aware Test Plan (Flat View) in the Verification Management Tracker Window .....	216
Figure 6-3. Power Aware Test Plan (Flat View) in an XML Format .....	217
Figure 6-4. Power Aware Test Plan (Hierarchical View) in the Verification Management Tracker Window .....	218
Figure 6-5. Power Aware Test Plan (Hierarchical View) in an XML Format .....	219
Figure 7-1. UPF Objects in the Structure (sim), Objects, and Wave Windows .....	240
Figure 7-2. Color-Coded HDL Design Elements .....	251
Figure 7-3. Gray Area Indicates a Power Domain That is Off .....	252
Figure 7-4. UPF Source File: Right-Click and Choose Power Domain .....	252
Figure 7-5. UPF Source File: Hover the Mouse and View Tool Tip .....	253
Figure 7-6. Power Aware Highlighting in the Wave Window .....	255
Figure 7-7. Power Aware State Machine List Example .....	258
Figure 7-8. Power Aware State Machine Viewer Window Example .....	261
Figure 7-9. Results Analysis Window With Static Check Results .....	267
Figure A-1. Supply Paths to Power Domains .....	338
Figure A-2. Multiple Isolation Cells .....	340
Figure A-3. Design Consisting of Hard Macros .....	343
Figure C-1. Design Hierarchy .....	419
Figure D-1. A Power-Managed Design.....	438
Figure D-2. An ARM-based SoC with Active Power Management .....	446



# List of Tables

---

Table 3-1. Power Aware Debug Support for Live-Sim and Post-Sim Modes .....	61
Table 4-1. Power Aware Reports .....	94
Table 4-2. Power Domain Section Fields .....	96
Table 4-3. Power Switch Section Fields .....	97
Table 4-4. Retention Strategy Section Fields .....	98
Table 4-5. Isolation Strategy Section Fields .....	99
Table 4-6. Level Shifter Strategy Section Fields .....	100
Table 4-7. Power State Table Section Fields .....	101
Table 4-8. Power State Information Section Fields .....	102
Table 4-9. Ack Port Driver Report Fields .....	105
Table 4-10. Port or Net Connection Section Fields .....	106
Table 4-11. Root Supply Connection Section Fields .....	108
Table 4-12. Design Element Scope Section Fields .....	109
Table 4-13. Corrupted Signal Section Fields .....	110
Table 4-14. State Element Section Fields .....	110
Table 4-15. Dynamic UPF Report Fields .....	114
Table 4-16. Macro Cell Report Fields .....	116
Table 4-17. Missing Liberty Cell Report Fields .....	118
Table 4-18. Multi-Rail Cell Report Fields .....	119
Table 4-19. Power Cell Section Fields .....	121
Table 4-20. Level Shifter Cell Section Fields .....	121
Table 4-21. [BOUNDARY_ANALYSIS] Section Fields .....	123
Table 4-22. [OBJ] Section Fields .....	124
Table 4-23. [APS] Section Fields .....	125
Table 4-24. [PST] Section Fields .....	126
Table 4-25. [CPST] Section Fields .....	127
Table 4-26. [DUPLICATE_PSTS] Section Fields .....	127
Table 4-27. [UNDETERMINED_STATES] Section Fields .....	128
Table 4-28. [UNREACHABLE_STATES] Section Fields .....	129
Table 4-29. Source Sink Report Fields .....	134
Table 4-30. Static Checks Summary Report Section Fields .....	136
Table 4-31. Domain Wise Static Checks Detailed Report Section Fields .....	136
Table 4-32. Unassociated Cell Report Fields .....	141
Table 5-1. Power Aware Checks Command Reference .....	154
Table 5-2. Static RTL Checks .....	155
Table 5-3. Static RTL Isolation Checks .....	156
Table 5-4. Static RTL Level Shifter Checks .....	158
Table 5-5. Static RTL Path Analysis Checks .....	160
Table 5-6. Static RTL Back-to-Back Checks .....	162
Table 5-7. Static GLS Checks .....	163

---

Table 5-8. Static GLS Isolation Checks . . . . .	165
Table 5-9. GLS Static Level Shifter Checks . . . . .	169
Table 5-10. Static GLS Retention Checks . . . . .	174
Table 5-11. Static GLS Back-to-Back Checks . . . . .	175
Table 5-12. Miscellaneous Checks . . . . .	177
Table 5-13. Static GLS Switch Checks . . . . .	177
Table 5-14. Dynamic Checks . . . . .	179
Table 5-15. Dynamic Isolation Checks . . . . .	180
Table 5-16. Dynamic Level Shifter Checks . . . . .	184
Table 5-17. Dynamic Retention Checks . . . . .	186
Table 5-18. Dynamic Miscellaneous Checks . . . . .	190
Table 5-19. Argument Values for Static RTL and Dynamic Checks . . . . .	193
Table 7-1. Source Window: UPF Context Menu . . . . .	244
Table 7-2. Power Aware State Machine List Window Columns . . . . .	259
Table 7-3. Power Aware State Machine List Window Popup Menu . . . . .	259
Table 7-4. PA State Machines Menu . . . . .	259
Table 7-5. Power Aware State Machine Viewer Window Popup Menu . . . . .	262
Table 7-6. FSM View Menu, Specific to Power Aware State Machines . . . . .	263
Table 8-1. Supported Tcl Commands . . . . .	272
Table 8-2. List of Supported UPF 3.0 HDL Package Functions for SystemVerilog . . . . .	275
Table 8-3. List of Unsupported Properties . . . . .	276
Table A-1. Power Aware Arguments for vopt . . . . .	282
Table A-2. Power Aware Arguments for vsim . . . . .	285
Table A-3. Power Aware Actions for vopt -pa_enable and -pa_disable . . . . .	286
Table A-4. Argument Values for vopt -pa_upfextensions . . . . .	310
Table A-5. Power State Conversion Warning Messages . . . . .	333
Table C-1. Power Aware Simulation Modes . . . . .	358
Table C-2. List of Supported UPF Commands . . . . .	359
Table C-3. List of Supported Query Commands . . . . .	417
Table C-4. Supported UPF Attributes . . . . .	427
Table C-5. Supported Simulator-specific UPF Attributes . . . . .	428

# Chapter 1

## Getting Started With Power Aware Simulation

---

This chapter provides a brief overview of Power Aware simulation and how it fits in your conventional design flow.

---

### Note

---

 If you are using ModelSim SE, the Power Aware functionality requires the purchase of an additional license. Refer to “[License Feature Names](#)” in the *Installation and Licensing Guide* for more information, or contact your Mentor Graphics sales representative.

Further, some Power Aware commands, features, and capabilities are available only for the Questa SIM simulator—these are indicated, wherever necessary throughout the manual.

---

<b>Introduction to Power Aware Simulation .....</b>	<b>15</b>
<b>Power Aware in Your Design Flow .....</b>	<b>16</b>
<b>Documentation—Scope and Organization .....</b>	<b>18</b>
<b>How to Use This Manual .....</b>	<b>19</b>

## Introduction to Power Aware Simulation

Some designs require that you minimize dynamic and static power consumption. A common low-power design technique—power gating—involves switching off certain portions of the design when their operation is not needed and restoring power when operation is needed again. Other low-power design techniques include the use of multi-voltage supplies, state retention, isolation, and level shifting.

The Questa SIM simulator supports Power Aware simulation for VHDL or Verilog designs that use the above techniques in both register transfer level (PA-RTL) and gate-level (PA-GL) simulation.

To apply Power Aware simulation, use your conventional Questa SIM simulation flow, along with some power-specific arguments to the `vopt` and `vsim` commands, along with a power intent specification written using the Unified Power Format (UPF) defined in IEEE Std 1801.

With Power Aware simulation, you can perform functional verification of low-power designs together with the power management structures defined by your power intent. Different types of power gating design management structures can be verified, such as:

- Multiple switchable power domains with a single voltage.
- Multiple switchable power domains with different (fixed) voltages per domain.
- Power domains that can be put into bias mode for state retention with low static leakage.

To verify these structures, create a power intent specification in the UPF format by defining the following:

- The power domains within your design and the instances that belong to each power domain.
- The power states of those power domains, their corresponding primary supplies, and the simstates associated with primary supply states.
- The strategies for inserting, powering, and controlling retention cells in a power domain, to retain state during power down.
- The strategies for inserting, powering, and controlling isolation cells and level shifter cells at power domain boundaries, to mediate interactions between power domains in different states or at different voltage levels.
- The supply networks, including supply ports, nets, and power switches required to provide power to each domain.

## Power Aware in Your Design Flow

Before you begin to add power-gating to your design environment, you should evaluate where you are in your overall design flow.

[Figure 1-1](#) shows an approximation of a typical design sequence and where power-gating might occur in that sequence.

Before running Power Aware simulation, work on the following stages of your design:

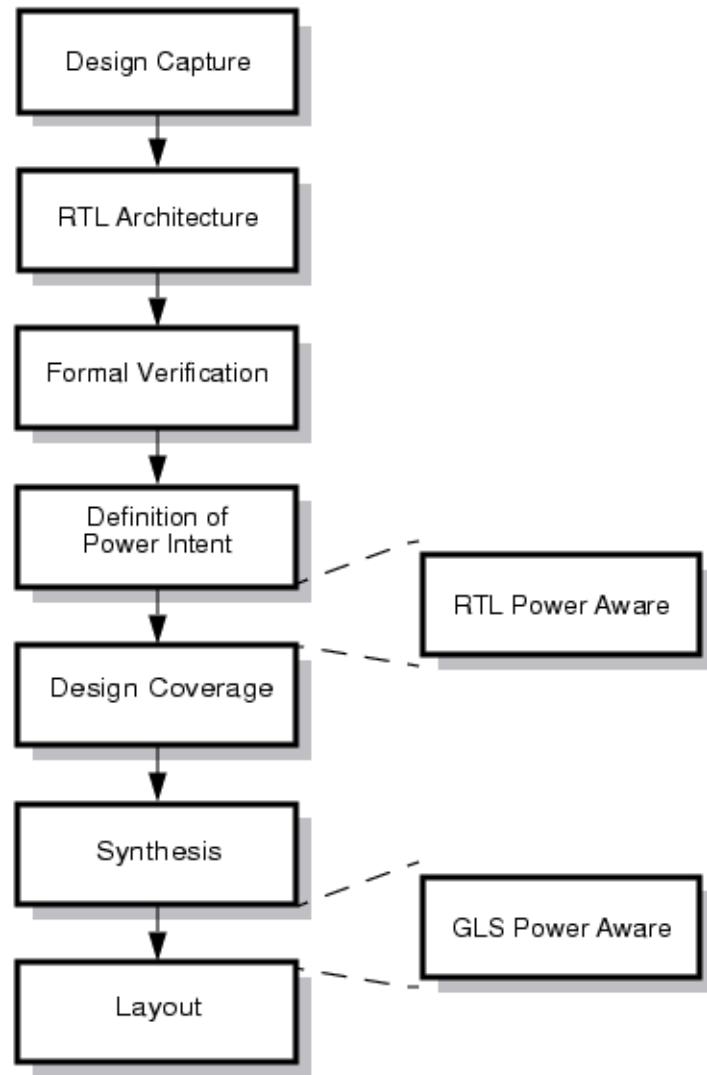
- Create your design
- Define your RTL architecture
- Perform Formal verification
- Define your power intent

After running a Power Aware RTL simulation, use the results to make appropriate topology or performance changes to your power-sensitive design blocks. After a gate-level simulation (GLS), make library cell changes based on the performance characteristics.

The scope of Power Aware simulation as a low-power solution spans multiple manifestations of design architecture.

- Power Aware Simulation — Simulation that includes active power management elements and their behavior. The power management architecture is specified in UPF; the behavior of those elements is inferred from the UPF specification.
- Power Aware Verification — Collaborative usage of various products and methods for verifying that a design operates correctly under active power management. These include Power Aware simulation (for verifying the correct operation of the power management architecture), formal verification (for verifying the correct operation of power control logic), and hardware/software co-verification (for verifying that software power control interacts correctly with power control logic). This relates to all of the components of the Questa Verification Platform that can be used for Power Aware Verification, including Questa PASim, Questa ADMS, Questa Formal, Questa Codelink, and Questa Verification Management.
- Power Efficient Design — Design of hardware that involves active power management. This includes design decisions involved in allocating power budget, partitioning the design into power domains, and defining the power management architecture, the power control logic, and the power control software, as well as products used for verification and implementation of such designs. This relates to the entire range of Mentor Graphics products that are involved in the design, verification, and implementation of low-power IP, chips, and systems.

**Figure 1-1. Typical Location of Power Aware Simulation in Design Flow**



## Documentation—Scope and Organization

Power Aware simulation augments normal HDL simulation capabilities with the ability to specify, model, and simulate the effects of active power management logic that is added to the design during implementation.

The success of applying Power Aware features depends on understanding the structure of your design and having the ability to run the Questa SIM simulator, plus—more generally—your goals of using simulation and verification software products.

The purpose of this manual is to provide basic usage and reference information on how to run a Power Aware simulation in the Questa SIM simulator. The primary focus is on how to define

the power intent of your design and then apply your conventional Questa SIM simulation flow to verify power-gating behavior.

Note that there are some areas related to Power Aware operation that this manual is not intended to cover:

- UPF standards — The [UPF Commands and Reference](#) appendix lists the UPF commands and options that are currently supported for version 1.0, 2.0, 2.1 and 3.0 of the UPF standard. However, this manual does not duplicate the complete usage information from those standards.
- Basic and advanced Questa SIM usage — Please refer to the other manuals of the Questa SIM simulator for information on operations related to Power Aware simulation, such as: command usage, the graphical user interface (GUI), design optimization, waveform analysis, and finite state machines.
- Power design — Reporting power estimation or creating an RTL architecture for optimized power implementation.

Specifically, the scope of this manual falls into the following broad areas:

## Usage

- Terminology definitions.
- Basic operating instructions for Power Aware simulation.
- Questa SIM commands specific to Power Aware.
- Differences between RTL and GLS.
- UPF commands for the power specification file.
- Reporting of results.

## Design

- General discussion of low-power design and Power Aware simulation.
- HDL models used for Power Aware simulation.

## Flow

- General discussion of low-power analysis as part of overall design flow.
- The distinctions between RTL and GLS, and the advantages of each at different points in the flow.

# How to Use This Manual

Use the following suggestions to learn more about Power Aware simulation.

If you are just beginning to learn about Power Aware simulation, read the following chapter:

- [Concepts for Using Power Aware Simulation](#)

If you are looking for an overview of Power Aware simulation, concepts and capabilities and how they are used, read the following chapter:

- [Power Aware Simulation](#)

If you are looking for an overview of the information provided by the Questa SIM simulator during Power Aware simulation, to help you understand how active power management works in your design, identify problem areas, and track what is verified, read the following chapters:

- [Power Aware Reports](#)
- [Power Aware Checks](#)
- [Graphical Display of Power Aware Operations and Results](#)
- [UPF Commands and Reference](#)

For more information about specific topics such as UPF and its use, Power Aware simulation commands and flows, report formats, messages, read the appropriate appendices of this manual.

# Chapter 2

## Concepts for Using Power Aware Simulation

---

This chapter provides a brief description of basic usage elements for running Power Aware simulation.

<b>Power Aware Models.....</b>	<b>22</b>
Power Aware Model Overview .....	22
Corruption Models .....	23
Isolation Models .....	24
Retention Models .....	25
Bias Models .....	28

## Power Aware Models

---

Conceptual information about Power Aware models.

<b>Power Aware Model Overview</b> .....	<b>22</b>
<b>Corruption Models</b> .....	<b>23</b>
<b>Isolation Models</b> .....	<b>24</b>
<b>Retention Models</b> .....	<b>25</b>
<b>Bias Models</b> .....	<b>28</b>

## Power Aware Model Overview

Verilog and VHDL both make the fundamental assumption that all logic is powered on at the beginning of simulation and remains powered on throughout simulation.

However, this assumption is no longer true for most complex devices being designed today. In these systems, power is typically provided to a given portion of a chip only when that part needs to function. This involves the inclusion of additional hardware to control power, the saving of a state when the power is turned off, and the mediation of interactions between portions of the system that are powered differently.

Power Aware simulation makes it possible to model these power management aspects of a system in simulation, even before the power management features have been implemented in the design. To do so, additional logic is included in the simulation model. This additional logic does the following:

- Defines the power management architecture to be imposed on the design.
- Implements the behavior of power management elements.
- Adapts the behavior of the design itself to reflect changes in power.

To run Power Aware simulation, the normal build process for constructing the simulation model is modified so that this additional logic can be added.

Power Aware simulation involves adding appropriate power intent to the design under test. For an RTL design with no power management content, this may involve inferring power domains, retention cells, isolation cells, level shifters, and the power supply network from the UPF power intent specification. For a gate-level design, created by a synthesis tool, that reads and implements UPF, some of the power management content may already be present, and therefore less of the power management content may need to be inferred from the UPF specification.

Application of power intent to the design includes addition of functionality to model the corruption of signal values when insufficient power is provided for normal operation. It also includes addition of functionality to model the saving and restoring of system state that is

performed by state retention elements in the power managed system. Addition of this functionality is done automatically as part of the preparation for Power Aware simulation.

**Tip** It is also possible for you to construct custom models for these behaviors. Refer to Appendix B, [Model Construction for Power Aware Simulation](#) for more information on Power Aware modeling.

---

## Corruption Models

Corruption refers to the situation where the value of a signal becomes unpredictable when the power supply for the element driving that signal is disconnected, changes to OFF, or drops below some threshold.

Corruption of a signal is represented by assigning a particular value to the signal. The corruption value depends upon the type of the signal and is user-definable. Corruption is typically applied to the signal drivers and propagates to all sinks of the signal that are not isolated from their source.

When a design instance is turned off, every sequential element within the powered-down instance and every signal driven from within the powered-down instance is corrupted. As long as the power remains off, no additional activity takes place within the powered down instance—all processes within the powered down instance become inactive, regardless of their original sensitivity list. Events that were scheduled before the power was turned off and whose target is inside a powered down instance have no effect.

When a design instance is turned on (restored), corruption of sequential elements and signals within the powered down element ends.

Continuous assignments once again become sensitive to changes to their right-hand side expressions, and other combinational processes (such as an always\_comb block in SystemVerilog) resume their normal sensitivity list operation. All continuous assignments and other combinational processes are evaluated at power up to ensure that constant values and current input values are properly propagated. Sequential elements are re-evaluated on the next clock cycle after power up.

### Corruption Values

Signals are corrupted by assigning them their default initial value (such as X for 4-state types). Default corruption values for Verilog and SystemVerilog are:

- 4-state logic types: ‘X
- 2-state logic types: ‘0
- SystemVerilog user-defined types: SystemVerilog default value

Default corruption values for VHDL are:

- Logic types: ‘X’
- Real types: ‘left
- For any type T: T`LEFT

### Default Corruption Semantics

During Power Aware simulation, if the driver of a net is powered down, then the output of the driver is corrupted, and this corrupted value propagates to all sinks of that net.

To understand how corruption occurs in a given design, it is necessary to recognize the elements of the design that represent or contain drivers.

- In an RTL code, any statements involving arithmetic or logical operations or conditional execution are interpreted as representing drivers and cause corruption when powered down. Unconditional assignments and Verilog **buf** primitives do not represent drivers and therefore do not cause corruption when powered down, but they may propagate corrupted signals from upstream drivers.

By default, the Questa SIM simulator does not corrupt signals that are driven by constants. For example:

```
assign out = 1'b1
```

---

#### Note

 Use the vopt –pa\_disable=constasfeedthru command to enable corruption of signals that are driven by constants.

---

- In a Gate-Level code, all cell instances are interpreted as containing drivers. As a result, buffer cell instances in a gate-level netlist causes corruption when powered down. Refer to “[Gate-level Cell Detection](#)” for information on the detection of any particular model as a gate-level cell.

## Isolation Models

Isolation ensures correct logical and electrical interactions between a powered down domain that is the source of a signal and a powered up domain that is the receiver of that signal.

---

#### Note

 Isolation cells themselves must be powered up in order to function correctly. If the power supply for an isolation cell is turned off, the isolation output is corrupted regardless of the state of the source power domain.

---

Isolation ensures that the receiving power domain sees a stable, known logic value, while at the same time preventing so-called “crowbar current” due to an indeterminate source value partially enabling both transistors of a CMOS inverter at the same time.

A given power domain may be powered down while another domain is operating in normal mode. Isolation ensures the following:

- Powered-down regions do not drive unknown values into the rest of the design (isolation on outputs).
- The rest of the design receives values that are functionally correct (isolation on inputs).

## Retention Models

Retention ensures saving the value of a design element in a power domain prior to switching off the power to that element, then restoring that value after power to the element is switched back on.

The set\_retention (and in UPF 1.0 set\_retention\_control) UPF command determines which registers in a power domain need to be retention registers and set the corresponding save and restore signals for the retention registers.

In UPF, you specify a retention strategy where state preservation is required, such as a latch, flip-flop, or retention memory.

The general sequence for specifying retention in UPF is:

1. Define your power domains:

```
create_power_domain
```

2. Specify the retention strategy—a set of registers in the domain requiring retention:

```
set_retention
```

3. Specify the retention control signals for the strategy:

```
set_retention (in UPF 2.0 and newer)  
set_retention_control (in UPF 1.0)
```

In simulation, two additional processes are added for each register that is to be retained, to model the retention behavior:

- One is sensitive to the save\_signal in accordance to the save sense.
- The second is sensitive to the restore\_signal in accordance to the restore sense.

A retention memory is also created for each sequential element that needs to be retained.

## Edge-sensitive and Level-sensitive Control of Retention Models

Power Aware simulation provides default Verilog models for retention cells that support both edge-sensitive and level-sensitive detection of input control signals for save and restore functions. Note that there are separate models for single and dual control signals:

- Single control signal — uses opposite (inverted) edge or level of one input signal to initiate save and restore.
- Dual control signals — uses edge or level of two different input signals to initiate save and restore.

## Automatic Model Selection

Based on the retention control signals specified in the UPF specification, Power Aware simulation automatically selects an edge-sensitive or level-sensitive model for retention. Automatic selection occurs according to the following conditions:

- If both save\_signal and restore\_signal are level-sensitive and the same signal is used for save and restore, then the single control, level-sensitive model is selected.
- If both save\_signal and restore\_signal are level-sensitive and two different signals are used for save and restore, then the dual control, level-sensitive model is selected.
- If any control signal (save\_signal or restore\_signal) is edge-sensitive and the same signal is used for save and restore, then the single control, edge-sensitive model is selected.
- If either save\_signal or restore\_signal is edge-sensitive and two different signals are used for save and restore, then the dual control, edge-sensitive model is selected.

## Custom Model Requirements

You can use custom retention models that meet the requirements of zero pin retention. The zero pin retention modeling requires that the clock, set, and reset be parked low when retention is enabled and disabled. If such is not the case, then both the flip-flop output and the retained value is corrupted.

To enable custom retention model with balloon-latch retention configuration, use the following argument with the vopt command:

```
vopt -pa_enable=customblret
```

To enable custom retention model with master-slave retention configuration, use the following argument with the vopt command:

```
vopt -pa_enable=custommmsret
```

## Level-sensitive Retention Model Protocol Example

For the following UPF retention command with level-sensitive controls, the single control, level-sensitive model is selected:

```
set_retention -domain PD1 \
  -save_signal {save_restore high} \
  -restore_signal {save_restore low}
```

The level-sensitive model accurately duplicates the behavior of a level-sensitive Liberty cell. Based on the save\_signal/restore\_signal level, the retention register switches between normal and retention operations as follows:

- When save\_signal is active, normal register behavior occurs—a balloon latch keeps latching the output of the register.
- When save\_signal is de-asserted, a balloon latch on the register output saves the register value.
- When restore\_signal is active, the saved value is retained.
- When restore\_signal is de-asserted, the normal operation of the register resumes.

The save or restore events are defined as trailing edge of the level-sensitive event. So, for this command, the register output is saved when save\_restore goes from high to low (the save event), and the retained value is transferred to the register output at the low to high (the restore event) transition of the save\_restore signal.

The following is an example of the protocol followed by level-sensitive model:

- In the save phase, normal register operation happens: D -> Q at clock edges. At the save event (defined above), the register output gets latched.
- In the restore phase, D has no effect on Q, so Q gets the retained value. At the restore event, normal operation resumes and Q gets new value of D from next active edge of clock.
- The register output gets corrupted when the primary power or retention power goes off.
- On primary power-up (and also if retention power is on), retention behavior or normal behavior of the register resumes.
- Retention power off corrupts the register output and the retained value, regardless of the primary power.
- For dual control signals, the retained value and register value both get corrupted when save and restore signals are simultaneously active.

## Master-slave (slave-alive) Retention Protocol

The following list identifies the protocol:

- When retention\_condition is enabled the register goes in retain mode where register gets the retained value and the effect of clock, set and reset is disabled.
- Register output is corrupted at power down.
- At power up the register goes in retain mode if retention\_condition is active else the normal operation resumes.
- If retention\_condition goes down during power down then the retained value is corrupted.

## Bias Models

Power Aware simulation provides support for bias modes that enables state retention in a different manner. In the power intent specification, you can specify bias power states of the primary supply of a power domain to simulate bias functionality.

When you use a bias mode, the power domain is powered on but runs with reduced functionality. Depending upon the particular bias mode involved, activity inside the domain may corrupt the contents of the domain.

Implementing a bias mode performs retention behavior without inserting explicit retention registers. This saves in area usage and also helps to reduce the leakage power. However, the electrical characteristics of the domain during this period prohibit normal logic functioning and thus timing requirements may not be met.

# Chapter 3

## Power Aware Simulation

---

This chapter describes the steps to run a Power Aware simulation.

<b>Required Inputs .....</b>	<b>30</b>
<b>Power Aware Simulation Commands .....</b>	<b>30</b>
<b>Power Aware Simulation Flows .....</b>	<b>34</b>
Using the Standard Power Aware Simulation Flow .....	35
Using the Delayed Optimization Flow .....	37
PDU-Based Simulation .....	38
Power Aware in Interactive Mode .....	40
<b>Macro Models Concepts .....</b>	<b>46</b>
Non-Power Aware HDL Models .....	46
Power Aware HDL Models .....	46
Extended Power Aware HDL Models .....	47
Liberty Models of Hard Macros .....	48
<b>Liberty Library Models.....</b>	<b>52</b>
Liberty File Concepts .....	52
Using Liberty Source Files with your Power Aware Simulation .....	53
Liberty Attribute Library Management .....	54
<b>Checker Modules and the bind_check Command .....</b>	<b>56</b>
<b>Power Aware Simulation Debug .....</b>	<b>60</b>
<b>Power Intent Application to Mixed RTL and Gate-Level Designs .....</b>	<b>63</b>
Hard Macro Cell Detection.....	63
Gate-level Cell Detection .....	68
Power Management Cell Detection .....	69
Automatic Insertion of Power Management Cells .....	70
Automatic Corruption and Retention of UDPs.....	71
<b>UPF Supply Connections.....</b>	<b>73</b>
Implicit Connections.....	73
Explicit Connections.....	73
Automatic Connections.....	75
Simulation Semantics for UPF Supply Connections .....	78
<b>Value Conversion Tables.....</b>	<b>79</b>
Using VCT Commands.....	79
Connections Using Value Conversion Tables (VCTs).....	81
Questa-specific Value Conversion Tables .....	83
<b>Defining Isolation.....</b>	<b>83</b>

<b>Defining Retention</b>	<b>87</b>
-retention_supply_set	87
-no_retention	88
-use_retention_as_primary	90

## Required Inputs

Power Aware simulation requires a complete HDL representation of the design, including simulation models for leaf-level instances.

- You can express the HDL representation in Verilog, SystemVerilog, or VHDL code, or any combination of these languages, and it may be synthesizable RTL code, behavioral RTL code, gate-level netlist, or any combination of these forms.
- For designs that are, or that include, a gate-level netlist, and for designs that include instances of hard macros, a Liberty library may be available that specifies the characteristics of the liberty or low power cells (such as isolation, level-shifter, and retention cell), and/or hard macros used in the design. Liberty models typically include power-related information that must be considered during Power Aware simulation. Power Aware simulation reads Liberty libraries to collect this information and takes it into account in building and executing a Power Aware simulation model.
- Power Aware simulation also requires a specification of power intent for the design to be simulated. Power intent is specified using the IEEE 1801 Unified Power Format (UPF). Power Aware simulation supports UPF 1.0, UPF 2.0, UPF 2.1 and UPF 3.0.

## Power Aware Simulation Commands

You invoke Power Aware simulation with the same commands used for conventional Questa SIM simulation, although the optimization (vopt) and simulation (vsim) commands have additional arguments specific to Power Aware.

- **vlog** or **vcom** — Compiles Verilog, SystemVerilog, or VHDL source code. For Power Aware simulation, these commands are used in the same manner as your standard simulation.
- **vopt** — Controls optimization in the HDL design. For Power Aware simulation, the vopt command also processes the UPF power intent specification.
- **vsim** — Runs the simulation of the HDL design. For Power Aware simulation, the vsim command also applies Power Aware simulation semantics to the HDL design.

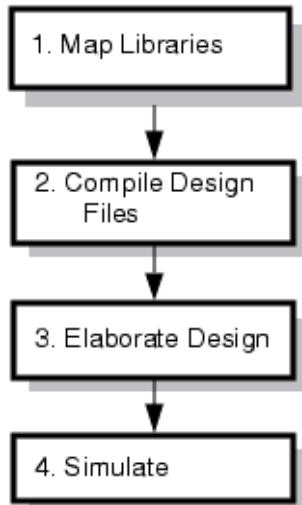
---

**Tip**

For a complete list of the Power Aware arguments, refer to “[Questa SIM Commands and Arguments](#)”.

---

## General Steps for Running Power Aware



1. Map your libraries:

```
vlib <library_name> vmap work <library_name>
```

2. Compile your VHDL or Verilog design files (ignore statements within translate\_off/on and synthesis\_off/on pragmas):

```
vcom <design_files>
vlog <design_files>
```

3. Elaborate your top-level design and apply the UPF power intent to the design:

```
vopt <design_top> -pa_upf <upf_file> -o <optimized_output>
```

There are multiple methods of using the vopt command to prepare and optimize a Power Aware simulation. Refer to [Table A-1](#) for a list of vopt arguments you can use for Power Aware.

4. Simulate the Power Aware version of your design.

```
vsim -pa <optimized_output>
```

## One-step Power Aware Simulation Using qverilog

You can use the [qverilog](#) command to run your Verilog-based or SystemVerilog-based Power Aware simulation. This command performs compilation, optimization, and simulation in a single command. For example:

```
qverilog reg_v1.v tb.v \
-R -voptargs="-pa_upf test.upf" -pa
```

Any arguments specific to vlog and vopt command must come before the -R group of arguments. You specify vsim arguments in the -R group.

## Control of X-State Propagation Using `vopt -xprop`

---

### Restriction

---

 Not available for ModelSim SE (Questa SIM only).

---

Power Aware simulation generates numerous X values as a result of the CORRUPTION simstate. This introduces more chances to miss design issues by being too optimistic due to RTL x-optimism. To help catch design issues related to X values earlier, you can run the `vopt -xprop` command as part of [Using the Standard Power Aware Simulation Flow](#).

---

### Note

---

 Refer to the following for more information on X propagation and the `vopt` command:

- [X Propagation in Simulation](#) in the *User's Manual*.
  - The [`vopt -xprop`](#) command in the *Command Reference Manual*.
- 

### Example

Consider the following asynchronous flip-flop, whose reset signal (`rst_n_s`) comes from a powered down domain and therefore its value is X.

```
always @(posedge clk_s or negedge rst_n_s)
begin: ff_verilog_model
  if(rst_n_s == 1'b0)
    q_s <= 1'b0;
  else
    q_s <= d_s;
end
```

Here, if `rst_n_s` is X, then the flip-flop behaves normally (`q_s <= d_s`), and Power Aware simulation alone does not catch this issue immediately.

However, a Power Aware simulation using `vopt -xprop` command is able to catch the issue immediately. This situation often takes place at power-up when some of the important design signals do not come up to their proper values. Debugging such issues using normal simulation is difficult and time-consuming.

### Using `vopt -xprop` with Power Aware

- If a power domain goes into the CORRUPT simstate, then the `vopt -xprop` command automatically gets disabled for that domain. The command remains disabled until the domain is in the CORRUPT simstate. The command is disabled because during the CORRUPT simstate, all corrupted signals in that domain become X, which activates numerous X-propagation assertions that are essentially noise (false positives) and not a design issue. When the design comes out of the CORRUPT simstate, the X-propagation assertions automatically gets enabled.
- If a power domain goes into any simstate besides CORRUPT, then X-propagation assertions remain enabled during that simstate.

- X-propagation assertions are also applied on UPF control signals. As a result, any X value on them is readily caught.

An example of using the entire vopt command, including the -xprop argument:

```
vopt -o opt1 tb -pa_upf tb.upf -xprop
```

Note that the -xprop argument also provides fine control values (which are described under the vopt command in the *Command Reference Manual*):

```
xprop [,mode=<pass|resolve|xtrap|none> [,object=<objectSelection>]]
```

# Power Aware Simulation Flows

Power Aware simulation follows the standard simulation flows of the Questa SIM simulator.

- [Using the Standard Power Aware Simulation Flow](#) — The standard (three-step) flow, where you explicitly invoke an optimization step.
- [Using the Delayed Optimization Flow](#) — The delayed optimization (two-step) flow, where optimization is invoked implicitly before simulation.

The optimization step used in these flows, whether invoked explicitly or implicitly, provides the following advantages for Power Aware simulation:

- Significantly better performance for Power Aware simulation of RTL designs.
- Support for Power Aware simulation of Gate-Level and mixed RTL/Gate-Level designs.
- More robust support for isolation of arbitrary SystemVerilog data types.
- Display of isolation and level-shifter cells in dataflow and schematic windows.
- Power management objects in structure, waveform, and object windows.
- Support for coverage data collection for power states and transitions and Power Aware dynamic checks.
- Support for user-defined power aware assertions using the UPF `bind_checker` command.
- Automatic Power Aware test plan generation.

Power Aware simulation involves register/latch detection in order to identify state elements in the design that need to be corrupted and may need to have their state retained during power down. However, your design may contain code that is not at the appropriate level of abstraction for register/latch detection, so you may need to exclude such code from power intent processing. Also, you may want to exclude parts of the design from Power Aware simulation for other reasons (see “[Argument to Exclude Design Element](#)” on page 297 in Appendix A).

The Questa SIM simulator provides additional vopt arguments for Power Aware simulation that you can use when generating the optimized design. For more information refer to “[Questa SIM Commands and Arguments](#)”.

---

## Note

 A simulation model created for Power Aware simulation contains specific Power Aware simulation artifacts—this model cannot be used for normal simulation.

---

<b>Using the Standard Power Aware Simulation Flow</b> .....	<b>35</b>
<b>Using the Delayed Optimization Flow</b> .....	<b>37</b>
<b>PDU-Based Simulation</b> .....	<b>38</b>
<b>Power Aware in Interactive Mode</b> .....	<b>40</b>

# Using the Standard Power Aware Simulation Flow

The standard Power Aware simulation flow is based on the three-step command flow. Use this flow to perform Power Aware simulation for RTL, Gate-Level, or mixed RTL/Gate-Level designs.

## Prerequisites

- Create your power specification file in the UPF format. You can write the UPF power intent specification in one of the two ways: either with respect to the top of the design-under-test (DUT), or with respect to the top of the testbench (TB). Refer to “[UPF Commands and Reference](#)” for more information.

## Procedure

### 1. Compile

Compile your design by running either the [vcom](#) (for VHDL) or the [vlog](#) (for Verilog) command as you do for any VHDL or Verilog/SystemVerilog design. Refer to either “[Compiling a VHDL Design—the vcom Command](#)” or “[Invoking the Verilog Compiler](#)” in the *Command Reference Manual* for more information.

### 2. Optimize

Use the [vopt](#) command with the following arguments:

- [-pa\\_top](#) — (optional) Specifies the hierarchical name of the design top instance. If unspecified, the default is the topmost instance of the design hierarchy. If the UPF specification is written with respect to the DUT top, then you must use this argument to specify the path from the top module down to and including the DUT instance.
- [-pa\\_upf](#) — Specifies the name of the UPF file containing the power intent specification. Power Aware simulation reads the UPF file and generates information required to run Power Aware simulation.
- [-pa\\_lib](#) — (optional) Specifies a destination library in which the power aware information is stored. Note that you must use the [vlib](#) command to create the library, then use the library name as the value for this argument. If unspecified, the default is the work library.
- [-o](#) — Specifies the name for the resultant optimized design.

Note that you can also specify any other [vopt](#) arguments used for conventional optimization to create the optimized design. This sequence is similar to the conventional [three-step optimization flow](#), described in the *User’s Manual*.

### 3. Simulate

After you compile and optimize the design, run the [vsim](#) command on the optimized design using the [-pa](#) argument to perform Power Aware simulation.

- **-pa** — Enables Power Aware simulation, including features for gate-level simulation.
- **-pa\_lib** — (optional) Instructs the Questa SIM simulator to load Power Aware information from the specified library. If unspecified, defaults to the current directory.

## Examples

### Compilation Examples

```
vcom <files>
vlog <files>
```

### Optimization Example — UPF written with respect to the top of the DUT

If the UPF specification is written with respect to the top of the design-under-test (DUT), and this module is instantiated in the testbench top module TB as TB.dut, then you need to invoke vopt as follows:

```
vopt TB -pa_top TB/dut -pa_lib work -pa_upf DUT.upf -o SimModel
[other vopt args]
```

- **TB** — Name of the top-level module of the test bench.
- **-pa\_top TB/dut** — Path from the test bench top, down to the design top instance.
- **-pa\_upf DUT.upf** — Name of the UPF file written with respect to the DUT top-level module.
- **-pa\_lib work** — Name of the destination library.
- **-o SimModel** — Name of the optimized output for your vsim run.
- **[other args]** — Any other vopt arguments used to control optimization.

### Optimization Example — UPF written with respect to the top of the test bench

If the UPF specification is written with respect to the top of the testbench (TB), then you need to invoke vopt as follows:

```
vopt TB_top -pa_upf TB.upf -pa_lib work -o SimModel [other vopt args]
```

- **TB\_top** — Name of the top-level module of the test bench.
- **-pa\_upf TB.upf** — Name of the UPF file written with respect to TB\_top.
- **-pa\_lib work** — Name of the destination library.
- **-o SimModel** — Name of the simulation-ready output file to be generated.
- **[other vopt args]** — Any other vopt arguments used to control optimization.

### Simulation Example

```
vsim SimModel -pa -pa_lib work [other vsim args]
```

- SimModel — Name of the simulation-ready output file generated by vopt.
- -pa — Invokes simulation in Power Aware mode.
- -pa\_lib work — Loads power aware information from library work.
- [other vsim args] — Any other vsim arguments used to control simulation.

## Related Topics

[Compiling a VHDL Design—the vcom Command](#) [Questa SIM User's Manual]

[Invoking the Verilog Compiler](#) [Questa SIM User's Manual]

# Using the Delayed Optimization Flow

Use the two-step, delayed optimization command flow to perform Power Aware simulation for RTL, Gate-Level, or mixed RTL/Gate-Level designs.

In conventional simulation, you can perform what is referred to as a [two-step optimization flow](#). In this flow, you implicitly call the vopt command from the vsim command line using the -voptargs argument.

---

### Note

 This flow may be useful in certain cases, such as automated scripts that assume only two steps are involved in building and running a simulation.

---

## Prerequisites

- Create your power specification file in UPF format. You can write the UPF power intent specification in one of two ways: either with respect to the top of the design-under-test (DUT), or with respect to the top of the testbench (TB). Refer to the chapter “[UPF Commands and Reference](#)” for more information.

## Procedure

1. Compile

Compile your design by running either the [vcom](#) (for VHDL) or the [vlog](#) (for Verilog) command as you do for any VHDL or Verilog/SystemVerilog design. Refer to either “[Compiling a VHDL Design—the vcom Command](#)” or “[Invoking the Verilog Compiler](#)” in the *Command Reference Manual* for more information.

2. Simulate

To implement simulation for this flow, run the vsim command on the test bench using the -pa argument, along with the -voptargs argument (specifying a UPF file) to invoke Power Aware simulation. When you invoke vsim on the test bench top module, it implicitly performs UPF processing and optimizes the design before beginning the Power Aware simulation.

## Examples

```
vsim TB_top -pa
  -voptargs="-pa_upf test.upf -pa_lib work [other vopt args]"
  [other vsim args]
```

- TB\_top — Name of the top-level module of the test bench.
- -pa — Invokes simulation in Power Aware mode.
- -voptargs — Instructs the Questa SIM simulator to apply arguments for the vopt command. For this flow, specify the following vopt arguments:
  - -pa\_upf <filename> — Causes the vopt command to apply the power intent specification.
  - -pa\_lib work — Name of the destination library.
  - [other vopt args] — Any other vopt arguments used to control UPF processing or optimization.
- [other vsim args] — Any other vsim arguments used to control simulation.

## PDU-Based Simulation

In conventional simulation, it is often desirable to optimize the design-under-test (DUT) separately from the test bench, so that you can use the same optimized DUT model (pre-optimized design unit, or PDU) with multiple test benches. You can also apply this approach to run Power Aware simulation, provided that the DUT appears at the same location in each test bench.

To use this approach, include the -pa\_defertop argument on the vopt command (in the three-step, standard flow) or in the -voptargs argument to vsim (in the two-step, delayed optimization flow). For example,

```
vopt -o <optimized_design_name> <top_level_module> -pa_defertop
  -pa_upf <upf_file_name> [other vopt args]


- -o <optimized_design_name> — Specifies the name of the pre-optimized design unit (PDU).
- <top_level_module> — Specifies the name of the top-level module of the design under test.
- -pa_defertop — Instructs the command to create a pre-optimized design.
- -pa_upf <upf_file_name> — Specifies the name of the UPF file written with respect to the DUT top-level module.
- [other vopt args] — Any other vopt arguments.

```

Then in simulation, run the vsim command, for example:

```
vsim -pa <testbench> [other vsim args]
```

- -pa — Invokes simulation in Power Aware mode.
- <testbench> — Specifies the name of the top-level module of the test bench, which has instantiations of the DUT.
- [other vsim args] — Any other vsim arguments used to control simulation.

You can optimize multiple DUTs separately to create multiple PDUs and link them all with a testbench in subsequent vsim executions, for example:

```
vopt -o opt1 DUT1 -pa_defertop -pa_upf dut1.upf
vopt -o opt2 DUT2 -pa_defertop -pa_upf dut2.upf
```

```
vsim TB -pa ...
```

where the testbench TB contains instantiations of DUT1 and DUT2.

For example, follow the instructions in the README found in the *<install\_dir>/examples/pa\_sim/example\_two/* directory.

## Power Aware Simulation with Multiple PDUs

Before running a Power Aware simulation, you can use multiple vopt commands to designate different pre-optimized DUTs for use with a given test bench. To do this, you need to specify the -pa\_defertop argument with each vopt command that specifies a separate DUT.

For example:

```
vopt dut1 -o opt1 -pa_upf test1.upf -pa_defertop
vopt dut2 -o opt2 -pa_upf test2.upf -pa_defertop

vopt tb -o tb_opt
...
-- Found PDU module dut1(opt1)
-- Found PDU module dut2(opt2)
```

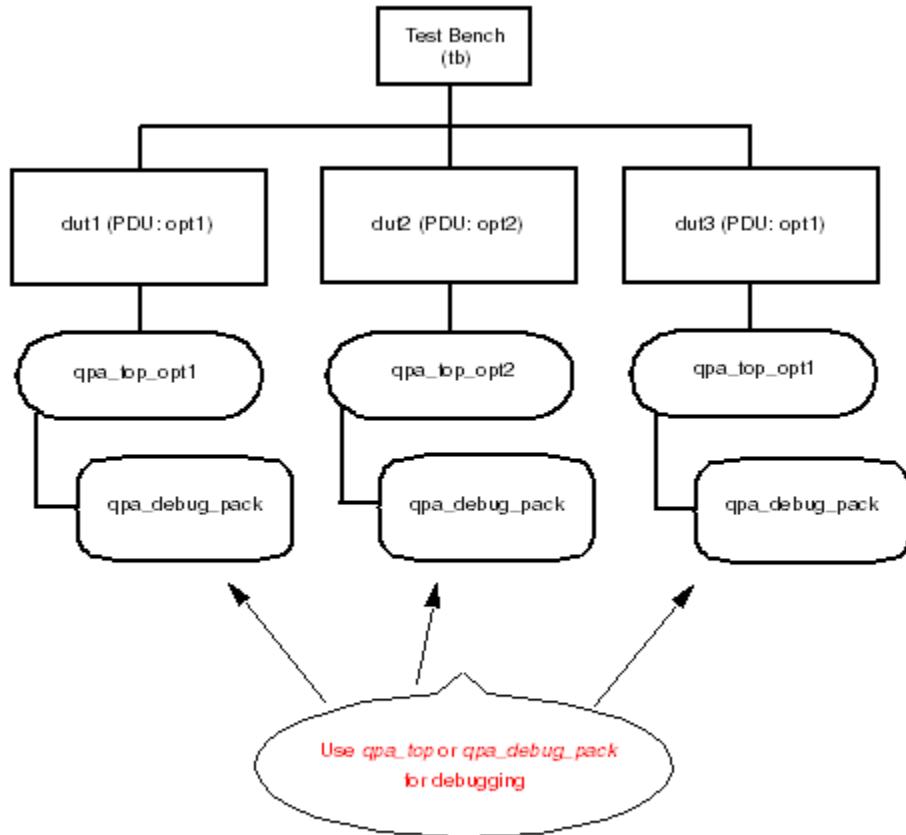
You can then use the vsim command to run the simulation on either the original test bench (tb) or the optimized test bench (tb\_opt). The Questa SIM simulator recognizes each DUT and applies the power intent defined in its UPF file (test1.upf, test2.upf).

For example:

```
vsim tb -pa -c
...
# loading dut1(PDU: opt1)
# loading dut2(PDU: opt2)
```

Figure 3-1 shows a block diagram of a test bench loading multiple DUTs as pre-optimized design units.

**Figure 3-1. Simulation Flow for Multiple PDUs**



#### Report generation

Following the simulation, you can use the pa report command with the -scope argument to generate a report for a specific DUT. For example:

```
pa report -scope /tb/dut1
```

## Power Aware in Interactive Mode

The -pa\_interactive argument of the vopt command changes the Questa SIM command line to a Power Aware session. This lets you run subsequent Questa SIM commands without reloading the design.

Power Aware simulation often requires running the vopt command that specifies the UPF file multiple times during development or debugging. Making minor changes to the UPF file and

rerunning vopt requires reloading the design to verify changes. In many cases, the time it takes to reload the design can be significantly long.

To remove this limitation, the Questa SIM simulator provides an interactive mode of using Power Aware, which enables you to load a design once and then apply the UPF specification multiple times. For successive runs, you can observe results immediately because you do not have to wait for loading.

## Interactive Mode on the Command Line

To activate interactive mode, you specify the `-pa_interactive` argument as part of the `vopt` command that you use to initiate a Power Aware simulation. The result of this is that the design is loaded once, as usual, but the command line prompt changes to the following when loading has finished:

**pa >**

This is a Tcl command prompt that indicates that the Power Aware session is in interactive mode. You can now run any Questa SIM command any number of times without reloading the design.

To exit this mode and return to the Questa SIM command line, enter either of the following:

**exit**  
**quit**

### Usage Notes

Running interactive Power Aware consists of the following actions:

- Add the `-pa_interactive` argument to any other arguments and files that you want to specify for the Power Aware `vopt` command. Refer to “[Using the Standard Power Aware Simulation Flow](#)” for more information on using `vopt` to initiate Power Aware.
- Run the `vopt` command and wait for the design to load for the Power Aware simulation.
- Note that the prompt on the command line changes to “`pa>`” after the design has loaded.
- At this new prompt, enter any Questa SIM command that you want to run on the design. Note that the `pa>` prompt reappears after the completion of the last command.
- Make any changes to the UPF file or your simulation configuration that you want.
- Rerun any Power Aware commands you want (including `vsim`), even with arguments or specifications that differ from previous commands you ran in this interactive session. The Questa SIM simulator will apply them without reloading the design.

---

#### **Note**

 You do not need to use the `-pa_interactive` argument again with other `vopt` commands you enter during this interactive session.

---

- Alternatively, you can specify a do file as part of the vopt -pa\_interactive command that contains a list of commands that will run at the pa> prompt. That is, once the design has loaded and the command line enters interactive mode, the Questa SIM simulator will run the commands in the do file sequentially (from top to bottom).

For example:

```
vopt tb -pa_interactive -do vopt_cmd.do
```

where the vopt\_cmd.do file contains the following commands for Power Aware checking:

```
vopt -pa_upf test1.upf -pa_staticchecksonly
cat report.static.txt
mv report.static.txt run1.static.txt
vopt -pa_upf test2.upf -pa_staticchecksonly
cat report.static.txt
mv report.static.txt run2.static.txt
vopt -pa_upf test3.upf -pa_checks=s -o t
cat report.static.txt
mv report.static.txt run3.static.txt
vsim t -pa -c -do "run -all"
```

- Terminate the interactive Power Aware session by entering “quit” or “exit.” This returns the Questa SIM command line to its previous mode.

## Examples

- Run a vopt command that includes the -pa\_interactive argument. It will load the complete design along with Liberty models, followed by the “pa” command prompt that awaits further input. At the prompt, you can enter multiple and repeated commands without reloading the design.

```
vopt tb -work lib1 -pa_interactive <other arguments>
```

```
Top level modules:
tb

Analyzing design...
-- Loading module tb
-- Loading module top_vl
```

```
pa>
```

- After you load the design with your initial vopt command using -pa\_interactive, you can enter subsequent vopt commands with new or different specifications without reloading the design. Here, the following vopt command applies different UPF definitions (in the test1.upf file), performs checking (-pa\_checks), and generates a report on design elements (-pa\_genrpt=de). Note the “pa” prompt before and after running the command.

```
pa> vopt -pa_upf src/test1.upf -o t -pa_genrpt=de -pa_checks
```

```
Analyzing design (analysis time 100 seconds)...
-- Processing UPF File: src/test1.upf
-- Resolving UPF Supply Network
-- Processing power-aware module tb(fast)
-- Processing power-aware module top_vl(fast)
#
# pa report
# Generated Power Aware Report : report.static.txt
# Generated Power Aware Report : report.de.txt

-- Invoking Power Hierarchy Compilation

Analyzing design (analysis time 104 seconds)...
Optimizing 44 design-units (inlining 22/70 module instances, 0/0 UDP
instances):
-- Inlining module top_vl(fast)
-- Inlining module top_vl(fast__1)
-- Optimizing module tb(fast)
-- Optimizing module bl_edge_model(fast)
-- Optimizing package mtiUPF.UPF(fast)
-- Optimizing package mspa_pack(fast)
Optimized design name is t
End time: 11:31:58 on Jul 29, 2015, Elapsed time: 0:01:45
Errors: 0, Warnings: 0
```

**pa>**

- After the previous vopt command has finished, you can enter another vopt command with different specifications. Here, the following vopt command applies different UPF definitions (test2.upf), performs checking (-pa\_checks), and generates a report on architecture and design elements (-pa\_genrpt=pa+de).

**pa> vopt -pa\_upf src/test2.upf -o t -pa\_genrpt=pa+de -pa\_checks**

```
Analyzing design (analysis time 201 seconds)...
-- Processing UPF File: src/test2.upf
-- Resolving UPF Supply Network
Incremental compilation check found no design-units have changed.
#
# pa report
# Generated Power Aware Report : report.static.txt
# Generated Power Aware Report : report.pa.txt
# Generated Power Aware Report : report.de.txt

-- Invoking Power Hierarchy Compilation

Analyzing design (analysis time 202 seconds)...
Incremental compilation check found no design-units have changed.
Optimized design name is t
End time: 11:33:36 on Jul 29, 2015, Elapsed time: 0:03:23
Errors: 0, Warnings: 0
```

**pa>**

- Once you have used one or more vopt commands to apply power intent definitions and instructions for checking and reporting, you can use the vsim command to run the Power Aware simulation.

```
pa> vsim t -pa -c

Reading pref.tcl

# DEV-main 2876996

# vsim t -pa -c
# Start time: 11:34:25 on Jul 29, 2015
# //
# // Copyright 1991-2015 Mentor Graphics Corporation
# // All Rights Reserved.
# //
# // THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION
# // WHICH IS THE PROPERTY OF MENTOR GRAPHICS CORPORATION OR ITS
# // LICENSORS AND IS SUBJECT TO LICENSE TERMS.
# // THIS DOCUMENT CONTAINS TRADE SECRETS AND COMMERCIAL OR FINANCIAL
# // INFORMATION THAT ARE PRIVILEGED, CONFIDENTIAL, AND EXEMPT FROM
# // DISCLOSURE UNDER THE FREEDOM OF INFORMATION ACT, 5 U.S.C. SECTION 552.
# // FURTHERMORE, THIS INFORMATION IS PROHIBITED FROM DISCLOSURE UNDER
# // THE TRADE SECRETS ACT, 18 U.S.C. SECTION 1905.
# //
# Loading sv_std.std
# Loading mtiUPF.UPF
# Loading work.mspa_pack(fast)
....
# Loading work.MSPA_T_TRIG(fast)
# Initializing Power Aware mode
VSIM 1> run -all
# ** Note: (vsim-8916) QPA_UPF_RET_CTRL_INFO: Time: 4 ns, Retention Strategy
# (pd_retention), Retention RESTORE (/tb/top_v1/clk), Retention Sense
# (negedge), switched to polarity (0). Power Domain: pd
# ** Note: (vsim-8916) QPA_UPF_RET_CTRL_INFO: Time: 8 ns, Retention Strategy
# (pd_retention), Retention RESTORE (/tb/top_v1/clk), Retention Sense
# (negedge), switched to polarity (0). Power Domain: pd
...
# ** Note: (vsim-8916) QPA_UPF_RET_CTRL_INFO: Time: 2778 ns, Retention
# Strategy (pd_retention), Retention RESTORE (/tb/top_v1/clk), Retention
# Sense (negedge), switched to polarity (0). Power Domain: pd
# ** Note: (vsim-8916) QPA_UPF_RET_CTRL_INFO: Time: 2782 ns, Retention
# Strategy (pd_retention), Retention RESTORE (/tb/top_v1/clk), Retention
# Sense (negedge), switched to polarity (0). Power Domain: pd
# ** Note: $stop : src/tb.v(173)
# Time: 2783 ns Iteration: 0 Instance: /tb
# Break at src/tb.v line 173
# Stopped at src/tb.v line 173
VSIM 2> quit
# End time: 11:35:48 on Jul 29, 2015, Elapsed time: 0:01:23
# Errors: 0, Warnings: 0
```

pa>

- Because interactive mode for Power Aware is a Tcl session, the command line responds to the “pa” prompt as any Tcl or shell command prompt. This means you can also enter Tcl or shell commands.

pa> cat report.de.txt

```
-----  
---- ModelSim Power Aware Design Element Report File ----  
-----  
-- ModelSim Version :  
-- Generated on      : Tue Aug  4 12:47:17 2015  
-----  
-- Format:  
--   Power Domain / Scope Mapping:  
--     <domain name>[(PD_ID)] : {<path #>} = scope <full path to element> [<>]  
--           specifies scope that is part of power domain <domain name>  
--           <> identifies a scope at the power domain boundary  
--           SIM identifies a scope as simulation only  
--           PD_ID : Power Domain ID assigned by the tool if another power domain  
--                   of same name is present in a different scope.  
--           The PD_ID information is present in power architecture report.  
--   Power Domain Element Identification:  
--     <domain name>[(PD_ID)] : {<path #>}/{<element name>} [<element type>]  
--           specifies design element in power domain <domain name>  
--           <element type> is one of  
--             NPM_LA      => Non Retention Latch  
--             NPM_FF      => Non-retention Flip-Flop  
--             MEM         => Non-retention Memory  
--             UDP_LA      => Non-retention UDP latch  
--             UDP_FF      => Non-retention UDP Flip-Flop  
--             R           => Retention element  
--             <none>      => Combinational logic  
-----  
pd: {Path2} = scope /tb/top_vl<>  
  
pd: {Path2}/q2 R  
pd: {Path2}/q3 NPM_FF  
pd: {Path2}/q1 NPM_FF  
pd: {Path2}/q_latchv1 NPM_FF  
pd: {Path2}/q_regv1 NPM_FF
```

**pa>**

- To terminate Power Aware interactive mode, enter exit or quit.

**pa> quit**

## Macro Models Concepts

---

You can process macro models in various ways, depending upon whether they are power aware or not, and if not, how you want to add power aware behavior to the model.

You can experiment with some example code at the location `<install_dir>/examples/pa_sim/example_three/` directory.

<b>Non-Power Aware HDL Models.....</b>	<b>46</b>
<b>Power Aware HDL Models.....</b>	<b>46</b>
<b>Extended Power Aware HDL Models .....</b>	<b>47</b>
<b>Liberty Models of Hard Macros.....</b>	<b>48</b>

## Non-Power Aware HDL Models

The simulation model for a hard macro is a normal functional model that does not include any power-related structures or power aware behavior. Such models are processed along with the design just like any other RTL or Gate-Level module.

Since a non-power aware HDL model has no supply ports, you cannot explicitly connect UPF supply nets to the model. Instead, each instance of the macro is implicitly connected to the primary supply of the domain in which it is instantiated, and the simulation of the instance is modified to reflect the simstate of that supply's current power state at any given time.

To use this mode of operation for a given hard macro, simply process the hard macro simulation model along with the rest of the design code. Do not provide a Liberty model for that hard macro, otherwise the Liberty model is interpreted to determine the power aware behavior of each instance of the macro.

## Power Aware HDL Models

Hard macros often have multiple supplies that provide power to different portions of the macro. In such cases, implicit connection to the primary supply of the containing domain is not sufficient.

To enable connection of multiple supplies, the HDL simulation model for a hard macro may include port declarations representing each of the power supplies required by the macro. The existence of such port declarations identifies the model as a power aware HDL simulation model.

Since a power aware HDL model has supply ports, you can explicitly connect UPF supply nets to the model. If a supply port has the **UPF\_pg\_type** attribute associated with it, either by an HDL attribute specification or a UPF **set\_port\_attributes** command, then a connection to that supply port can be made automatically based upon its **pg\_type**. In this case, the appropriate value conversion table (VCT) is also inserted based on the **pg\_type** of the port. Otherwise, a

connection to that port can be made explicitly, by referring to the name of the port in the **connect\_supply\_net** command. In this case, any VCT required must be specified explicitly as part of the **connect\_supply\_net** command.

A power aware HDL model must read the values of its supply ports and use those values to determine when its internal registers and its outputs should be corrupted to represent the effects of power shutoff or low voltage. In particular, it should assign appropriate values to those objects to indicate that the relevant power supplies are insufficient to support normal operation. If the hard macro contains embedded isolation or retention features, it should also monitor the relevant control signals and model the isolation and/or retention effects that result when those control signals are asserted.

To use this mode of operation for a given hard macro, first ensure that the simulation model for the hard macro is a power aware model. For each instance of the model, specify UPF **connect\_supply\_net** or **connect\_supply\_set** commands to connect the appropriate power supply to each supply port of the model, along with the appropriate VCT as required. Simstate-based corruption semantics are automatically disabled as a result of those connections.

## Extended Power Aware HDL Models

It is sometimes convenient to have simulation models for hard macros that can be used in either normal simulation or Power Aware simulation without modification, without adding extra levels of hierarchy, and without leaving ports unconnected. An extended Power Aware simulation model makes this possible.

In an extended Power Aware simulation model, supply ports are represented by internal wires or registers rather than by port declarations. Each internal object is assigned a constant value that represents the normal operational state of each supply. This constant value is always the value of the supply object in normal simulation, but in a Power Aware simulation, a UPF command can connect a supply net to the object, and the value of that supply net overrides the constant value during simulation. As in the case of a power aware model, both explicit connections and automatic connections based on the **UPF\_pg\_type** attribute can be used, and the model should be written in the same way that a power aware HDL model is written, except that it should refer to the internal supply objects rather than to supply ports.

To use this mode of operation for a given hard macro, follow the same procedure as described in the section [Power Aware HDL Models](#). In addition, include the following argument on the **vopt** command line to enable use of supply connections to internal objects:

**vopt -pa\_upfextensions=internalconn**

Refer to “[Supported UPF Extensions](#)” for more information.

## Liberty Models of Hard Macros

A Liberty model for a hard macro is an alternative way of providing power aware behavior for instances of that hard macro. The Liberty model can be read along with a non-power aware HDL simulation model. Power-related attributes in the Liberty model essentially add power aware behavior to the non-power aware HDL model.

In this mode of operation, both a non-power aware HDL simulation model and a Liberty model for the same hard macro are provided to Power Aware simulation. In this case, the Liberty model defines all the supplies of the macro as **pg\_pins**, including internal **pg\_pins** representing the output of embedded switches, bias generators, regulators, or LDOs. UPF supply nets or supply set functions can be connected to the **pg\_pins** of the Liberty model, either explicitly or automatically based upon their **pg\_type**. As with the other modes, VCTs are either specified explicitly or are also inserted automatically based on **pg\_type**.

Liberty attributes are used to determine when inputs and outputs of a hard macro instance should be corrupted. The Liberty attributes involved are as follows:

- **switch\_function, pg\_function** — These attributes of the cell define the enabling condition and the input supply of an embedded power switch. The output of the power switch is modeled as an internal **pg\_pin**. The internal **pg\_pin** may be referenced in the **related\_power\_pin**, **related\_ground\_pin**, or **power\_down\_function** attribute definitions.
- **related\_power\_pin, related\_ground\_pin** — These attributes of each logic pin define the power and ground supplies of the logic receiving an input or driving an output of the hard macro. These supplies are typically used to determine the corruption of inputs to the hard macro.
- **power\_down\_function** — This attribute of each output pin defines the condition under which the output should be corrupted. Its value is an expression that may refer to any input or internal supply.

To use this mode of operation for a given hard macro, follow the same procedure as described in the section “[Non-Power Aware HDL Models](#)”. In addition, include the following argument on the **vopt** command line to enable use of a corresponding Liberty model to provide the required Power Aware simulation behavior:

```
vopt -pa_enable=libertypamodel
```

which enables use of Liberty model **pg\_pin/pg\_type** definitions of supply ports and Liberty model attributes for corruption.

Refer to “[Usage of vopt -pa\\_enable and -pa\\_disable](#)” for more information.

It also enables many modeling consistency and application checks, including the following:

- Checks that automatic connections are not applied to internal pg pins (because the direction of internal pg pins affects their use, but direction is not considered in UPF automatic connections)
- Checks that any supply net connected to one or more internal pg pins is resolved appropriately

If you want to corrupt only the outputs of the hard macro, you can disable input corruption by including the following argument:

**vopt -pa\_noinpcorr**

which disables Liberty attribute-based input corruption but does not affect Liberty attribute-based output corruption.

## Liberty Models and Simstate Behavior

When you connect a supply net to a supply port of an instance the simulator disables simstate behavior only when the supply port is present in HDL. If the supply port is missing from HDL, but is present in the Liberty model, the simulator does not disable the simstate behavior for that instance.

Given this behavior, you do not need to specify `set_simstate_behavior` to TRUE if you want to corrupt a non-power aware HDL model based on liberty attributes, regardless of any automatic or explicit connections. For this scenario, the simulator ensures that only ports of the instance are corrupted using Liberty attributes and everything inside the instance is treated as always on.

## Connections to Liberty Model Internal Power/Ground Pins

When hard macros containing embedded power sources are involved in a design, those power sources are sometimes connected externally in the implementation flow. The Questa SIM simulator recognizes various typical configurations of such connections when Liberty models are used to define the power aware semantics of hard macros.

Liberty models representing hard macros may contain internal supply pins (pg pins) as well as external supply pins. Internal pg pins represent the output of a switch or voltage generator block such as a bias generator, regulator, or LDO embedded within the macro. These internal pg pins are typically used in different ways depending upon their directions.

- Internal pg pins with direction '**internal**' represent the output of a voltage generator block that is typically intended to supply power only to objects within the macro. Such internal pg pins cannot be connected to supply nets outside the macro.
- Internal pg pins with direction '**inout**' are similar to pg pins with direction 'internal', except that when there are multiple instances of the same switch (either embedded in cells or external to the cells) with the same input power supplies and control inputs, the

internal pg pin of each instance and any external switch outputs can be strapped together via connection to a common supply net to synchronize and distribute the loading of the switches embedded in each of the instances. In this case the common supply net must be resolved parallel.

- Internal pg pins with direction '**output**' represent the output of an embedded power switch or voltage generator block that is typically intended to export power from the macro. Such an internal pg pin can be connected to an external supply net that is used as a power source for elements outside the macro. The external supply net may be unresolved, resolved one-hot, or resolved parallel, depending upon the number of supply sources it has and whether it is exporting a single source, muxing multiple sources, or merging multiple sources.

## Synchronization of Internal Power/Ground Pin Sources

When an internal pg pin with direction '**inout**' defined by the Liberty model of a hard macro instance is connected to an external supply net using UPF commands, and the supply net is a resolved parallel supply net with multiple sources, an additional check is performed by the Questa SIM simulator to ensure that the sources of the resolved parallel supply net are all consistent. When this check is applied, a warning message is generated if the state of any source is UNDETERMINED, or the states of the sources are not all the same.

This check is applied whenever you specify the vopt argument:

```
vopt -pa_enable=libertypamodel
```

Refer to “[Usage of vopt -pa\\_enable and -pa\\_disable](#)” for more information.

The same check can also be applied to resolve parallel supply nets whose sources include internal pg pins with direction '**output**', by specifying the following vopt argument:

```
vopt -pa_pgoutsynchchk
```

## Consistency of Power Aware Models and Liberty Models

When both a Liberty model and a power aware HDL simulation model are available for the same module, it is possible to determine whether the two models define the same set of supply and logic pins. If there is a difference between the two, the Questa SIMsimulator issues an error message.

The Questa SIM simulator does not attempt to check whether the corruption behavior of a power aware simulation model is consistent with the corruption behavior indicated by the corresponding Liberty model. In any case, when both a Liberty model and a power aware HDL simulation model are available, the Questa SIM simulator defaults to using the HDL simulation model without any additional information from the Liberty model.

## Related Topics

[Usage of vopt -pa\\_enable and -pa\\_disable](#)

## Liberty Library Models

Liberty libraries define standard cells and macro cells that can be used in a system. Liberty models include information that defines how power supplies provided to instances of such cells are used to power logic receiving inputs or driving outputs of those cell instances. Liberty models also include other information pertaining to power management cells.

<b>Liberty File Concepts .....</b>	<b>52</b>
<b>Using Liberty Source Files with your Power Aware Simulation.....</b>	<b>53</b>
<b>Liberty Attribute Library Management .....</b>	<b>54</b>

## Liberty File Concepts

The information contained in a Liberty library may be of use in both RTL and Gate-Level simulations as well as in mixed RTL/GL simulations. For example, Liberty models of hard macros may be of use in applying power intent to the RTL portion of a design, and information about standard cells in a Liberty library may be of use in applying power intent to the gate-level portion of a design.

Information on Liberty files contains the following terms:

- Liberty Source File — this is a Liberty library file that contains one or more cells and typically has the file suffix.*.lib*.
- Liberty attribute library — this is a file output by the vopt command using the -pa\_dumplibertydb argument that combines the information from one or more Liberty source files.

Power Aware simulation uses the information present in the Liberty library to create a proper connection of the supplies and the control signals that are defined in the UPF power intent specification. Power Aware simulation also uses such information to assist in automatic recognition of power management cells that are present in a design.

## Liberty Attribute Libraries

You can preprocess Liberty libraries to extract power-related information into an attribute library to be used during UPF processing in preparation for Power Aware simulation. This preprocessing step enables the most efficient use of Liberty libraries when the same library is used many times. Refer to the section “[Liberty Attribute Library Management](#)” for more information.

Cell names, as defined in a Liberty source file, are unique within the attribute library, however multiple libraries can define the same cell name. When this occurs, the attribute library contain versions of each cell. If your design references the cell name with multiple instances, vopt picks one at random and issue a warning message about multiple cells of the same name.

# Using Liberty Source Files with your Power Aware Simulation

---

There are two ways to use Liberty files with your Power Aware simulation.

Specifying Liberty Cells via a Liberty Attribute Library .....	53
Specifying Liberty Source Files Directly .....	53

## Specifying Liberty Cells via a Liberty Attribute Library

After creating a Liberty attribute library you can access the attribute library during the processing of the UPF power intent.

Refer to “[Liberty Attribute Library Management](#)” for details.

### Procedure

Perform the following task:

- Use the -pa\_loadlibertydb argument with your standard Power Aware simulation vopt command.

### Examples

```
vopt design_top -pa_upf compile.upf -pa_lib work  
-pa_loadlibertydb=lib_datafile [other vopt args]
```

## Specifying Liberty Source Files Directly

Access the Liberty files directly during the processing of the UPF power intent.

### Procedure

Perform the following task.

- Use the -pa\_libertyfiles argument with your standard Power Aware simulation vopt command.

### Examples

The following command analyzes the *a.lib* and *b.lib* Liberty files in the order in which they are specified in the -pa\_libertyfiles argument, and creates an internal database for these libraries. This internal database is used for Liberty data in a Power Aware analysis; the internal database is deleted at the end of the vopt run.

```
vopt design_top -pa_upf compile.upf -pa_lib work  
-pa_libertyfiles=a.lib,b.lib
```

# Liberty Attribute Library Management

With the vopt command you can create and manipulate Liberty attribute libraries, which enable you to easily specify Liberty cells to use during the processing of the UPF power intent.

<b>Creating a Liberty Attribute Library .....</b>	<b>54</b>
<b>Updating a Liberty Attribute Library.....</b>	<b>54</b>
<b>Refreshing a Liberty Attribute Library .....</b>	<b>55</b>

## Creating a Liberty Attribute Library

Use the vopt command to parse the Liberty files and create a saved attribute library containing information extracted from the Liberty files that is used in Power Aware simulation.

### Procedure

Perform the following task.

- Use the -pa\_libertyfiles and -pa\_dumplibertydb arguments on the vopt command, where:
  - -pa\_libertyfiles — Specifies the Liberty files to read. You can specify multiple files by separating file names with a comma.
  - -pa\_dumplibertydb — Specifies the name of the Liberty attribute library to be saved for future use.

### Examples

```
vopt -pa_libertyfiles=a.lib,b.lib -pa_dumplibertydb=lib_datafile
```

Alternatively, you can add these values to an external file and use the -f argument with vopt. This can be useful if you need to specify a lot of Liberty files. For example, given the text file *liberty.txt*:

```
#liberty.txt
-pa_libertyfiles=a.lib,b.lib,c.lib,d.lib,e.lib,f.lib,g.lib,h.lib
```

You write your vopt command as:

```
vopt -f liberty.txt -pa_dumplibertydb=lib_datafile
```

## Updating a Liberty Attribute Library

Update an existing Liberty attribute library with new information about Liberty cells. This is most commonly used when a *.lib* file is edited to correct an error and needs to be reprocessed.

When updating a Liberty attribute library you can do the following:

- Update with a Liberty source file that was not used previously.  
The new cells from are added to those added during the initial creation or a previous update.
- Update with a Liberty source file you used before.  
The contents of the Liberty source file overwrites any cells created during the addition of the file during the initial creation or the previous update.

You can add a single cell to a Liberty attribute library by:

- creating a new Liberty source file containing that cell and then updating the Liberty attribute library
- adding the cell to a Liberty source file used during the initial creation or a previous update and then updating the Liberty attribute library.

You can remove a cell from a Liberty attribute library by removing the cell from the Liberty source file used during the creation of the Liberty attribute library and then updating the attribute library with this new Liberty source file.

## Procedure

Perform the following task.

- Use the -pa\_libertyupdate argument with the vopt command as if your were creating a new Liberty attribute library.

## Examples

Assuming that a Liberty attribute library is created already at location */home/user/libdbs/LVT*, for Liberty files *a.lib*, *b.lib*, the command:

```
vopt -pa_libertyfiles=a.lib,b.lib -pa_dumplibertydb=/home/user/libdbs/LVT  
-pa_libertyupdate
```

analyzes *a.lib* and *b.lib* files again and overwrites the previous attribute library with a new one.

## Refreshing a Liberty Attribute Library

If you created a Liberty attribute library with a previous version of the Questa SIM simulator, you can modify the attribute library to work with the recent version.

## Procedure

Perform the following task.

- Add the -pa\_libertyrefresh argument to your standard Power Aware simulation vopt command.

## Examples

This command refreshes the Liberty attribute library lib\_db to the current version of the Questa SIM simulator.

```
vopt design_top -pa_upf compile.upf -pa_lib work -pa_loadlibertydb=lib_db  
-pa_libertyrefresh [other vopt args]
```

# Checker Modules and the bind\_check Command

Use the UPF bind\_checker command to insert a checker module into a design without modifying the design code or introducing functional changes. This allows you to verify your design by writing custom assertions and implementing coverage for UPF strategy elements.

Using the bind\_checker command, you can insert these custom assertions and coverage models in the design. However, writing a separate checker command for every individual design element would make the list of bind checker commands unreasonably large. The Questa SIM simulator provides the capability to write bind checkers that are generic enough to cover all the elements of a particular strategy.

For example, the following retention strategy applies retention on every sequential logic element within power domain RS1.

```
set_retention RS1 -domain PD1 ...
```

If you need to put custom assertions on every such sequential logic element, then you would have to write many individual bind\_checker commands. The following sections describe how to use UPF generics and a query command to configure references to elements, which you can then specify in a bind\_checker command.

## UPF Generics

UPF provides support for symbolic references called generics, which allows creating a placeholder for a parameter value. You can use generics to write a bind\_checker command that will cover all elements of a given power strategy. The bind\_checker command provides an option named -parameters, which can accept the value defined for a generic.

The simulator supports the following generics, which you can use with the -parameter option of the bind\_checker command:

- UPF\_ELEMENT\_HIER\_PATH — Captures the name of the signal (such as RET\_Q).

- UPF\_ELEMENT\_MSB — Captures the most significant bit of the vector signal (defaults to 0, if msb not present).
- UPF\_ELEMENT\_LSB — Captures the least significant bit of the vector signal (defaults to 0, if lsb not present).
- UPF\_GENERIC\_CLOCK — Captures the clock of the retention signal.
- UPF\_GENERIC\_ASYNC\_LOAD — Captures the asynchronous controls of the retention signal.

The following example demonstrates how to create a generic bind\_checker command.

### **Example 3-1. bind\_checker Assertion for Retention**

Consider the following retention strategy, which establishes the power domain and associated signals for saving and restoring:

```
set_retention RET \
    -domain PD \
    -save_signal {SAVE posedge} \
    -restore_signal {RESTORE posedge} \
    -restore_condition {!UPF_GENERIC_CLOCK}
```

The following assertion module establishes the coverage for retention in the power domain, on which you can perform binding:

```
module checker_retention (ret_elem, restore_sig, clock) ;
    parameter ELEM_NAME;
    parameter ELEM_MSB;
    parameter ELEM_LSB;
    parameter RET_NAME;
    input [ELEM_MSB:ELEM_LSB]ret_elem;
    input restore_sig, clock;
    always@(posedge restore_sig)
        assert (clock != 1) else $error ("Incorrect restore protocol \
            for %s[%d:%d]", ELEM_NAME, ELEM_MSB, ELEM_LSB);
endmodule
```

Now, the following query\_retention command assigns generics to the parameters of the assertion module:

```
array set RET_DETAILS [query_retention RET -domain PD -detailed]
set RET_NAME           $RET_DETAILS(retention_name)
set RESTORE            $RET_DETAILS(restore_signal)
set RET_ELEMENTS       $RET_DETAILS(elements)
set RET_ELEM_NAME      $RET_DETAILS(upf_element_hier_path)
set RET_ELEM_MSB       $RET_DETAILS(upf_element_msb)
set RET_ELEM_LSB       $RET_DETAILS(upf_element_lsb)
set RET_CLK             $RET_DETAILS(upf_generic_clock)
```

Finally, the following bind\_checker command performs generic binding for all retention elements:

```
bind_checker ret_checker_inst -module ret_checker \
    -elements { $RET_ELEMENTS } \
    -parameters { {ELEM_NAME      $RET_ELEM_NAME} \
                  {ELEM_MSB       $RET_ELEM_MSB} \
                  {ELEM_LSB       $RET_ELEM_LSB} \
                  {RET_NAME        $RET_NAME} } \
    -ports " {ret_element $RET_ELEMENTS} \
            {restore_signal $RESTORE} \
            {clock $RET_CLK} "
```

As a result, this bind\_checker command inserts as many assertion modules as there are sequential elements.

The following instances show what these inserted assertions would look like:

```
checker_retention #( .RET_NAME("RET") , .ELEMENT_NAME("q1") ) chk_ret_q1
(.restore(tb.ret) , .upf_generic_clock(clk) , .element(q1) );

checker_retention #( .RET_NAME("RET") , .ELEMENT_NAME("q2") , .ELEMENT_LSB(4) ,
.ELEMENT_MSB(5) ) \chk_ret_q2[5:4] (.restore(tb.ret) , .upf_generic_clock(clk) ,
.element(q2[5:4]) );
```

### **Example 3-2. bind\_checker Assertion for Isolation**

Consider the following isolation strategy, which establishes the power domain and associated signals for isolation:

```
Isolation strategy
=====
set_isolation pd_iso \
    -domain pd \
    -isolation_supply_set ISO_SS \
    -clamp_value 0 \
    -isolation_signal iso \
    -isolation_sense high \
    -applies_to outputs
```

The following assertion module establishes the check that determines if the isolated port is properly clamped. This is the isolation checker module which is bound to the design.

```

module checker_isolation(iso_out, iso_ctrl, iso_clamp);
    parameter string ELEMENT_NAME = "";
    parameter int ELEMENT_LSB = 0;
    parameter int ELEMENT_MSB = 0;

    input [ELEMENT_MSB : ELEMENT_LSB] iso_out;
    input iso_ctrl, iso_clamp;

    always @(posedge iso_ctrl)
    begin
        assert((iso_out [ELEMENT_LSB] != iso_clamp)
            else $error("Incorrect clamp value for isolated port : %s[%d:%d]\n",
ELEMENT_NAME, ELEMENT_MSB, ELEMENT_LSB);
    end

endmodule

```

The following query\_isolation command assigns generics to the parameters of the assertion module:

```

array set ISO_DETAILS [query_isolation ISO -domain PD -detailed]
set ISO_NAME $ISO_DETAILS(isolation_name)
set ISO_ELEMENTS $ISO_DETAILS(elements)
set ISO_ELEM_NAME $ISO_DETAILS(upf_element_hier_path)
set ISO_ELEM_MSB $ISO_DETAILS(upf_element_msb)
set ISO_ELEM_LSB $ISO_DETAILS(upf_element_lsb)
set ISO_OUTPUT $ISO_DETAILS{upf_generic_output}
set ISO_CTRL $ISO_DETAILS{isolation_signal}
set ISO_CLAMP $ISO_DETAILS{clamp_value}

```

Finally, the following bind\_checker command performs generic binding for all isolation signals:

```

Bind checker stmt :
=====

bind_checker iso_checker_inst -module checker_isolation \
-elements { $ISO_ELEMENTS } \
-parameters { {ELEM_NAME $ISO_ELEM_NAME} \
{ELEM_MSB $ISO_ELEM_MSB} \
{ELEM_LSB $ISO_ELEM_LSB} \
{ISO_NAME $ISO_NAME} } \
-ports " {iso_out $ISO_OUTPUT} \
{iso_ctrl $ISO_CTRL} \
{iso_clamp $ISO_CLAMP} "

```

As a result, this bind\_checker command inserts as many assertion modules as the number of isolated ports specified in the strategy.

The following instances show what these inserted assertions would look like:

```
checker_isolation #(.ISO_NAME("ISO"), .ELEMENT_NAME("q1")) chk_iso_q1
(.iso_ctrl(tb.iso), .iso_clamp(0), .iso_out(q1) );

checker_isolation #(.ISO_NAME("ISO"), .ELEMENT_NAME("q2"), .ELEMENT_LSB(4),
.ELEMENT_MSB(5)) \chk_iso_q2[5:4] (.iso_ctrl(tb.iso), .iso_clamp(0),
.iso_out(q2[5:4]) );
```

## Power Aware Simulation Debug

Power Aware simulation enables you to create a database of information that for analyzing the results after completion of a live- simulation.

This is useful when you need to debug a Power Aware simulation that is run in a regression setup or where you are not able to debug a live-simulation.

The database contains information that enables you to perform supply network debugging and information for colorizing power information in the Dataflow window.

To create the post-simulation debug database, use the **-pa\_enable=supplynetworkdebug** argument to the vopt command (in the three-step, standard flow) or in the -voptargs argument to the vsim command (in the two-step, delayed optimization flow). For example:

```
vopt top -pa_upf top.upf -o DebugOut -pa_enable=supplynetworkdebug
[other vopt args]
```

Then, when you run simulation, add the **-pa\_debugdir <directory>** argument, which creates the necessary debug information for you to access post-simulation, for example:

```
vsim -pa_debugdir padbug -pa DebugOut -do "log -r /*; run -all"
```

After simulation has completed, load the WLF file and debug database to perform any debug activities on the results:

```
vsim -view vsim.wlf -pa_debugdir padbug -pa
```

**Table 3-1** provides an overview of the debugging features available during a live-simulation and post-simulation mode.

**Table 3-1. Power Aware Debug Support for Live-Sim and Post-Sim Modes**

<b>Power Aware Debug Feature</b>	<b>Live-Sim Mode</b>	<b>Post-Sim Mode</b>
Signal highlighting in the Wave window, including: <ul style="list-style-type: none"><li>• coloring/hashing</li><li>• popup information</li></ul> Refer to the section “ <a href="#">Power Aware Waveform Display</a> ” for more information	Supported	Supported
UPF object display in the Structure, Object, and Wave windows. Refer to the section “ <a href="#">UPF Object Display</a> ” for more information	Supported	Supported
Power Aware information in the: <ul style="list-style-type: none"><li>• Structure window</li><li>• Object window</li><li>• Wave window</li></ul> Refer to <a href="#">Graphical Display of Power Aware Operations and Results</a> for more information	Supported	Supported
Coloring, Isolation and level-shifter cell display in the Dataflow and Schematic windows Refer to the section “ <a href="#">Power Aware Schematic Display</a> ” for more information	Supported	Supported
Power Aware Static Check Debug using the Results Analysis window  <b>Note:</b> Not available for ModelSim SE (Questa SIM only). Refer to the section “ <a href="#">Power Aware Static Check Display</a> ” for more information	Supported	Supported Works without running the simulation, but does require access to the Results Analysis window

**Table 3-1. Power Aware Debug Support for Live-Sim and Post-Sim Modes**

Power Aware Debug Feature	Live-Sim Mode	Post-Sim Mode
Access to Power Aware checks in the Assertion window and message viewer	Supported	Supported <ul style="list-style-type: none"> <li>Invoke vopt should with +acc=a to view Power Aware checks in the Assertion window.</li> <li>Use -msgmode both to access information in the Message Viewer window.</li> </ul>
Power Aware test plan   <b>Restriction:</b> Not Available for ModelSim SE (Questa SIM only).  Refer to the section “ <a href="#">Analyzing Coverage Using the Power Aware Test Plan</a> ” for more information	Supported	Supported Use viewcov mode
Power/Port/PST State display  Refer to the section “ <a href="#">Power State and Transition Display</a> ” for more information	Supported	Not Supported
Power Aware source code debug	Supported	Supported
Power Aware coverage  Refer to the section “ <a href="#">Power Aware Coverage</a> ” for more information	Supported	Supported Use viewcov mode

# Power Intent Application to Mixed RTL and Gate-Level Designs

---

During Power Aware simulation of a mixed RTL/Gate-Level design, the power intent is applied automatically.

The following steps describe the flow:

1. Detect gate-level cells in the mixed netlist and perform the application of corruption based on the corresponding UPF strategies.
2. Detect the power aware cells in the design and match them with corresponding UPF strategies at that point in the design and run and flag power aware checks to report the valid cases and error out for anomalous cases.
3. Insert power aware cells wherever required by the corresponding UPF strategy and not already present in the netlist.
4. Detect UDPs and apply corruption and retention based on the corresponding UPF strategy.

You can experiment with some example code at the location *<install\_dir>/examples/pa\_sim/example\_four/* directory.

<b>Hard Macro Cell Detection</b> .....	<b>63</b>
<b>Gate-level Cell Detection</b> .....	<b>68</b>
<b>Power Management Cell Detection</b> .....	<b>69</b>
<b>Automatic Insertion of Power Management Cells</b> .....	<b>70</b>
<b>Automatic Corruption and Retention of UDPs</b> .....	<b>71</b>

## Hard Macro Cell Detection

Power Aware simulation detects certain instances in the design hierarchy as hard macro instances.

Detection of hard macro instances has two important effects in Power Aware simulation:

- Ports of a hard macro instance may define part of the lower boundary of a power domain, if their related supplies are different from the primary supply of the containing domain. This in turn enables insertion of isolation and level shifting for those ports.
- Hard macro instances may have their Power Aware simulation semantics provided by a Liberty model instead of the standard simstate-based Power Aware simulation behavior defined by UPF.

## Detecting and Enabling Hard Macros

An HDL simulation model is detected as a hard macro model if the simulation model has the attribute:

```
UPF_is_macro_cell=TRUE
```

associated with it, either by an HDL attribute specification or by a UPF **set\_design\_attributes** specification.

A Liberty model is detected as a hard macro model if the Liberty model contains the attribute:

```
is_macro_cell : true;
```

You can treat any HDL cell that has a Liberty model present as a hard macro by using the following vopt argument:

```
vopt -pa_enable=libertycellcrpt
```

The library-based corruption semantics of a hard macro is enabled by

```
vopt -pa_enable=libertypamodel
```

Without this argument, driver-based corruption semantics are applied for macro cells.

## Liberty-based Corruption Semantics of Macro Cells

When the supply ports are present in the HDL model, then the simulator assumes that you are using a Power Aware HDL model. Simulation semantics are disabled for such design elements. If the HDL model is not power aware and there is a corresponding Liberty cell for this model, then Liberty-based corruption semantics are applied. In this case, input ports are corrupted using their related supply, and output ports are corrupted using the power-down function.

To obtain a more optimized Liberty-based corruption, you can use the following argument:

```
vopt -pa_enable=libertypamodelopt
```

With this argument, Liberty corruption semantics are applied:

- For combinatorial cells, only outputs are corrupted.
- For sequential cells, both outputs and inputs are corrupted.

If your Liberty hard macro model has some other supplies (such as backup\_power or bias power/ground pins), and these supplies are unconnected, this may result in spurious corruption semantics. To avoid this problem, you can use the following argument:

```
vopt -pa_enable=powerunconnnets
```

This vopt argument causes the unconnected pins for a Liberty macro model to be treated as power-on, so that these unconnected pins do not cause undesired corruption semantics.

## Corruption Semantics of Power Management Cells

When a Liberty model is present for power management cells, the simulator applies a connection to the supply pins (such as primary\_power, backup\_power, bias) according to the Power Aware cell type, as described below. After that, the usual Liberty based corruption semantics are applied.

### Retention Cell

If a cell is detected as a retention cell but there is no matching UPF strategy, then the simulation semantics of the retention cell is disabled.

If a cell is inferred as retention cell, and it is matched with a UPF strategy, then the simulator tries to apply automatic connections (if explicit connections are not present). Primary\_power is connected to supply of power domain and backup\_power is connected to retention supply from the UPF strategy. If the strategy specifies set\_retention -use\_retention\_is\_primary, then primary\_power and backup\_power both are connected to the retention supply from the UPF strategy.

- If liberty model of the cell is present: Liberty based corruption semantics are applied.
- If the liberty model of the cell is absent: Driver based corruption is applied based on the retention supplies specified with the UPF strategy

### Always-on Cell

Because the simulator does not apply any default automatic connections for always-on cells, you need to make explicit connections for these cells. Always-on cells are corrupted only if the backup power goes off. To disable Liberty-based corruption of always-on cells, use vopt -pa\_disable=aoncellcrpt. Corruption semantics of always-on cells are disabled when the Liberty model for these cells is not present.

### Switch Cell

If a cell is detected as a switch cell but there is no matching UPF strategy, then the simulation semantics of the switch cell are disabled.

If a cell is inferred as a switch cell and is matched with a UPF strategy, then the simulation semantics of the switch cell are applied according to the following:

- Liberty model of the cell is present — If the explicit connection is missing then the simulator applies the default auto connection using supplies specified with the UPF strategy. If both default auto connection and explicit connections are missing, then simulation semantics are disabled. Otherwise, Liberty-based corruption semantics are applied. To disable Liberty-based corruption semantics for switch cells, use vopt -pa\_disable=swcellcrpt.

- Liberty model of the cell is absent — Driver-based corruption is applied based on the -supply\_set specified with the UPF strategy. If the supply is not specified, then simulation semantics are disabled.

### Level Shifter Cell

If a cell is detected as a level shifter cell but there is no matching UPF strategy, then the simulation semantics of the level shifter cell are disabled.

If a cell is inferred as a level shifter cell and is matched with a UPF strategy, then the simulation semantics of the level shifter cell are applied according to the following:

- Liberty model of the cell is present — If the explicit connection is missing then the simulator applies the default auto connection using supplies specified with the UPF strategy. If both default auto connection and explicit connections are missing, then simulation semantics are disabled. Otherwise, Liberty-based corruption semantics are applied. To disable Liberty-based corruption semantics for level shifter cells, use vopt -pa\_disable=lscellcrpt.
- Liberty model of the cell is absent — Driver-based corruption is applied based on the supplies specified with the UPF strategy. If the supply is not specified, then simulation semantics are disabled.

### Isolation Cell

If a cell is detected as an isolation cell but has no matching UPF strategy, simulation semantics of the isolation cell are disabled.

If a cell is inferred as an isolation cell that has a matching UPF strategy, corruption behavior is applied according to the following:

- Liberty model of the cell is present — If explicit supply connections are missing, then default auto connections are applied, depending on the location of the cell. For more information on these cell location dependencies, see “[Mapping Power, Ground, or Bias Pins of Isolation Cells with UPF Supplies](#)”. Liberty-based corruption semantics are applied.
- Liberty model of the cell is absent — Driver-based corruption is applied based on the set\_isolation -isolation\_supply\_set as specified with the UPF strategy.

### Enable Level Shifter (ELS) Cell

The simulator detects a cell as an ELS cell when either of the following Liberty attributes is specified in the Liberty file:

- is\_level\_shifter : TRUE and is\_isolation\_cell : TRUE
- is\_level\_shifter : TRUE, and level\_shifter\_enable\_pin or isolation\_cell\_enable\_pin : TRUE

If a cell is detected as an ELS cell, the simulator connects the power supply of the ELS cell according to the connections specified in the following commands, and in the given order:

1. [connect\\_supply\\_net](#)
2. [set\\_level\\_shifter](#) with supply sets
3. vopt -pa\_enable=autoconnls (See “autoconnls” in [Table A-3](#).)

If the above commands are missing, the simulator follows the listed semantics, and in the given order:

1. If an ELS cell matches with a level shifter strategy with no or incomplete supply set details, the simulator connects the power supply of the ELS cells according to the following connection semantics:

- o The input supply port is connected to the source domain primary supply set.
- o The output supply port is connected to the sink domain primary supply set.

If the ELS cell has bias pins, the nwell or pwell supply ports are specified as a related bias supply to either the input or output supply port.

- o If the nwell or pwell supply port is specified as a related bias supply to the input supply port, then the nwell or pwell supply port is connected to the nwell or pwell supply set function of the source domain primary supply set.
- o If the nwell or pwell supply port is specified as a related bias supply to the output supply port, then the nwell or pwell supply port is connected to the nwell or pwell supply set function of the sink domain primary supply set.

2. If an ELS cell matches with an isolation strategy, the simulator connects the power supply of the ELS cells according to the connection semantics mentioned in “Isolation Cell” section of [“Corruption Semantics of Power Management Cells”](#).
3. If an ELS cell does not match with any UPF strategy, then the simulator does not connect the power supply of the ELS cell.

## Mapping Power, Ground, or Bias Pins of Isolation Cells with UPF Supplies

When you map power, ground, or bias pins of isolation cells with UPF supplies, you should observe the following requirements that depend on the location of the cell.

### Cell Location Is Self

The following requirements apply when mapping pins at the location of the isolation cell.

- Connect cell primary supply pins with primary supply set of domain in which the cell is present.
- Connect cell backup or secondary supply pins with isolation supply set of isolation strategy specified in the UPF strategy.

- For well bias supply connections, map according to the following cases:
  - When a well bias pin is related to only primary supply or to both primary plus backup supply, or it is not related to either primary or backup supply in Liberty—connect the well bias pin with well bias supply from primary supply set of domain in which the cell is present.
  - When well bias pin is related to only backup supply in Liberty—connect the well bias pin with well bias supply from isolation supply set of the isolation strategy specified in the UPF strategy.

### Cell Location Is Parent

The following requirements apply when mapping pins for an isolation cell in the parent location.

- Cell primary supply pins should be connected with isolation supply set of isolation strategy specified in UPF.
- Cell backup/secondary supply pins should be connected with primary supply set of domain in which the cell is present.
- For well bias supply connections, map according to the following cases:
  - When a well bias pin is related to only the primary supply or to both primary plus backup supply, or it is not related to either primary or backup supply in Liberty—connect the well bias pin with well bias supply from isolation supply set of the isolation strategy specified in the UPF.
  - When well bias pin is related to only the backup supply in Liberty—connect the well bias pin with well bias supply from the primary supply set of the domain in which the cell is present.

### Related Topics

[Liberty Models of Hard Macros](#)

[Macro Models Concepts](#)

## Gate-level Cell Detection

Power Aware simulation automatically treats a module as a gate-level cell if the module contains the ``celldefine` attribute or the module contains the `specify` block.

Corruption occurs on the output ports and sequential logic of any detected gate-level cells. For RTL cells the standard processing of driver-based corruption is still applied.

Power Aware simulation enables a gate-level cell to be treated as an RTL cell, specifically a candidate for driver based corruption. You can achieve this by setting the UPF attribute `UPF_is_leaf_cell` as 'false' on the module. Also an RTL cell (where the module has no ``celldefine` attribute or a ``specify` block, or it is a VHDL cell) can be treated as gate-level cell

(candidate for output only corruption) by setting the UPF attribute UPF\_is\_leaf\_cell as 'true' on the cell. The attribute UPF\_is\_leaf\_cell can be applied on both Verilog and VHDL cells.

## Power Management Cell Detection

Power Aware simulation automatically detects isolation, level-shifter and retention cells in the netlist and infers them for any corresponding UPF strategies. Power Aware simulation then validates the cells, with respect to strategies in the UPF, and run power aware check on them.

Gate-level netlists may have some of the power management cells (isolation, level-shifter, or retention) corresponding to the UPF strategy already instantiated in the netlist. Some of these cells may already be specified as a value for the -instance option of the set\_isolation, set\_level\_shifter, or set\_retention commands of the UPF strategy in UPF file.

For the rest of the cells that are not specified in the UPF file, Power Aware simulation automatically detects the right UPF strategy to which they belong and treats them in a similar way to cells of that strategy specified with an -instance option.

Auto detection of power management cells leverages the following information for:

- Liberty attributes
  - is\_isolation\_cell
  - is\_level\_shifter
  - retention\_cell
- lib\_cells specified with the following UPF commands:
  - map\_isolation\_cell
  - map\_level\_shifter cell
  - map\_retention\_cell
- Options for the UPF name\_format command:
  - -level\_shift\_prefix
  - -level\_shift\_suffix
  - -isolation\_prefix
  - -isolation\_suffix
- Synopsys pragmas
  - isolation\_upf
  - retention\_upf

## Reports

The cells detected as an instance of some strategy are reported in *report.pa.txt*, for example:

```
Power Domain: A, File: ./src/case1/test.upf(11).
    Creation Scope: /tb/dut
    ...
    Isolation Strategy: ISO1, File: ./src/case1/test.upf(22).
        Isolation Supplies:
            power : /tb/dut/VDD_0d99
            ground : /tb/dut/VSS_0d99
        Isolation Control (/tb/dut/restore), Isolation Sense (HIGH), Clamp
Value (0), Location (fanout)
        Signals with -instance isolation cells:
            1. Signal : /tb/dut/instA/out, isolation cell : /tb/dut/
iso_1_UPF_ISO
```

## Messages

When Power Aware simulation is unable to detect the UPF strategy of an isolation/level-shifter cell, Power Aware simulation semantics are disabled and the cell is treated as always ON. The following message is displayed:

```
** Warning: (vopt-9768) Power aware simulation semantics disabled for '/tb/dut/instA/ls_0_UPF_LS' as its power aware strategy could not be identified.
```

For a cell identified as a retention cell, Power Aware simulation flags a warning if the cell is also identified as a level-shifter or isolation cell. Power Aware simulation then processes it as either a retention or an isolation/level-shifter cell, and the following message is displayed:

```
** Warning: UPF: (vopt-9823) Power aware cell '/tb/dut/iso_1' identified as both 'isolation' cell and 'retention' cell. Assuming it to be a 'isolation' cell
```

# Automatic Insertion of Power Management Cells

Power Aware simulation inserts power management cells (level-shifter, isolation, and retention) into the netlist based on UPF strategy commands.

You can selectively disable each type of insertion with the following vopt arguments:

- `-pa_disable=insertiso` — to disable isolation cell insertion
- `-pa_disable=insertls` — to disable level\_shifter cell insertion
- `-pa_disable=insertret` — to disable retention cell insertion

For pure gate-level designs in which all isolation, level shifting, and retention cells are already all present, and therefore none of these cells need to be inserted, the following vopt argument can be used as a shorthand where `-pa_gls` is equivalent to `-pa_disable=insertiso -pa_disable=insertls -pa_disable=insertret`

## Automatic Corruption and Retention of UDPs

Power Aware simulation detects sequential UDPs in the design and applies corruption or retention behavior based on the corresponding UPF strategies.

This functionality is not supported in the no-optimization flow. In that flow, a warning is issued if there is any sequential UDP in the design.

### UDP Corruption and Retention Modes

This section defines different modes of operation of the retention and corruption logic for sequential UDPs in a gate-level or mixed RTL and gate-level simulation. These modes describe the functionality of different sequential UDPs introduced to mimic corruption and retention behavior.

- Save mode (single control) — Occurs when save is active.
- Save mode (dual control) — Occurs when save is active and restore is inactive.
- Restore mode (single control) — Occurs when save is inactive.
- Restore mode (dual control) — Occurs when save is inactive and restore is active.
- Error mode (dual control) — When the UPF **set\_retention** parameter SAV\_RES\_COR is true the output of the register and the retained value of the register are both corrupted if the save and restore both are active and **save\_condition** and **restore\_condition** are true. This occurs only when neither the save signal nor the restore signal are edge sensitive.
- No change mode (dual control) — Occurs when save and restore are both inactive. In this case, the register is neither in save nor in restore mode. Therefore, the behavior would be as if it is a normal register with no retention facility present.

---

#### Note

 User defined models specified using the **map\_retention\_cell** command is not honored for sequential UDP retention. However the RTL portion of design follows the user defined model simulation semantics given with **map\_retention\_cell** command.

---

### Limitations

- Power Aware simulation is unable to determine whether retention logic is already present in a sequential UDP and therefore routinely inserts retention logic for such UDPs. Use the UPF **set\_retention** command with the -instance <inst\_name> option to avoid application of retention semantics to UDPs that already include retention logic.
- Power Aware simulation is unable to determine whether a UDP is power aware--that is, whether the UDP references supply ports and corrupts its outputs based on the values of those supply ports. Consequently Power Aware simulation assumes that all UDPs are not power aware and routinely applies corruption on UDP logic.

- Power Aware simulation is unable to process sequential UDPs that involve multiple clocks.

# UPF Supply Connections

Supply connections between various UPF objects can made between supply nets, supply sets, supply ports, power switches and UPF objects such as retention, isolation, level shifter cells. You can define these connections in the following ways:

<b>Implicit Connections</b> .....	<b>73</b>
<b>Explicit Connections</b> .....	<b>73</b>
<b>Automatic Connections</b> .....	<b>75</b>
<b>Simulation Semantics for UPF Supply Connections</b> .....	<b>78</b>

## Implicit Connections

Implicit connections provide a way to connect supply nets to elements that do not have supply ports. Any design element that is present in the extent of power domain and does not have supply ports connected or is excluded from Power Aware processing (-pa\_excludefile) is implicitly connected with the primary supplies (power and ground) of the power domain. Thus, the corruption of that element depends on the state of the primary power net and primary ground net of the power domain.

## Explicit Connections

You can explicitly connect a supply net to a supply port using the `connect_supply_net` UPF command,

This explicit connection overrides (has higher precedence than) the implicit and automatic connection semantics that might otherwise apply.

Explicit connections include:

- Connections to UPF-created supply ports
- Connections to HDL-created supply ports (`supply_net_type` or 1-bit type)
- Connections to supply ports of power switches

Explicit connection to an HDL supply port has simulation semantics disabled by default (see “[Simulation Semantics for UPF Supply Connections](#)”). Driving an HDL port from UPF overrides its RTL connection.

### Explicit Connections to HDL Ports

Using the `connect_supply_net` command, you can connect a UPF-created supply net to an HDL-created supply port and a UPF-created supply port to an HDL-created supply net. To ensure the supply net state and voltage values are propagated and modeled in the various HDLs, you must use the `supply_net_type` datatype. You can make these datatypes visible by importing

UPF defined HDL packages (refer to Appendix B in UPF 1.0 and Annex B in UPF 2.0 onwards). For example:

- HDL specification (Verilog):

```
module memory(input supply_net_type vdd, ...);  
...  
endmodule
```

- HDL specification (VHDL):

```
entity memory is  
port (vdd : in supply_net_type; ...)  
...  
end entity
```

- UPF specification:

```
connect_supply_net vdd_switchable -ports mem/vdd
```

## Explicit Connections to 1-bit HDL Ports

UPF also enables supply connections between supply nets and 1-bit Verilog/VHDL ports for building simple functional models. In these cases, HDL supply ports are connected to the ON/OFF state bit of the supply net. For complex modeling, you can use value change tables (VCTs) for the conversion from the supply net state to values relevant to an HDL type, or vice-versa.

---

### Note

---

 Enumerated HDL types are not supported.

---

- HDL specification (Verilog):

```
module memory(input vdd, ...);  
...  
endmodule
```

- HDL specification (VHDL):

```
entity memory is  
port (vdd : in std_logic; ...)  
...  
end entity
```

- UPF specification:

```
connect_supply_net vdd_switchable -ports mem/vdd
```

### Explicit Connections to Supply Ports of Power Switch

You can use the connect\_supply\_net command to define connections to supply ports of power switch. For example:

```
create_power_switch SW \
    -input_supply_ports {IN_SW} \
    ...
connect_supply_net VDD_IN -ports {SW.IN_SW}
```

You can also specify connections using the create\_power\_switch command. For example:

```
create_power_switch SW \
    -input_supply_ports {IN_SW VDD_IN} \
    ...
```

## Automatic Connections

The Questa SIM simulator supports the UPF automatic connection semantics of supply nets and supply sets as defined by IEEE Std 1801-2009.

Automatic connections semantics are defined for:

- Supply nets — using the connect\_supply\_net command (see “[Automatic Connections for Supply Nets](#)”)
- Supply sets — using either the connect\_supply\_set or create\_power\_domain command (see “[Automatic Connections for Supply Sets](#)”)

The necessary conditions for the supply nets (or supply nets of supply sets) to automatically connected to the supply ports of design elements are:

- Design element has a port with -pg\_type attribute. The pg\_type attribute includes
  - String type HDL attribute named either pg\_type or UPF\_pg\_type.
  - Implementation library model with pg\_type attribute.
- Value of the pg\_type attribute of the port matches with the pg\_type value specified using UPF commands.

Simulation semantics of all the design elements whose ports are automatically connected to power supply are disabled (see “[Simulation Semantics for UPF Supply Connections](#)”).

### Automatic Connections for Supply Nets

You can define automatic connection semantics on individual supply nets using the following UPF command:

```
connect_supply_net -pg_type -domain -cells
```

The Questa SIM simulator automatically connects the specified supply net with the supply ports on the specified cell or the design elements within the extent of power domain which fulfills the necessary conditions for automatic connection semantics. You can also specify Value Conversion Tables (VCTs) for automatic connection semantics by using the -vct option.

Using the -vct option has a restriction that all the ports that are connected using connect\_supply\_net should have matching types, since the type matching for VCT produces error during connections.

For example:

```
connect_supply_net snet -domain PD -pg_type primary_power -vct my_sv_vct
```

This produces an error message when domain PD has a VHDL instance contain supply port with pg\_type primary\_power and different type and create the connection, assuming a 1-bit connection.

### Command Syntax

UPF 1.0:

```
connect_supply_net net_name
  [-ports list] [-pins list]
  [<-cells list | -domain domain_name>]
  [<-rail_connection rail_type | -pg_type pg_type>] *
  [-vct vct_name]
```

UPF 2.0:

```
connect_supply_net net_name
  [-ports list]
  [-pg_type {pg_type_list element_list}]*
  [-vct vct_name] [-pins list]
  [-cells list] [-domain domain_name]
  [-rail_connection rail_type]
```

Currently, the connect\_supply\_net command in UPF 2.0 is modeled as its equivalent command in UPF 1.0. That is, connect\_supply\_net command in UPF 2.0 does not accept element\_list in -pg\_type option.

### Examples

```
connect_supply_net VDD -domain PD_SW -pg_type primary_power
connect_supply_net Vdd_backup -cells {RET_CELL} -pg_type backup_power
connect_supply_net VSS -domain PD_SW -pg_type primary_ground -vct
  UPF_GNDZERO2SV_LOGIC.
```

## Automatic Connections for Supply Sets

You can define automatic connection semantics on supply sets using either of the following UPF commands:

```
connect_supply_set -connect
create_power_domain -define_func_type
```

Power Aware simulation automatically connects supply nets of supply sets to the supply ports of design elements. This connection is based on the purpose of the supply set in a given domain or strategy context and the function that a supply net performs in the context of supply set.

### Limitations

- Supply sets specified in strategy context is not automatically connected to design elements owing to separate infrastructure in -instance.
- Supply nets in supply sets functioning as predefined supply set functions are not automatically connected as per their predefined function. Explicit specification of automatic connections must be specified.

### Command Syntax

Using connect\_supply\_set:

```
connect_supply_set supply_set_ref
  {-connect {supply_function {pg_type_list}}}*/
  [-elements element_list]
  [-exclude_elements exclude_list]
  [-transitive ]
```

Using create\_power\_domain:

```
create_power_domain
  [-define_func_type {supply_function {pg_type_list}}]*
```

### Examples

```
create_power_domain PD \
  -define_func_type {power primary_power} \
  -define_func_type {always_on backup_power} \
  ..

connect_supply_set PD.primary \
  -connect {power primary_power} \
  -elements TOP

connect_supply_set PD.ISO.isolation_supply_set \
  -connect {iso_power primary_power} \
  -connect {iso_ground primary_ground} \

connect_supply_set PD.RET.retention_supply_set \
  -connect {ret_backup_power backup_power} \
  -connect {switchtable_supply primary_power}
```

## Simulation Semantics for UPF Supply Connections

Power Aware simulation semantics are automatically disabled for design elements that are explicitly or automatically connected to a particular supply net or supply set. Simulation semantics are disabled for all the descendants of the instance of model.

When automatic connection semantics are applied, simulation semantics of all the instances whose ports are automatically connected is disabled. This may display a notification message similar to the following:

RTL Specification (Verilog):

```
module memory(input vdd, ...);  
...  
endmodule
```

RTL Specification (VHDL):

```
entity memory is  
port (vdd : in std_logic; ...)  
..  
end entity
```

UPF specification:

```
create_power_domain mem_pd -elements mem  
connect_supply_net vdd_switchable -ports mem/vdd  
  
** Note: top.upf(12): (vopt-9693) Power Aware simulation semantics  
disabled for /testbench/rtl_top/mem
```

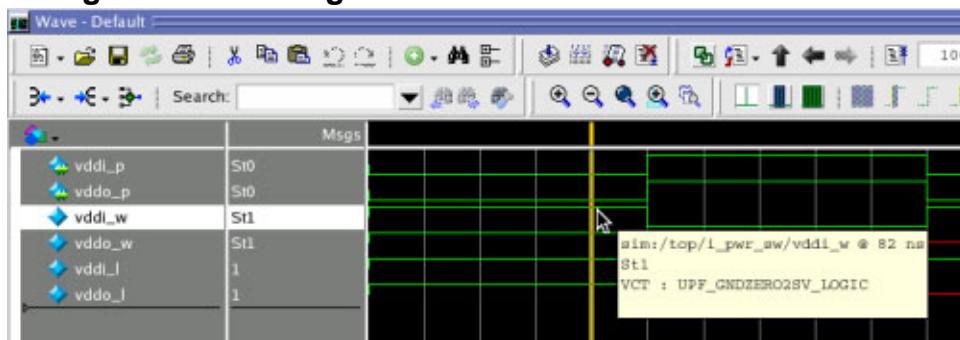
It is recommended that you use vopt -suppress 9693 to suppress this message.

## Value Conversion Tables

A value conversion table (VCT) is a UPF definition (IEEE Std1801-2009) of how to convert, or map, a value from a supply net state relevant to an HDL variable type (and from an HDL variable type to a supply net state). This mapping enables complex modeling of connections between supply nets and RTL ports.

You can identify which VCT is used for a given pin in the Wave window by hovering the mouse pointer over the waveform for the port/net, as shown in [Figure 3-2](#).

**Figure 3-2. Viewing VCT Information in the Wave Window**



<b>Using VCT Commands .....</b>	<b>79</b>
<b>Connections Using Value Conversion Tables (VCTs) .....</b>	<b>81</b>
<b>Questa-specific Value Conversion Tables .....</b>	<b>83</b>

## Using VCT Commands

You can use the `create_upf2hdl_vct` and `create_hdl2upf_vct` commands in the UPF file for modeling complex connections between a supply net and RTL port.

Power Aware simulation also automatically inserts VCT for pins detected as power and ground pins when you specify `vopt -pa_upfextensions`.

### Predefined VCTs Supported from the UPF Standard

Power Aware simulation provides several predefined VCTs described in Annex C of IEEE Std1801-2009, which you can use with the `connect_supply_net -vct` command.

You can also refer to these predefined VCTs in the file:

```
<install_dir>/upf_src/upfUtilities.upf
```

### Examples

The following examples show the `create_upf2hdl_vct` and `create_hdl2upf_vct` commands in a UPF file.

```
create_hdl2upf_vct VHDL_SL2UPF \
    -hdl_type {vhdl} \
    -table { {'U' UNDETERMINED} \
        {'X' UNDETERMINED} \
        {'0' OFF} \
        {'1' FULL_ON} \
        {'Z' UNDETERMINED} \
        {'L' OFF} \
        {'H' FULL_ON} \
        {'W' UNDETERMINED} \
        {'-' UNDETERMINED} }
```

```
create_upf2hdl_vct UPF_GNDZERO2SV_LOGIC \
    -hdl_type sv \
    -table {{UNDETERMINED X} \
        {PARTIAL_ON X} \
        {OFF 1} \
        {FULL_ON 0} }
```

If VCT definition for a connection is not applicable or fails, then default 1-bit connection semantics is applied. For example:

```
module memory (input gnd, ...);
...
endmodule

connect_supply_net gnd_switchable -ports mem_sv/gnd -vct
SV_LOGIC2UPF_GNDZERO
```

Data flow for the connection is UPF to RTL but VCT is specified as RTL to UPF. Hence, specified VCT is not applicable.

## Limitations

Only the following HDL data types are supported:

- System Verilog — logic, bit, wire, reg
- Verilog — wire, reg
- VHDL — bit, std\_logic, std\_ulogic, Boolean (NOTE: Any subtypes of these are not supported.)

# Connections Using Value Conversion Tables (VCTs)

You can model more complex connections between supply nets and HDL ports by using either of the following UPF commands to define value conversions:

```
create_upf2hdl_vct
```

```
create_hdl2upf_vct
```

## Limitations

Only following types are supported in HDL data types:

- SystemVerilog (logic, bit, wire, reg)
- Verilog (wire, reg)
- VHDL (bit, std\_logic, std\_ulogic, Boolean).

Any subtypes of these are not supported.

## Examples

```
create_hdl2upf_vct VHDL_SL2UPF \
-hdl_type {vhdl} \
-table { {'U' UNDETERMINED} \
{'X' UNDETERMINED} \
{'0' OFF} \
{'1' FULL_ON} \
{'Z' UNDETERMINED} \
{'L' OFF} \
{'H' FULL_ON} \
{'W' UNDETERMINED} \
{ '-' UNDETERMINED} }
```

```
create_upf2hdl_vct UPF_GNDZERO2SV_LOGIC \
-hdl_type sv \
-table {{UNDETERMINED X} \
{PARTIAL_ON X} \
{OFF 1} \
{FULL_ON 0}}
```

In addition, you can specify an existing value conversion table (VCT) using the following UPF command and option:

```
connect_supply_net -vct
```

**Note**

 Power Aware simulation provides several predefined VCT definitions (see “[Value Conversion Tables](#)”) that you can connect using `connect_supply_net -vct`. However, if the VCT specification for a connection is not applicable or valid, then Power Aware simulation applies the default 1-bit connection semantics.

---

## Examples

RTL Specification (Verilog):

```
module memory (input VSS, VDD);  
..  
endmodule
```

RTL Specification (VHDL):

```
entity memory is  
port (VSS : in std_logic; VDD: in std_logic;..)  
..  
end entity
```

UPF specification:

```
connect_supply_net VDD_switchable -ports mem_sv/VDD  
connect_supply_net VSS_switchable -ports mem_sv/VSS -vct  
UPF_GNDZERO2SV_LOGIC  
  
connect_supply_net VDD_switchable -ports mem_vhd/VDD  
connect_supply_net VSS_switchable -ports mem_vhd/gnd -vct  
UPF_GNDZERO2VHDL_SL
```

RTL Specification (Verilog):

```
module memory (input VSS, ..);  
..  
endmodule
```

UPF specification:

```
connect_supply_net VSS_switchable -ports mem_sv/VSS -vct  
SV_LOGIC2UPF_GNDZERO
```

Data flow for the connection is UPF to HDL, but VCT is specified as HDL to UPF. As a result, the specified VCT is not applicable.

## Questa-specific Value Conversion Tables

Automatic VCTs are applied when you specify the `-pa_upfextensions=v` and `-pa_enable=libertypamodel` vopt arguments.

You see reference to these VCTs in the `report.cell.txt` and `report.connection.txt` report files.

- VCTs used for pg\_types: primary\_power, backup\_power, internal\_power and nwell
  - MSPA\_CONNECT\_UPF\_2\_HDL\_PWR — converts a UPF supply net to an HDL single bit. Specifically, it converts a FULL\_ON state to 1 and all other supply states to 0.
  - MSPA\_CONNECT\_HDL\_2\_UPF\_PWR — converts an HDL single bit to a UPF supply net type. Specifically, it converts an HDL 1 to a FULL\_ON state and everything else to OFF.
- VCTs used for pg\_types: primary\_ground, backup\_ground, internal\_ground and pwell
  - MSPA\_CONNECT\_UPF\_2\_HDL\_GND — converts a UPF supply net to an HDL single bit. Specifically, it converts an OFF state to 0 and all other states to 1.
  - MSPA\_CONNECT\_HDL\_2\_UPF\_GND — converts HDL single bit to UPF supply net type. Specifically, it converts an HDL 0 to a FULL\_ON state and everything else to OFF.
- VCTs used for when either no pg\_type is specified or the pg\_type does not match any of the pg\_types specified above
  - MSPA\_CONN\_UPF\_2\_HDL — converts a UPF supply net to an HDL single bit. Specifically, it converts a FULL\_ON state to 1 and all other supply states to 0.
  - MSPA\_CONN\_HDL\_2\_UPF — converts an HDL single bit to a UPF supply net type. Specifically, it converts an HDL 1 to FULL\_ON state and everything else to OFF.

## Defining Isolation

Isolation is used to separate signals that originate in a design element with power off from a part of the design that has power on and that can still read the signals from the powered down element. A particular domain may be powered off while another domain is operating in normal mode.

There are two methods for defining an isolation cell.

- Method 1: Isolation is already explicitly present

In this case, the design (in either a RTL or a GL netlist) contains an explicit cell instance that functions as an isolation cell. The UPF file includes a `set_isolation -instance` command that identifies this explicit instance as an isolation cell. In this case:

- No isolation needs to be added
- Power Aware does not insert an additional isolation cell
- Power Aware does not modify the explicit isolation cell that is present (it is assumed that the user-provided isolation cell does what it is supposed to do)

- Method 2: Isolation needs to be added

In this case, you use the `set_isolation` command to provide an isolation strategy (without the `-instance` option). Power Aware simulation then implicitly inserts an isolation cell on any port that satisfies all of the following:

- Matches the criteria defined in the isolation strategy
- Requires isolation because the power state table indicates that the two power domains on either side of the port can be ON/OFF or OFF/ON respectively
- Does not already have an explicit isolation cell present and identified (per Method 1)

By default, if Power Aware simulation does insert an isolation cell, Power Aware simulation uses a built-in behavioral model for isolation. However, you can cause the simulation to insert a different model (when required) by using the `map_isolation_cell` UPF command to identify the isolation model to be used.

---

#### Note

 To prevent a redundant isolation cell from being inserted on a port, use the `set_isolation -instance` command to identify the instance of the existing isolation cell.

---

## Specifying Isolation Cells

Power Aware simulation reads the `set_isolation` command in the UPF file and identifies ports as candidates for isolation cell insertion. These ports are also dumped into the UPF report file (`report.pa.txt`), as in [Example 3-3](#).

### Example 3-3. UPF Report File for Isolation Ports (*report.pa.txt*)

```
Isolation Strategy: iso_PD_mid1, File: test.upf(44).
  Isolation Supplies:
    power : /tb/TOP/tb_pow
    ground : /tb/TOP/mid1_GND_NET
  Isolation Control (/tb/TOP/ctrl), Isolation Sense (HIGH), Clamp Value
(1), Location (automatic)

  Isolated Signals:
    1. Signal : /tb/TOP/mid1/out2_bot
    2. Signal : /tb/TOP/mid1/out1_bot
```

#### Isolation Cell Instances

If you have used the set\_isolation -instance command in a UPF file to instantiate RTL isolation cells, Power Aware simulation infers those cells and perform isolation checks on them.

**Tip**  Power Aware simulation automatically infers the right UPF strategy for cells that are not specified in the UPF file. For more information, refer to “[Power Management Cell Detection](#).”

---

A cell instance is identified as an isolation cell in any of the following cases:

- Instance is specified with -instance option of set\_isolation command.
- For a GLS design, any of the following is present:
  - Synopsys pragma synopsys isolation\_upf. For example:

```
ISOLOD1BWP trafficWriteEnable_UPF_ISO ( .I(trafficWriteEnable),
.ISO(n16), .Z(n57) ); //synopsys isolation_upf PD_C2_ISS1+PD_C2
```

- Liberty attribute is\_isolation\_cell. For example:

```
cell (ISO_LO) {
  is_isolation_cell : true;
  ...
```

- HDL attribute is\_isolation\_cell. For example:

```
(* is_isolation_cell = 1 *)
  module ISOCELL ( I, ISO, Z) ;
  ...
endmodule
```

- Cell in accordance with map\_isolation\_cell UPF command. For example:

```
UPF Command : map_isolation_cell mem_ctrl_iso_0 -domain
PD_mem_ctrl -lib_cells {ISO_LO}
  Instantiation in hdl : ISO_LO addr_0 ( .I(n158),
.ISO(n174), .Z(address[0]) );
```

- Instance name in accordance with name\_format UPF command. For example:

```
UPF Command : name_format -isolation_prefix "MY_ISO_"
-isolation_suffix "UPF_ISO"
    Instantiation in hdl : SEN_OR2_4
    MY_ISO_t_state_21_UPF_ISO (.A1 ( n151 ) , .A2 ( n354 ) , .X (
        t_state[21] ) );
```

### Limitations

If isolation is required and is not present (and identified with set\_isolation -instance), then Power Aware simulation effectively inserts an isolation cell.

The current Power Aware architecture implements isolation by *actually* inserting a cell—an instance of a behavioral model for isolation—into the design where isolation is required. However, ports on an isolation cell inserted this way have a limitation of not supporting enum types.

For ports of type enum, Power Aware reverts to the implementation approach used in previous releases. In that approach, an isolation cell is not *actually* inserted into the design. Instead, internal mechanisms (such as the force command) make an existing design port (highconn or lowconn) behave as if isolation were present.

The isolation effect of these two approaches is identical—you see exactly the same results in reports and in the GUI. The only difference is that the new architecture shows explicit instances of isolation cells that have been added to the design.

# Defining Retention

---

To define retention registers in a power domain and set the corresponding save and restore signals for the retention registers, you use the `set_retention` command in your UPF file. In particular, you can use the following options (supported in UPF 2.0 and newer versions) to define your retention strategy:

- `-retention_supply_set`— Defines the supply set used to power the logic inferred by the `<retention_name>` strategy.
- `-no_retention` — Storage elements specified by this option are prevented from having retention capability.
- `-use_retention_as_primary` — Specifies that the storage element and its output are powered by the retention supply.

<b><code>-retention_supply_set</code></b> . . . . .	<b>87</b>
<b><code>-no_retention</code></b> . . . . .	<b>88</b>
<b><code>-use_retention_as_primary</code></b> . . . . .	<b>90</b>

## **-retention\_supply\_set**

This option defines the supply set used to power the register holding the retained value.

Note the following cases:

- If you have specified both retention power and retention ground nets, using this command creates an implicit retention supply set and is used with the specified strategy. The retention power net serves the power function in the retention supply set and the retention ground net serves the ground function in the retention supply set.
- If you have specified the retention power net but not the retention ground net, then the domain's primary supply sets ground function as the retention ground.
- If you have specified the retention ground net but not the retention power net, then the domain's primary supply sets power function as the retention power.

The power of the retention cell is preserved (an extra port RETPWR (Retention Power) is added to the retention models).

Whenever the power of a retention cell (RETPWR) goes down, the value stored in retention cell gets corrupted, and the 'X' value is restored when restore signal is triggered.

## Example

Consider the following logic added in a retention model:

```
// store x in RETPWRDOWN
  always @(negedge RETPWR)
  begin
    -> pa_store_x ;
  end
```

An extra port RETPWR has been added in the Power Aware retention models.

Whenever you use your own retention models using the [map\\_retention\\_cell](#) UPF command:

- For a user-defined model with RETPWR port and corresponding logic—Behavior is in accordance with the RETPWR logic.
- For a user-defined model without a RETPWR port — A warning is issued during the vsim simulation. For example:

```
# ** Warning: (vsim-PA-8944) Retention Model : 'MyRetModel' does not
have Port 'RETPWR'. Ignoring it.
# Region: /mspa_top/bk0/inst0_MyRetModel
```

## **-no\_retention**

This option disables retention on specified storage elements.

## Example

```

upf_version 2.0
set_scope tb1
create_power_domain PD_TOP -elements { top1/bot1/tdsig}
create_power_domain PD_TOP2 -elements { top1/q1 }
create_power_domain PD_TOP3 -elements { top1/ }
...
set_domain_supply_net PD_TOP -primary_power_net PD_TOP_primary_power -
primary_ground_net GND_net
set_domain_supply_net PD_TOP2 -primary_power_net PD_TOP_primary_power2 -
primary_ground_net GND_net
set_domain_supply_net PD_TOP3 -primary_power_net PD_TOP_primary_power3 -
primary_ground_net GND_net
create_power_switch PD_TOP_sw \
    -domain PD_TOP \
    -output_supply_port { out_sw_PD_TOP PD_TOP_primary_power } \
    -input_supply_port { in_sw_PD_TOP VDD_net } \
    -control_port { ctrl_sw_PD_TOP pg } \
    -on_state { normal_working in_sw_PD_TOP {ctrl_sw_PD_TOP } } \
    -off_state { off_state {!ctrl_sw_PD_TOP} }
...
set_retention PD_TOP_retention3 -domain PD_TOP -retention_power_net
VDD_net -elements { top1/bot1/tdsig[0][1] } -no_retention
set_retention_control PD_TOP_retention3 -domain PD_TOP -save_signal { ret3
posedge } -restore_signal { ret1 negedge }

set_retention PD_TOP_retention2 -domain PD_TOP -retention_power_net
VDD_net -elements { top1/bot1/tdsig }
set_retention_control PD_TOP_retention2 -domain PD_TOP -save_signal { ret2
posedge } -restore_signal { ret1 negedge }

set_retention PD_TOP_retention1 -domain PD_TOP -retention_power_net
VDD_net -elements { top1/bot1/tdsig[0][0] }
set_retention_control PD_TOP_retention1 -domain PD_TOP -save_signal { ret1
posedge } -restore_signal { ret1 negedge }
...

```

As a result of this UPF command, no retention strategy is applied to:

top1/bot1/tdsig[0][1]

## Reports

```
report.pa.txt
-----
Power Domain: PD_TOP, File: ./src/bitwise_1/test.upf(7) .
  Creation Scope: /tb1
  Primary Supplies:
    power : /tb1/PD_TOP_primary_power
    ground : /tb1/GND_net
  Power Switch: PD_TOP_sw, File: ./src/bitwise_1/test.upf(34) .
    Output Supply port:
      out_sw_PD_TOP(/tb1/PD_TOP_primary_power)
    Input Supply ports:
      1. in_sw_PD_TOP(/tb1/VDD_net)
  Control Ports:
    1. ctrl_sw_PD_TOP(/tb1/pg)
  Switch States:
    1. normal_working(ON) : (ctrl_sw_PD_TOP )
    2. off_state(OFF) : (!ctrl_sw_PD_TOP)
  Retention Strategy: PD_TOP_retention3, File: ./src/bitwise_1/
test.upf(58).
    Retention Supplies:
      power : /tb1/VDD_net
      ground : /tb1/GND_net
    No Retention
    Retention SAVE (/tb1/ret3), Retention Sense (posedge)
    Retention RESTORE (/tb1/ret1), Retention Sense (negedge)
  Retention Strategy: PD_TOP_retention2, File: ./src/bitwise_1/
test.upf(61).
    Retention Supplies:
      power : /tb1/VDD_net
      ground : /tb1/GND_net
    Retention SAVE (/tb1/ret2), Retention Sense (posedge)
    Retention RESTORE (/tb1/ret1), Retention Sense (negedge)
  Retention Strategy: PD_TOP_retention1, File: ./src/bitwise_1/
test.upf(64).
    Retention Supplies:
      power : /tb1/VDD_net
      ground : /tb1/GND_net
    Retention SAVE (/tb1/ret1), Retention Sense (posedge)
    Retention RESTORE (/tb1/ret1), Retention Sense (negedge)

-----
Power Domain: PD_BOT, File: ./src/bitwise_1/test.upf(70) .
...
```

## **-use\_retention\_as\_primary**

This option powers the storage element and the output drivers of the register using the retention supply.

## Example

```
upf_version 2.0
set_scope tb
create_power_domain pd_aon -include_scope
...
connect_supply_net vdd_net -ports { vdd_port }
connect_supply_net gnd_net -ports { gnd_port }
create_supply_set pd_aon_ss \
    -function { power vdd_net } \
    -function { ground gnd_net }
...
#####
# Retention Strategy for pd
#####
set_retention pd_retention1 -domain pd -save_signal { ret posedge
} -restore_signal { ret negedge } -elements {top_vh} -
use_retention_as_primary
map_retention_cell pd_retention1 -domain pd -lib_model_name
upf_retention_ret -lib_cell_type FF_CKHI
associate_supply_set pd_aon_ss -handle pd.pd_retention1.supply
...
```

## Report

```
report.de.txt

-----
----- ModelSim Power Aware Report File -----
-----

Total ( tb )
  upf_retention_ret # 2
  NPM_FF # 1
  NPM_LA # 3
  OUTPUT # 3
-----

pd sub_total ( /tb/top_vh /tb/top_vl )
  upf_retention_ret # 1
  /tb/top_vl/q_regvl 1
  NPM_LA # 2
  /tb/top_vh/q_latvh 1
  /tb/top_vl/q_latvl 1
  OUTPUT # 2
  /tb/top_vh/q_combvh 1
  /tb/top_vl/q_combvl 1
-----

pd.pd_retention1.use_retention_as_primary sub_total ( /tb/top_vh )
  upf_retention_ret # 1
  /tb/top_vh/q_regvh 1
-----

pd_aon sub_total ( /tb/top_aon )
  NPM_FF # 1
  /tb/top_aon/q_regvl 1
  NPM_LA # 1
  /tb/top_aon/q_latvl 1
  OUTPUT # 1
  /tb/top_aon/q_combvl 1
-----

-- NPM_FF => Denotes all Non Power Management Flip Flops of a Power
Domain.
-- NPM_LA => Denotes all Non Power Management Latches of a Power Domain.
-- OUTPUT => Denotes all outputs and power signals, which are not
sequential elements, of a Power Domain.
```

# Chapter 4

## Power Aware Reports

---

Power Aware simulation generates various reports that you use to analyze and validate the power intent of your design.

<b>Generating Reports for Power Aware .....</b>	<b>93</b>
<b>Power Aware Reports Reference .....</b>	<b>94</b>

## Generating Reports for Power Aware

Use the `-pa_genrpt` argument with the `vopt` command to generate the Power Aware reports.

### Procedure

1. Add the following arguments to your “`vopt`” command:
  - **-pa\_genrpt** — Generates the Power Aware reports listed in “[Power Aware Reports Reference](#)”.
    - If you do not specify a value to the `-pa_genrpt` argument, all reports are generated.
    - To specify more than one value to the `-pa_genrpt` argument, use the plus sign (+) operator between the values. For example:

```
vopt -pa_genrpt=pa+de+cell
```
  - **-pa\_reportdir** — (optional) Changes the default location of where the reports are saved.
    - When you run the `vopt` command to generate the reports, the default location is the `pa_reports/` directory in the current working directory. To change the location where report files are saved, add the following argument to your `vopt` command:

```
vopt -pa_reportdir <pathname>
```

2. Execute your `vopt` command to generate the Power Aware reports.
3. (optional) To generate reports at the `vsim` prompt, enter the “`pa report`” command.

---

#### Note



At the `vsim` prompt, the reports are generated only when the design is loaded for simulation.

---

# Power Aware Reports Reference

This section contains the details of various Power Aware reports that you use to analyze your Power Aware simulation.

The name of the report is presented as commented text at the top of the report. Comments are marked with double hyphens (--) at the beginning of a line.

**Table 4-1. Power Aware Reports**

Report	Description
Architecture Report	The architecture report contains information related to the Power Aware architecture that results from the power intent defined in the UPF file, when applied to the design.
Ack Port Driver Report	The ack port driver report contains information related to the acknowledge (ack) port drivers of the power switch cells.
Connection Report	The connection report contains information related to the connections between your UPF, HDL, and Liberty files.
Design Element Report	The design element report contains information related to the elements present in your design, and the corresponding Power Aware information.
Dynamic UPF Report	The dynamic UPF report contains information related to the power domain status, and controls of various UPF objects such as isolation, retention, and power switch.
Macro Cell Report	The macro cell report contains information related to the macro cells present in your Power Aware design.
Missing Liberty Cell Report	The missing liberty cell report contains a list of cells that do not have any Liberty cell definition.
Multi-Rail Cell Report	The multi-rail cell report contains a list of multi-rail cells that do not have any explicit UPF connections.
Non-Retention Synchronous Flop Report	The non-retention synchronous flop report contains a list of hierarchical paths of flip-flops that are non-retention in nature and do not have an asynchronous control signal.
Power Cell Report	The power cell report contains a list of unconnected cells that the simulator connects to the always-on power supply. These cells are either present inside a macro cell, or are level shifter cells.
PST Analysis Report	The PST analysis report contains information related to the PSTs in your design, including the composition details.
Source Sink Path Report	The source sink path report contains information related to the source sink-paths that are analyzed for static checks.

**Table 4-1. Power Aware Reports (cont.)**

<b>Report</b>	<b>Description</b>
<a href="#">Static Check Report</a>	The static check report contains information related to the static checks performed on your design.
<a href="#">Supply Network Initialization Report</a>	The supply network initialization report contains initial value of the UPF input supply port, output supply port (if the port has no driver), default isolation and retention supply set of a power domain, and switch supply set of a power domain. These initial values are set by the simulator.
<a href="#">Unassociated Cell Report</a>	The unassociated cell report contains a list of Power Aware cells that are not associated with any UPF strategy.

## Architecture Report

Filename: *report.pa.txt*

vopt argument: -pa\_genrpt=pa [+b]

The architecture report contains information related to the Power Aware architecture that results from the power intent defined in the UPF file, when applied to the design.

---

### Note

 Specify the argument value “b” with “pa” in the -pa\_genrpt argument to include bitwise expanded information in the report:

---

```
vopt -pa_genrpt=pa+b
```

---

The architecture report contains the following sections:

- [Power Domain](#) (including supplies)
- [Power Switch](#) (including supplies)
- [Retention Strategy](#) (including supplies)
- [Isolation Strategy](#) (including supplies)
- [Level Shifter Strategy](#)
- [Power State Table](#)
- [Power State Information](#)

For more information, refer to the “[Example of the Architecture Report](#)”.

## Power Domain

The Power Domain section contains the details of the power domain, such as the name of the power domain, reference to the UPF file where you create the power domain, creation scope, primary supplies, and power domain status.

### Format

```
Power Domain: /tb/T1/PD_1, File: ./src/test.upf(26).
Creation Scope: /tb/T1
```

**Extents:**

1. Instance : /tb/T1/G1
2. Instance : /tb/T1/G1/A\_mux, Lower Boundary
3. Instance : /tb/T1/G1/B\_f, Lower Boundary
4. Instance : /tb/T1/G1/B\_mux, Lower Boundary

**Primary Supplies:**

```
power : /tb/T1/VDD_1_net
ground : /tb/T1/GND_1_net
```

**Supply Set handles:**

1. primary: /tb/T1/PSET\_1

Functions:

1. power : /tb/T1/VDD\_1\_net
2. ground : /tb/T1/GND\_1\_net

**PD Status:** Switchable, Reason: Primary supplies start with default ON and supply\_off routine found at ./src/tb.v(78).

**Table 4-2. Power Domain Section Fields**

Field	Description
Power Domain	Name of the power domain.
File	UPF file where you create the power domain, preceded by its path, and the line number.
Creation Scope	Full hierarchical path of the creation scope.
Extents	Details of the extents and lower boundary information.
Primary Supplies	Details of the primary supplies of the power domain. When a power domain is not connected to a primary supply, a warning message, similar to the following, is displayed at the vopt command line, and that domain is treated as always-on:  ** Warning: test.upf(7): (vopt-9665) Power domain: 'pd' created in scope '/top_v1' has unassociated primary power/ground supply.
Supply Set handles	Details of the supply set handles associated with the power domain, and the absolute hierarchical path of the supply set associated with them. The supply set functions and the associated supply nets are listed below the handle name. If a supply set handle is left unassociated with a supply set or supply net, the report displays <Anonymous> in the relevant field.

**Table 4-2. Power Domain Section Fields (cont.)**

Field	Description
PD Status	Details of the power domain status.

## Power Switch

The Power Switch section contains the details of the power switch, such as the name of the power switch, reference to the UPF file where you create the power switch, input, output and control ports of the switch, switch states, and switch instances.

### Format

```

Power Switch: pd_sw, File: ./src/test.upf(35).
  Output Supply port:
    out_sw_pd(/tb/top/pd_pwr)
  Input Supply ports:
    1. in_sw_pd1(/tb/top/V_pd_net1)
    2. in_sw_pd2(/tb/top/V_pd_net2)
  Control Ports:
    1. ctrl_sw_pd1(/tb/top/pwr1)
    2. ctrl_sw_pd2(/tb/top/pwr2)
  Switch States:
    1. normal_working2(ON) : ( (ctrl_sw_pd2 && !ctrl_sw_pd1) )
    2. normal_working1(ON) : ( (ctrl_sw_pd1 && !ctrl_sw_pd2) )
    3. off_state(OFF) : (!ctrl_sw_pd1 && !ctrl_sw_pd2)
  Switch Instances:
    1. /tb/dut/SW_INST_1
    2. /tb/dut/SW_INST_2

```

**Table 4-3. Power Switch Section Fields**

Field	Description
Power Switch	Name of the power switch.
File	UPF file where you create the power switch, preceded by its path, and the line number.
Output Supply port	Details of the output supply port of the power switch in the following format: <code>&lt;port_name&gt; (&lt;externally_connected_net&gt;)</code>
Input Supply ports	Details of the input supply ports of the power switch.
Control Ports	Details of the control supply ports of the power switch.
Switch States	Details of the states of the power switch in the following format: <code>&lt;state_name&gt; (&lt;switch state&gt;) : &lt;Boolean_expression&gt;</code>

**Table 4-3. Power Switch Section Fields (cont.)**

Field	Description
Switch Instances	<p>Details of the power switch instances in the design associated with the UPF switch command.</p> <p> <b>Note:</b> The Switch Instances field is available only when you provide the following argument to the vopt command:</p> <pre>vopt -pa_enable=detectsw</pre>

## Retention Strategy

The Retention Strategy section contains the details of the retention strategy, such as the name of the retention strategy, reference to the UPF file where you define the retention strategy, retention supplies, and retention control signals.

### Format

```

Retention Strategy: pd_retention, File: ./src/test.upf(39).
Retention Supplies:
    power : /tb/SEC_FF_net
    ground : /tb/GND_FF_net
Supply Set handles:
    1. supply: PD_FF.RET1.supply
        Functions:
            1. power : /tb/SEC_FF_net
            2. ground : /tb/GND_FF_net
Retention SAVE (/tb/top/ret), Retention Sense (posedge)
Retention RESTORE (/tb/top/ret), Retention Sense (negedge)
Retained Signals:
    1. Scope: /tb/top_vh, File: reg_vh.vhdl(12)
        Model: Default UPF Retention
            1. /tb/top_vh/q_regvh
            2. /tb/top_vh/q_latvh
Signals with Retention cells:
    1. Signal : /tb/top/out1[0], retention cell : /tb/top/inst_ff

```

**Table 4-4. Retention Strategy Section Fields**

Field	Description
Retention Strategy	Name of the retention strategy.
File	UPF file where you define the retention strategy, preceded by its path, and the line number.
Retention Supplies	Details of the retention supplies with the full hierarchical path of the supply nets.
Supply Set handles	Supply set handles of the retention supply. The supply set functions and the associated supply nets are listed below the handle name.

**Table 4-4. Retention Strategy Section Fields (cont.)**

Field	Description
Retention SAVE	Name of the retention control signals.
Retention RESTORE	
Retention Sense	Retention sense of the retention control signals.
Retained Signals	Details of the signals on which retention is to be applied.
Signals with Retention cells	Details of the signals on which retention is already applied.

## Isolation Strategy

The Isolation Strategy section contains the details of the isolation strategy, such as the name of the isolation strategy, reference to the UPF file where you define the isolation strategy, isolation supplies, and isolation control signals.

### Format

```

Isolation Strategy: pd_isolation, File: ./src/test.upf(45).
Isolation Supplies:
    power : /tb/SEC_LOG_net
    ground : /tb/GND_LOG_net
Supply Set handles:
    1. isolation_supply_set: PD_LOG.ISO1.isolation_supply_set
        Functions:
            1. power : /tb/SEC_LOG_net
            2. ground : /tb/GND_LOG_net
Isolation Control (/tb/top/iso), Isolation Sense (HIGH),
    Clamp Value (1), Location (parent)
Signals with default isolation cells:
    1. Signal : /mem_test/top/id_in, isolation cell :
        /mem_test/top/id_in_UPF_ISO
    2. Signal : /mem_test/top/mem_r/ar, isolation cell :
        /mem_test/top/mem_r/ar_UPF_ISO, Lower Boundary

```

**Table 4-5. Isolation Strategy Section Fields**

Field	Description
Isolation Strategy	Name of the isolation strategy.
File	UPF file where you define the isolation strategy, preceded by its path, and the line number.
Isolation Supplies	Details of the isolation supplies with the full hierarchical path of the supply nets.
Supply Set handles	Supply set handles of the isolation supply. The supply set functions and the associated supply nets are listed below the handle name.
Isolation Control	Control signal that triggers the isolation clamp value.

**Table 4-5. Isolation Strategy Section Fields (cont.)**

Field	Description
Isolation Sense	Sense of the control signal at which the clamp value is applied on the isolated cell.
Clamp Value	Clamp value of the isolation cell.
Location	Location of the isolation cell.
Signals with default isolation cells	Details of one of the following signals, which depends upon the option that you provide in the set_isolation command. <ul style="list-style-type: none"> <li>• Signals with -instance isolation cells</li> <li>• Signals with map_isolation_cell</li> <li>• Signals with default isolation cells</li> </ul>

## Level Shifter Strategy

The Level Shifter Strategy section contains the details of the level shifter strategy, such as the name of the level shifter strategy, and reference to the UPF file where you define the level shifter strategy.

### Format

```

Level Shifter Strategy: LS1, File: ./src/test.upf(110).
  Rule (low_to_high), Threshold (0), Applies_to (outputs), Location
  (parent).
    Level Shifted Candidate Ports:
      1. Signal : /tb/top/out1
    Signals with Level Shifter cells:
      1. Signal : /tb/top/out2[0], level shifter cell : /tb/top/ls_0
      2. Signal : /tb/top/out2[1], level shifter cell : /tb/top/ls_1

```

**Table 4-6. Level Shifter Strategy Section Fields**

Field	Description
Level Shifter Strategy	Name of the level shifter strategy
File	UPF file where you define the level shifter strategy, preceded by its path, and the line number
Rule	Shift direction of the level shifter cell
Threshold	Threshold value (in volts) of the level shifter cell
Applies_to	Value of the -applies_to option that you specify in the set_level_shifter command
Location	Location of the level shifter cell
Level Shifted Candidate Ports	Details of the signals on which a level shifter cell is to be inserted

**Table 4-6. Level Shifter Strategy Section Fields (cont.)**

Field	Description
Signals with Level Shifter cells	Details of the signals on which a level shifter cell is already inserted

## Power State Table

The Power State Table section contains the details of the PST, such as the name of the PST, creation scope, hierarchical paths of the supply nets, and list of possible states.

### Format

```
Pst MyPst, File: ./src/dut.upf(80).
Scope => /tb
Header ==>
    Reboot      : VSS          VDD_A0
    dut.upf(84): s_state     ram_s
    Sleep       dut.upf(85): r_state   ram_r

List of possible states on:
    VSS [ source supply port: VSS, File: ./src/dut.upf(12) ]
        1. s_state: 3.20
        2. r_state: 2.20
```

**Table 4-7. Power State Table Section Fields**

Field	Description
Pst	Name of the power state table.
File	UPF file where you define the power state table, preceded by its path, and the line number.
Scope	Hierarchical path of the scope where the PST is created.
Header	<p>PST header that contains the name of the supply nets or ports with their hierarchical path.</p> <p>The lines that follow the Header field are divided into the following columns:</p> <p>&lt;state_pst&gt; &lt;file(line_num)&gt; &lt;state_net1&gt; &lt;state_net2&gt;</p> <ul style="list-style-type: none"> <li>• &lt;state_pst&gt; — State of the PST.</li> <li>• &lt;file(line_num)&gt; — UPF file where you define the state of the PST, preceded by its path, and the line number.</li> <li>• &lt;state_net&gt; — State of the supply port or net.</li> </ul>
List of possible states on	List of the supply nets or ports with the voltage information.
source supply port	<p>Source supply port of the supply net or port specified in the Header.</p> <p>You specify the source supply port in the add_port_state command, or it is the supply net/port which is directly connected to the port on which you specify the add_port_state command.</p>

**Table 4-7. Power State Table Section Fields (cont.)**

Field	Description
File	UPF file where you define the source supply port, preceded by its path, and the line number.

## Power State Information

This section lists the power state information of the supply sets, power domains, composite domains, and power state groups.

### Format

```

Power state info of Supply set : 'IN_PD.primary'.
  1. Power state 'ON_MODE', File: ./src/test.upf(142).
    Simstate : NORMAL.
    Logic Expression : ((OUT_PD.primary == ON_MODE) &&
      (DELIVER_PD.primary == ON_MODE)).
    Supply Expression : ((VDD_08_net == `{FULL_ON, 0.80}) && (GND_net ==
      `{FULL_ON, 0.00})) .

Power state info of Power domain : 'PD_top'.
  1. Power state 'FULL_POWER', File: ./src/test.upf(100).
    Logic Expression : ((PSET_1 == ON) && (PSET_2 == ON)) .

```

**Table 4-8. Power State Information Section Fields**

Field	Description
Power state info of Supply set	Power state information of one of the following objects, which depends upon the option that you provide in the add_power_state command: <ul style="list-style-type: none"> <li>• Supply set</li> <li>• Power domain</li> <li>• Composite power domain</li> <li>• Power state group</li> </ul>
Power state	Name of the power state
File	UPF file where you define the power state, preceded by its path, and the line number
Simstate	Simstate of the supply set
Logic Expression	Boolean expression defined in terms of logic nets and/or power states of supply sets and/or power domains
Supply Expression	Boolean expression defined in terms of supply ports, supply nets, and/or supply set handle functions

## Example of the Architecture Report

```
-----
----- QuestaSim Power Aware Architecture Report File -----
-----
<... Header information removed ...>
-----
Power Domain: PD_1, File: test.upf(29).
  Creation Scope: /tb/inst1/GEN_REG_1/REGX
  Extents:
    1. Instance : /tb/inst1/GEN_REG_1/REGX
  Primary Supplies:
    power : /tb/inst1/GEN_REG_1/REGX/VDD_1_net
    ground : /tb/inst1/GEN_REG_1/REGX/GND_1_net
  Supply Set handles:
    1. primary: PD_1.primary
      Functions:
        1. power      : /tb/inst1/GEN_REG_1/REGX/VDD_1_net
        2. ground     : /tb/inst1/GEN_REG_1/REGX/GND_1_net
  PD Status: Always On, Reason: Primary supplies start with default ON.
-----

Power Domain: pd, File: test.upf(16).
  Creation Scope: /tb
  Primary Supplies:
    power : /tb/pd_pwr
    ground : /tb/G_pd_net

  Power Switch: pd_sw, File: test.upf(37).
    Output Supply port:
      out_sw_pd(/tb/pd_pwr)
    Input Supply ports:
      1. in_sw_pd(/tb/V_pd_net)
    Control Ports:
      1. ctrl_sw_pd(/tb/pwr)
    Switch States:
      1. normal_working(ON) : ( ctrl_sw_pd )
      2. off_state(OFF)   : (!ctrl_sw_pd)

  Retention Strategy: pd_retention, File: test.upf(41).
    Retention Supplies:
      power : /tb/V_pd_net
      ground : /tb/G_pd_net
    Retention SAVE (/tb/ret), Retention Sense (posedge)
    Retention RESTORE (/tb/ret), Retention Sense (negedge)
    Retained Signals:
      1. Scope: /tb/top_vh, File: reg_vh.vhdl(12)
         Model: upf_retention_ret
           1. /tb/top_vh/q_regvh
      2. Scope: /tb/top_vl, File: reg_vl.v(1)
         Model: upf_retention_ret
           1. /tb/top_vl/q_regvl

  Isolation Strategy: pd_isolation, File: test.upf(48).
    Isolation Supplies:
      power : /tb/V_pd_net
      ground : /tb/G_pd_net
    Isolation Control (/tb/iso), Isolation Sense (HIGH), Clamp Value
```

```
(1),
    Location (parent)
Isolated Signals:
  1. Signal : /tb/top_vh/q_combvh
  2. Signal : /tb/top_vh/q_latvh
  3. Signal : /tb/top_vh/q_regvh
  4. Signal : /tb/top_vl/q_combvl
  5. Signal : /tb/top_vl/q_latvl
  6. Signal : /tb/top_vl/q_regvl

Level Shifter Strategy: ls_PD_mid2, File: test.upf(102).
  Rule (high_to_low), Threshold (0), Applies_to (both), Location
  (automatic).

Pst PST, File: test.upf(112).
Scope => /tb/TOP
Header ==>          : tb_pow mid1_MAIN_NET   mid2_MAIN_NET
state1 test.upf(113)  : tb_nom v_nom           v_nom
state2 test.upf(114)  : TBoff   v_nom           v_nom
state3 test.upf(115)  : tb_nom Voff            Voff

List of possible states on:
  tb_pow [ source supply port: TB_PRI, File: test.upf(5) ]
    1. tb_nom: 4.80
    2. TBoff : OFF
  mid1_MAIN_NET [ source supply port: mid1_PRI, File: test.upf(27) ]
    1. v_nom: 5.00
    2. Voff : OFF
  mid2_MAIN_NET [ source supply port: mid2_PRI, File: test.upf(69) ]
    1. v_nom: 5.20
    2. Voff : OFF
-----
Power state info of Power domain : 'PD_top'.
  1. Power state 'HIBERNATE', File: test.upf(100).
    Logic Expression : ((PSET_1 == OFF) || (PSET_2 == OFF)).
  2. Power state 'POWER_DOWN', File: test.upf(100).
    Logic Expression : ((PSET_1 == OFF) && (PSET_2 == OFF)).
  3. Power state 'FULL_POWER', File: test.upf(100).
    Logic Expression : ((PSET_1 == ON) && (PSET_2 == ON)).

Power state info of Supply set : 'PSET_1'.
  1. Power state 'OFF', File: test.upf(105).
    Logic Expression : (!/tb/pwr).
    Supply Expression : ((VDD_1 == `{OFF}) || (GND_net == `{OFF})).
  2. Power state 'ON', File: test.upf(105).
    Logic Expression : /tb/pwr.
    Supply Expression : ((VDD_1 == `{FULL_ON, 1.50}) && (GND_net == `{FULL_ON, 0.00}))
  ...
  ...
```

## Related Topics

[Generating Reports for Power Aware](#)

## Ack Port Driver Report

Filename: *report.ack.driver.txt*

vopt argument: None

The ack port driver report contains information related to the acknowledge (ack) port drivers of the power switch cells.

### **Note**

 The simulator generates the report only when there is at least one power switch with an ack port in your design. If you do not specify the HDL driver of the ack port, the simulator does not report the ack port. In this case, the ack port is driven by the logic specified in the IEEE Std 1801.

### Format

```
Switch Cell : sw_1    File : ./src/case1/chip.upf(15)
  Ack Ports : CTL1OUT(/tb/chip/ack1) ,
  HDL driver : sw1(user_instance)   File: ./src/case1/chip.v(42)
```

**Table 4-9. Ack Port Driver Report Fields**

Field	Description
Switch cell	Name of the power switch
File	UPF file where you create the power switch, preceded by its path, and the line number
Ack Ports	Acknowledge (ack) port of the power switch
HDL driver	HDL driver of the acknowledge (ack) port
File	HDL file where you specify the HDL driver, preceded by its path, and the line number

### Example of the Ack Port Driver Report

```
----- QuestaSim Power Switch ACK port HDL driver Report File -----
```

```
Switch Cell : sw_1    File : ./src/case1/chip.upf(15)
Ack Ports :
  CTL1OUT(/tb/chip/ack1) , HDL driver : sw1(user_instance)  File: ./
  src/case1/chip.v(42)

Switch Cell : sw_2    File : ./src/case2/chip.upf(17)
Ack Ports :
  CTL2OUT(/tb/chip/ack2) , HDL driver : sw2(user_instance)  File: ./
  src/case2/chip.v(42)

Switch Cell : sw_3    File : ./src/case3/chip.upf(18)
Ack Ports :
  CTL3OUT(/tb/chip/ack3) , HDL driver : sw3(user_instance)  File: ./
  src/case3/chip.v(44)
...
```

## Related Topics

[Generating Reports for Power Aware](#)

# Connection Report

Filename: *report.connection.txt*

vopt argument: -pa\_genrpt=conn

The connection report contains information related to the connections between your UPF, HDL, and Liberty files.

The connection report contains the following sections:

- [Port or Net Connection](#)
- [Root Supply Connection](#)

For more information, refer to the “[Example of the Connection Report](#)”.

## Port or Net Connection

This section lists the port or net connections between your UPF, HDL, and Liberty files.

### Format

```
/tb/top/pwr [HDL3] primary_power (UPF2SV_LOGIC) <=
(SV_LOGIC) /tb/VDD [UPFSN2] primary_power1 ./src/test.vhd(11)
```

**Table 4-10. Port or Net Connection Section Fields**

Field	Description
</tb/top/pwr>	Hierarchical path to a port or net for the left-hand side of the connection.

**Table 4-10. Port or Net Connection Section Fields (cont.)**

Field	Description
HDL3	<p>One of the following identifiers, created by the simulator, for the left-hand side of the connection. The identifier is enclosed in brackets ([]).</p> <ul style="list-style-type: none"> <li>• UPFSN — UPF supply net</li> <li>• UPFSP — UPF supply port</li> <li>• UPFCP — UPF control port</li> <li>• UPFAP — UPF acknowledge port</li> <li>• UPFLP — UPF logic port</li> <li>• UPFLN — UPF logic net</li> <li>• HDL — HDL port or net</li> <li>• LIB — LIBERTY port or net</li> </ul>
<primary_power>	(optional) Power or ground name for the left-hand side of the connection.
<UPF2SV_LOGIC>	(optional) VCT used for the left-hand side of the connection, enclosed in parentheses.
<=	<p>One of the following values:</p> <ul style="list-style-type: none"> <li>• &lt;= — Left-hand side is driven by the right-hand side.</li> <li>• &lt;=&gt; — Inout connection between the two sides.</li> </ul>
<SV_LOGIC>	(optional) VCT used for the right-hand side of the connection, enclosed in parentheses.
</tb/VDD>	Hierarchical path to a port or net for the right-hand side of the connection.
UPFSN2	<p>Identifier created by the simulator for the right-hand side of the connection.</p> <p>The identifier is enclosed in brackets ([]), and can have values similar to the left-hand side of the connection.</p>
<primary_power1>	(optional) Power or ground name for the right-hand side of the connection.
<./src/test.vhd(11)>	File to which the connection is created, preceded by its relative path, and the line number.

## Root Supply Connection

This section lists the root supply, and the supplies that are connected to the root supply.

### Format

```

Root supply: /tb/VDD_TB_net [UPFSN4]
Connected Supplies:
    /tb/VDD_TB_port [UPFSP7]
```

**Table 4-11. Root Supply Connection Section Fields**

Field	Description
Root supply	(optional) Name of the root supply, followed by the identifier created by the simulator in brackets ([])
Connected Supplies	(optional) List of supplies connected to the root supply, followed by the identifier created by the simulator in brackets ([])

## Example of the Connection Report

```
-----
----- QuestaSim Power Aware Connection Report File -----
<.... Header information removed ...>
-----
/tb/top/inst_3/pwr [HDL3] primary_power (UPF2SV_LOGIC) <=
(SV_LOGIC) /tb/VDD_PD_1 [UPFSN2] primary_power1 ./src/test.vhd(11)
/tb/top_tb/inst_2/pwr [HDL4] primary_power (UPF2SV_LOGIC) <=
/tb/VDD_PD_1_net [UPFSN2] . ./src/test_50/elements.vhd(64)
/tb/GND [UPFSN4] <= /tb/GND_system [UPFSP2] . ./src/test02/test.upf(12)
/tb/VDD [UPFSN5] <= /tb/VDD_system [UPFSP3] . ./src/test02/test.upf(11)
/tb/dum_inst/GND_DUM_net [UPFSN5] <= /tb/dum_inst/GND_DUM_port [UPFSP8] .
/src/test/test.upf(40)
/tb/dum_inst/VDD_DUM_net [UPFSN6] <= /tb/dum_inst/VDD_DUM_port [UPFSP9] .
/src/test/test.upf(39)
...
-----Root Supply Connections-----
Root supply: /tb/VDD_TB_net [UPFSN4]
    Connected Supplies:
        /tb/VDD_TB_port [UPFSP7]
        /tb/VDD_TB_port1 [UPFSP10]
Root supply: /tb/GND_TB_net [UPFSN3]
    Connected Supplies:
        /tb/GND_TB_port [UPFSP6]
Root supply: /tb/dum_inst/net1 [UPFSN7]
    Connected Supplies:
        /tb/dum_inst/out_port [UPFSP2]
Root supply: /tb/dum_inst/VDD_DUM_net [UPFSN6]
    Connected Supplies:
        /tb/dum_inst/VDD_DUM_port [UPFSP9]
Root supply: /tb/dum_inst/GND_DUM_net [UPFSN5]
    Connected Supplies:
        /tb/dum_inst/GND_DUM_port [UPFSP8]
Root supply: /tb/VDD_PD_1_net [UPFSN2]
...
```

## Related Topics

[Generating Reports for Power Aware](#)

# Design Element Report

Filename: *report.de.txt*

vopt argument: -pa\_genrpt=de[+b]

The design element report contains information related to the elements present in your design, and the corresponding Power Aware information.

## **Note**

 Specify the argument value “b” with “de” in the -pa\_genrpt argument to include bitwise expanded information in the report:

---

```
vopt -pa_genrpt=de+b
```

---

The content in the report is listed in single-line records that contain keywords indicating the specific Power Aware characteristics.

The design element report contains the following sections:

- [Design Element Scope](#)
- [Corrupted Signal](#)
- [State Element](#)

For more information, refer to the “[Example of the Design Element Report](#)”.

## Design Element Scope

This section lists all scopes in the design that belong to a power domain.

### Format

```
pd_aon: {Path1} = scope /tb/top_h1 <>
--: {Path4} = scope /tb/top_vl<>
```

**Table 4-12. Design Element Scope Section Fields**

Field	Description
<pd_aon>	Name of the power domain. When the power domain is represented by two dashes ( -- ) it indicates that the element was excluded using the exclusion rules.
<Path1>	Identifier for the hierarchical path of the scope, enclosed in braces ({}), which is used in the rest of the report.
scope	Keyword to identify the current line as a design element scope, which assists in using a grep search.
</tb/top_h1>	Full hierarchical path of the design element (its scope).

**Table 4-12. Design Element Scope Section Fields (cont.)**

Field	Description
<>	(optional) Identifier for a scope that is present at the power domain boundary.

## Corrupted Signal

This section lists the corrupted signals that belong to a power domain. The signals inside the power domain are corrupted when the supply of the power domain is switched off.

### Format

```
pd: {Path3}/q_combvh
--: {Path4}/q_combvl
```

**Table 4-13. Corrupted Signal Section Fields**

Field	Description
<pd>	Name of the power domain. When the power domain is represented by two dashes ( -- ) it indicates that the element was excluded using the exclusion rules.
<Path3>	Identifier for the hierarchical path of the scope, which is used in the rest of the report.
<q_combvh>	Name of the signal.

## State Element

This section lists the signals in the design that act as state elements. These signals are listed in the same way as “[Corrupted Signal](#)”, along with a special keyword that identifies the kind of state element.

### Format

```
pd_aon: {Path1}/q_combvl NPM_LA
--: {Path2}/q_latvl UDP_LA
```

**Table 4-14. State Element Section Fields**

Field	Description
<pd_aon>	Name of the power domain. When the power domain is represented by two dashes ( -- ) it indicates that the element was excluded using the exclusion rules.
<Path1>	Identifier for the hierarchical path of the scope, which is used in the rest of the report.
<q_combvl>	Name of the signal.

**Table 4-14. State Element Section Fields (cont.)**

Field	Description
NPM_LA	<p>One of the following values, indicating the type of state element or a retention element:</p> <ul style="list-style-type: none"> <li>• NPM_LA — Non-retention Latch</li> <li>• NPM_FF — Non-retention Flip-Flop</li> <li>• MEM — Non-retention Memory</li> <li>• UDP_LA — Non-retention UDP Latch</li> <li>• UDP_FF — Non-retention UDP Flip-Flop</li> <li>• R — Retention element</li> <li>• &lt;no keyword&gt; — Combinatorial logic</li> </ul>

## Extracting Information From a Design Element Report

The structure and keywords of a design element report enable you to extract the information related to the power intent easily and effectively using any string matching utility, such as the grep command.

The following examples show how to use the grep command to extract specific details of your power intent.

### List all instances that belong to a power domain

Command:

```
grep pd_aon report.de.txt | grep scope
```

Output:

```
pd_aon: {Path1} = scope /tb/top_aon
```

### List all retention signals

Command:

```
grep " R" report.de.txt | grep -v "^--"
```

Output:

```
pd: {Path2}/q_regvh R
pd: {Path3}/q_regvl R
```

### Expand the path identifier

Command:

```
grep "{Path2} =" report.de.txt
```

Output:

```
pd: {Path2} = scope /tb/top_vh <>
```

**Search a particular signal for power intent**

Command 1:

```
grep "/tb/top_vh\>" report.de.txt
```

Output 1:

```
pd: {Path2} = scope /tb/top_vh <>
```

Command 2:

```
grep "{Path2}/q_regvh" report.de.txt
```

Output 2:

```
pd: {Path2}/q_regvh R
```

(implies that /tb/top\_vh/q\_regvh is a retention register inside power domain pd)

**List the instances at the power domain boundary**

Command:

```
grep "<>" report.de.txt
```

Output:

```
pd: {Path2} = scope /tb/top_vh <>
pd: {Path3} = scope /tb/top_vl <>
```

## Example of the Design Element Report

```
-----  
----- QuestaSim Power Aware Design Element Report File -----  
-----  
----- <... Header information removed ...>  
-----  
pd_aon: {Path1} = scope /tb  
pd_aon: {Path2} = scope /tb/top_aon  
pd: {Path3} = scope /tb/top_vh <>  
--: {Path4} = scope /tb/top_vl <>  
  
pd_aon: {Path1}/tie_high_low  
pd_aon: {Path1}/q_combaon  
pd_aon: {Path1}/q_latchaon  
pd_aon: {Path1}/q_regaon  
pd_aon: {Path1}/q_combvh  
pd_aon: {Path1}/q_latchvh  
pd_aon: {Path1}/q_regvh  
pd_aon: {Path1}/q_combvl  
pd_aon: {Path1}/q_latchvl  
pd_aon: {Path1}/q_regvl  
pd_aon: {Path1}/ret  
pd_aon: {Path1}/iso  
pd_aon: {Path1}/pwr  
pd_aon: {Path1}/set  
pd_aon: {Path1}/rst  
pd_aon: {Path1}/clk  
pd_aon: {Path1}/d  
pd_aon: {Path2}/q_combvl  
pd_aon: {Path2}/q_regvl NPM_FF  
pd_aon: {Path2}/q_latvl NPM_LA  
pd: {Path3}/q_combvh  
pd: {Path3}/q_regvh R  
pd: {Path3}/q_latvh NPM_LA  
--: {Path4}/q_combvl  
--: {Path4}/q_regvl R  
--: {Path4}/q_latvl NPM_LA  
...  
...
```

## Related Topics

[Generating Reports for Power Aware](#)

## Dynamic UPF Report

Filename: none (Transcript window only)

vopt argument: -pa\_genrpt=ud

The dynamic UPF report contains information related to the power domain status, and controls of various UPF objects such as isolation, retention, and power switch.

**Note**

 The dynamic UPF report is displayed as messages in the Transcript window, and these messages are logged in the *transcript* file.

---

The dynamic UPF messages contain the following information:

- Power domain status
- Time and polarity of controls, such as control port of a power switch, retention save and restore signals, and isolation enable signal
- supply\_on, supply\_off, and partial\_on information with the filename and line number
- Supply net and supply port toggle information

**Format**

`<severity>: <message_num> <mnemonic>: Time: <time>, <description>`

**Table 4-15. Dynamic UPF Report Fields**

Field	Description
<code>&lt;severity&gt;</code>	Severity of the message that the simulator displays in the transcript window
<code>&lt;message_num&gt;</code>	Vsim message number, enclosed in parentheses
<code>&lt;mnemonic&gt;</code>	Mnemonic of the message
<code>&lt;time&gt;</code>	Time at which the condition pertaining to the message occurs
<code>&lt;description&gt;</code>	Details of the message

## Example of the Dynamic UPF Report

```
# ** Note: (vsim-8916) QPA_UPF_RET_CTRL_INFO: Time: 15 ns, Retention Strategy (PD_BOT_retention), Retention SAVE (/tb/ret_bot_reg), Retention Sense (posedge), switched to polarity (1). Power Domain: PD_BOT

# ** Note: (vsim-8902) QPA_PD_STATUS_INFO: Time: 20 ns, Power domain 'PD_BOT' is powered down.

# ** Note: (vsim-8913) QPA_UPF_SWITCH_CTRL_INFO: Time: 20 ns, Power Switch (PD_BOT_sw), Control Signal (/tb/pg_bot), switched to polarity (0), Power Switch state (OFF). Power Domain: PD_BOT

# ** Note: (vsim-8902) QPA_PD_STATUS_INFO: Time: 35 ns, Power domain 'PD_BOT' is powered up.

# ** Note: (vsim-8913) QPA_UPF_SWITCH_CTRL_INFO: Time: 35 ns, Power Switch (PD_BOT_sw), Control Signal (/tb/pg_bot), switched to polarity (1), Power Switch state (ON). Power Domain: PD_BOT

# ** Note: (vsim-8916) QPA_UPF_RET_CTRL_INFO: Time: 40 ns, Retention Strategy (PD_BOT_retention), Retention RESTORE (/tb/ret_bot_reg), Retention Sense (negedge), switched to polarity (0). Power Domain: PD_BOT

# ** Note: (vsim-8914) QPA_UPF_ISO_CTRL_INFO: Time: 65 ns, Isolation Strategy (PD_BOT_isolation), Isolation Control (/tb/iso_bot), Isolation Sense (HIGH), switched to polarity (1). Power Domain: PD_BOT

# ** Note: (vsim-8902) QPA_PD_STATUS_INFO: Time: 70 ns, Power domain 'PD_BOT' is powered down.

# ** Note: (vsim-8913) QPA_UPF_SWITCH_CTRL_INFO: Time: 70 ns, Power Switch (PD_BOT_sw), Control Signal (/tb/pg_bot), switched to polarity (0), Power Switch state (OFF). Power Domain: PD_BOT

# ** Note: (vsim-8902) QPA_PD_STATUS_INFO: Time: 85 ns, Power domain 'PD_BOT' is powered up.

# ** Note: (vsim-8913) QPA_UPF_SWITCH_CTRL_INFO: Time: 85 ns, Power Switch (PD_BOT_sw), Control Signal (/tb/pg_bot), switched to polarity (1), Power Switch state (ON). Power Domain: PD_BOT

# ** Note: (vsim-8914) QPA_UPF_ISO_CTRL_INFO: Time: 90 ns, Isolation Strategy (PD_BOT_isolation), Isolation Control (/tb/iso_bot), Isolation Sense (HIGH), switched to polarity (0). Power Domain: PD_BOT

# ** Note: (vsim-8962) QPA_UPF_PACKAGE_FUNC: Time: 50 ns, Supply OFF applied on Signal:'dut/AVDD' , File:tb.sv(22)

# ** Note: (vsim-8960) QPA_UPF_SUPPLY_CHANGE_INFO: Time: 80 ns, Supply net '/tb/dut/AVSS' toggled to '{FULL_ON 0 V}' from '{OFF 0 V}'.
# File: test1.upf, Line: 5

...
```

## Related Topics

[Generating Reports for Power Aware](#)

## Macro Cell Report

Filename: *report.cell.txt*

vopt argument: -pa\_genrpt=cell

The macro cell report contains information related to the macro cells present in your Power Aware design.

### Format

```
Module : analog
  Verification Model File : ./src/case2/tb.sv(52)
  Liberty Model File : src/case2/analog.lib(2)
  is_macro_cell : true

  Ports :
    1. in, File : ./src/case2/tb.sv(53)
      direction : input
      related_ground_pin : DGND, File : src/case2/analog.lib(18)
      ...

  Instances :
    1. /tb/dut/ab1 ( Parent:top )
      Corruption mode : (--)
      UPF_is_macro_cell : true, File : ./src/case2/test.upf(16)
      ...
```

**Table 4-16. Macro Cell Report Fields**

Field	Description
Module	Name of the macro cell module
Verification Model File	HDL file where you specify the macro cell module, preceded by its path, and the line number
Liberty Model File	Liberty file where you define the macro cell module, preceded by its path, and the line number
is_macro_cell	Identifier for the type of cell: <ul style="list-style-type: none"><li>• is_macro_cell</li><li>• is_level_shifter</li><li>• is_isolation_cell</li><li>• retention_cell</li><li>• is_repeater</li><li>• upf_is_hard_macro</li><li>• upf_is_soft_macro</li></ul>
Ports	Port details of the macro cell module
Instances	Instance details of the macro cell module

## Example of the Macro Cell Report

```
-----
----- QuestaSim Power Aware Cell Report File -----
<... Header information removed ...>
-----

1. Module : analog
   Verification Model File : ./src/case2/tb.sv(52)
   Liberty Model File : src/case2/analog.lib(2)
   is_macro_cell : true

   Ports :
   1. in, File : ./src/case2/tb.sv(53)
      direction : input
      related_ground_pin : DGND, File : src/case2/analog.lib(18)
      related_power_pin : DVDD, File : src/case2/analog.lib(17)
      a : val, File : ./src/case2/test.upf(13)
   2. out, File : ./src/case2/tb.sv(54)
      direction : output
      related_ground_pin : DGND, File : src/case2/analog.lib(24)
      related_power_pin : DVDD, File : src/case2/analog.lib(23)
      power_down_function : (((~DVDD) | (~AVDD)) | DGND) | AGND),
      File : src/case2/analog.lib(22)
   3. DVDD, File : src/case2/analog.lib(3)
      pg_type : primary_power, File : src/case2/analog.lib(4)
      direction : input, File : src/case2/analog.lib(3)
   ...

   Instances :
   1. /tb/dut/ab1 ( Parent:top )
      Corruption mode : (--)
      UPF_is_macro_cell : true, File : ./src/case2/test.upf(16)

      Ports :
      1. in
         Corruption mode : related supply
      2. out
         Corruption mode : power_down_function
      3. AGND
         Connection : /tb/dut/VSS , VCT : QPA_CONNECT_UPF_2_HDL_GND
      ...

   2. /tb/dut/ab2 ( Parent:top )
      Corruption mode : (--)
      UPF_is_macro_cell : true, File : ./src/case2/test.upf(16)

      Ports :
      1. in
         Corruption mode : related supply
      2. out
         Corruption mode : power_down_function
      3. AGND
         Connection : /tb/dut/VSS , VCT : QPA_CONNECT_UPF_2_HDL_GND
      ...

```

## Related Topics

[Generating Reports for Power Aware](#)

# Missing Liberty Cell Report

Filename: *report.missingliberty.txt*

vopt argument: {[ -pa\_libertyfiles=<liberty\_filename> ] |  
[-pa\_loadlibertydb=<database\_pathname>]}

The missing liberty cell report contains a list of cells that do not have any Liberty cell definition.

## Format

```
analog1, File: ./src/pa_cell_conn/tb.sv(73)
```

**Table 4-17. Missing Liberty Cell Report Fields**

Field	Description
analog1	Name of the missing cell module
File	HDL file where you define the missing cell module, preceded by its path, and the line number

## Example of the Missing Liberty Report

```
-----  
----- QuestaSim Missing Liberty Report File -----  
-----  
analog1, File: ./src/pa_cell_conn/tb.sv(73)  
analog2, File: ./src/pa_cell_conn/tb.sv(79)  
...
```

## Related Topics

[Generating Reports for Power Aware](#)

# Multi-Rail Cell Report

Filename: *report.multiRail.cells.txt*

vopt argument: None

The multi-rail cell report contains a list of multi-rail cells that do not have any explicit UPF connections.

---

### Note

 The multi-rail cell report is generated only when your design has at least one multi-rail cell that does not have any explicit UPF connection.

---

### Format

```

PA multi-rail cell Module : analog
Verification Model File : ./src/tb.sv(40)
Liberty Model File : src/analog.lib(3)

Instances :
    ab, File: ./src/tb.sv(42)

```

**Table 4-18. Multi-Rail Cell Report Fields**

Field	Description
PA multi-rail cell Module	Name of the multi-rail cell module
Verification Model File	HDL file where you define the multi-rail cell module, preceded by its path, and the line number
Liberty Model File	Liberty file where you define the multi-rail cell module, preceded by its path, and the line number
Instances	Instance of the multi-rail cell module
File	HDL file where you specify the module instance, preceded by its path, and the line number

### Example of the Multi-Rail Cell Report

```

-----
--QuestaSim Report File for multi-rail cells without explicit connections-
-----
PA multi-rail cell Module : analog
Verification Model File : ./src/tb.sv(40)
Liberty Model File : src/analog.lib(3)

Instances :
    ab, File: ./src/tb.sv(42)
...

```

### Related Topics

[Generating Reports for Power Aware](#)

## Non-Retention Synchronous Flop Report

Filename: *report.nretsyncff.txt*

vopt argument: -pa\_genrpt=npu

The non-retention synchronous flop report contains a list of hierarchical paths of flip-flops that are non-retention in nature and do not have an asynchronous control signal.

**Note**

 The non-retention synchronous flop report is generated only when your design has at least one flip-flop that is non-retention in nature, and does not have an asynchronous control signal.

---

## Example of the Non-Retention Synchronous Flop Report

```
----- QuestaSim Power Aware Report File (NRS Flops) -----  
-----  
The list of Non-Retention Flops without Async control:  
/interleaver_tester/dut/i0/do_reg  
/interleaver_tester/dut/m1/Q_tmp  
/interleaver_tester/dut/m2/Q_tmp  
/interleaver_tester/dut/m3/Q_tmp  
/interleaver_tester/dut/m4/Q_tmp  
...
```

## Related Topics

[Generating Reports for Power Aware](#)

## Power Cell Report

Filename: *report.powercells.txt*

vopt argument: {[ -pa\_enable=conninsidemacro] | [-pa\_enable=autoconnls]}

The power cell report contains a list of unconnected cells that the simulator connects to the always-on power supply. These cells are either present inside a macro cell, or are level shifter cells.

**Note**

 The power cell report is generated only when there is at least one unconnected cell present inside a macro cell or an unconnected level shifter cell.

---

The power cell report contains the following sections:

- [Power Cell](#)
- [Level Shifter Cell](#)

For more information, refer to the “[Example of the Missing Liberty Report](#)”.

## Power Cell

This section lists the unconnected cells present inside the macro cells, which the simulator connects to the always-on power supply. This section is listed in the report only when your

design has at least one unconnected cell present inside the macro cell, and you specify the following argument in vopt:

```
vopt -pa_enbale=conninsidemacro

Macro Module : mid
Verification Model File : ./src/cell_in_macro/tb.sv(42)
Liberty Model File : src/cell_in_macro/analog.lib(54)

Power Cells : analog, File: ./src/cell_in_macro/tb.sv(53)
```

**Table 4-19. Power Cell Section Fields**

Field	Description
Macro Module	Name of the macro cell module
Verification Model File	HDL file where you define the macro cell module, preceded by its path, and the line number
Liberty Model File	Liberty file where you define the macro cell module, preceded by its path, and the line number
Power Cells	Name of the unconnected cell module
File	HDL file where you define the unconnected cell module, preceded by its path, and the line number

## Level Shifter Cell

This section lists the unconnected level shifter cells, which the simulator connects to the always-on power supply. This section is listed in the report only when your design has at least one unconnected level shifter cell, and you specify the following argument in vopt:

```
vopt -pa_enable=autoconnls
```

### Format

```
Level-Shifter Module : analog1
Verification Model File : ./src/ls_auto_conn/tb.sv(53)
Liberty Model File : src/ls_auto_conn/analog.lib(28)

Instances :
ab, File: ./src/ls_auto_conn/tb.sv(38)
```

**Table 4-20. Level Shifter Cell Section Fields**

Field	Description
Level-Shifter Module	Name of the level shifter module
Verification Model File	HDL file where you define the level shifter module, preceded by its path, and the line number

**Table 4-20. Level Shifter Cell Section Fields (cont.)**

Field	Description
Liberty Model File	Liberty file where you define the level shifter module, preceded by its path, and the line number
Instances	Instance of the level shifter module
File	HDL file where you specify the instance of the level shifter module, preceded by its path, and the line number

## Example of the Power Cell Report

```
-----  
-----QuestaSim Simulator connected Power Cells inside Macros and Level-  
Shifter  
Report File---  
-----  
Macro Module : mid  
Verification Model File : ./src/cell_in_macro/tb.sv(42)  
Liberty Model File : src/cell_in_macro/analog.lib(54)  
  
Power Cells :  
analog, File: ./src/cell_in_macro/tb.sv(53)  
  
Level-Shifter Module : analog1  
Verification Model File : ./src/ls_auto_conn/tb.sv(53)  
Liberty Model File : src/ls_auto_conn/analog.lib(28)  
  
Instances :  
ab, File: ./src/ls_auto_conn/tb.sv(38)  
abl, File: ./src/ls_auto_conn/tb.sv(47)  
...  
...
```

## Related Topics

- [Generating Reports for Power Aware](#)
- [Usage of vopt -pa\\_enable and -pa\\_disable](#)

## PST Analysis Report

Filename: *report.pst.txt*

vopt argument: -pa\_gen rpt

The PST analysis report contains information related to the PSTs in your design, including the composition details.

The PST analysis report contains the following sections, which are present as tags (strings within brackets) in the report. These tags also appear in the “[Static Check Report](#)”.

- [Power Domain Boundary Analysis \[BOUNDARY\\_ANALYSIS\]](#)

- Source Objects Referred in PSTs [OBJ]
- Power Domain/Supply Sets/Power State Group [APS]
- Power State Tables [PST]
- Composed PSTs [CPST]
- Duplicate PSTs [DUPLICATE\_PSTS]
- Undetermined States [UNDETERMINED\_STATES]
- Unreachable States [UNREACHABLE\_STATES]

For more information, refer to the “[Example of the PST Analysis Report](#)”.

## Power Domain Boundary Analysis [BOUNDARY\_ANALYSIS]

The [BOUNDARY\_ANALYSIS] section contains information about all power domain crossings that the simulator analyzes, and the results of the analysis. It is divided into numbered sections for each domain boundary.

### Format

```

1. Power Domain Boundary Analysis [BOUNDARY_ANALYSIS] :
  1. pd_C -> pd_B [PD1_to_PD2] :
    Isolation: Required
    Level shifter: Required
      Maximum voltage difference: 0.20 V
      Shift Direction: high_to_low
    Details:
      Analysis between supplies:
        Source Supply: VC, drives pd_C.primary.power
        Source Supply: G, drives pd_C.primary.ground
        Sink Supply: VB, drives pd_B.primary.power
        Sink Supply: G, drives pd_B.primary.ground
          Reason: Same source and sink GROUND supplies.
        PST used for analysis: Composed PST ID: [CP2]
+-----+-----+-----+
| State(s) | VC     | VB   |
|           | {Source} | {Sink} |
+-----+-----+-----+
| { [CP2_CS4] } | VC1   | VB1   | Ls (High to Low)
| { [CP2_CS2] } | VC1   | VB0   |
| { [CP2_CS3] } | VC0   | VB1   | Iso
| { [CP2_CS0] , [CP2_CS1] } | VC0   | VB0   |
+-----+-----+-----+

```

**Table 4-21. [BOUNDARY\_ANALYSIS] Section Fields**

Field	Description
<pd_C ->pd_B>	Source-sink power domain or supply set.
<PD1_to_PD2>	Power domain or supply set boundary tag, enclosed in brackets ([]).

**Table 4-21. [BOUNDARY\_ANALYSIS] Section Fields (cont.)**

Field	Description
Isolation	<p>One of the following values:</p> <ul style="list-style-type: none"> <li>• Required — Isolation is needed.</li> <li>• Not Required — Isolation is not needed.</li> <li>• Not Applicable (intermediate domains) — Isolation requirement is analyzed only for the actual source and sink domains. Any domain present in between the actual source and sink domain (intermediate domains) is not analyzed for isolation requirements. However, these intermediate domains are analyzed for level shifter requirements.</li> </ul>
Level shifter	<p>One of the following values:</p> <ul style="list-style-type: none"> <li>• Required — Level shifter is needed.</li> <li>• Not Required — Level shifter is not needed.</li> <li>• Not Analyzed — Level shifter requirement is not analyzed.</li> </ul> <p>The lines that follow the Level shifter field contain the details of the level shifter.</p>
Details	Details of the source and sink supplies.
PST used for analysis	Details of the PST used for analysis followed by the tabular representation of the port states of the supplies.
Ls	Identifier for the row that causes level shifting.
Iso	Identifier for the row that causes isolation.

### Source Objects Referred in PSTs [OBJ]

The [OBJ] section contains a list of driver supplies used in the PST analysis. The following are considered as driver supplies: an unconnected supply port, the output of any switch, an HDL supply port driving the supply net, or a resolved supply net.

#### Format

2. Source Objects Referred in PSTs [OBJ] :
  1. /tb/VDD\_08
  1. ON\_08 => VDD\_08 = {FULL\_ON, 0.80}

**Table 4-22. [OBJ] Section Fields**

Field	Description
</tb/VDD_08>	Name of the driver supply with its hierarchical path
<ON_08>	Power state of the driver supply
<VDD_08>	Name of the driver supply
<FULL_ON, 0.80>	Voltage information of the driver supply, enclosed in braces ({} )

## Power Domain/Supply Sets/Power State Group [APS]

The [APS] section contains information about the UPF objects that have power states specified by the add\_power\_state command in the UPF file. It also contains information about the converted PSTs and PST states. Refer to the section “[Power State Conversion](#)” for more information.

### Format

```

3. Power Domain/Supply Sets/Power State Group [APS] :
  1. SS:/testbench/DELIVER_PD.primary
    Converted PST: /testbench/DELIVER_PD.primary_pst
    Power States:
      1. ON_MODE, File: /src/test01_intf/test02.upf(125).
        Supply Expr:
          ((VDD_net == `FULL_ON, 0.80) && (GND_net == `FULL_ON, 0.00))
    Mapped PST states:
      1. ON_MODE
  
```

**Table 4-23. [APS] Section Fields**

Field	Description
SS or PD	One of the following UPF objects with its hierachal path: <ul style="list-style-type: none"> <li>• SS — Supply set</li> <li>• PD — Power domain</li> </ul>
Converted PST	Name of the converted PST
Power States	Details of the power states
<ON_MODE>	Power state of the supply set or power domain
File	UPF file where you define the add_power_state command, preceded by its path, and the line number
Supply Expr	Supply expression corresponding to the power state
Mapped PST states	PST states mapped to the power state

## Power State Tables [PST]

The [PST] section contains information about all unique PSTs specified in your UPF, where duplicate PSTs are not listed. If the PST has more than eight columns (default), no tables are displayed, instead a message and a list of columns of the PST is displayed.

Use the -pa\_pstcompflags=pstids argument with the vopt command line to add identifier tags to the PST table.

Warnings may be issued in the following instances:

- When a PST has more than one header referring to the same or equivalent supply (vopt-9865).

- When a PST has more than one header referring to the same or equivalent supply, and it has combinations referring to conflicting values (vopt-9866).

## Format

```

4. PSTs [PST] :
1. /tb/CPU_PST
+-----+
| CPU_PST | CPU_SW_VDD | VDD_08 | VDD_15 |
+-----+
| Full_on | ON_MODE   | ON_MODE | ON_MODE |
| PART_ON | ON_MODE   | ON_MODE | OFF_MODE |
+-----+

2. /tb/pd1.primary_pst
+-----+
| pd1.primary_pst | VSS[O1] | VDD[O5] | pd1.primary[O6] |
+-----+
| ON2[P12_S3]    | ON2[O1_V0] | ON[O5_V2] | ON[O6_V0] |
| OFF2[P12_S3]   | OFF2[O1_V0] | OFF[O5_V2] | OFF[O6_V0] |
| OFF2[P12_S3]   | OFF1[O1_V0] | OFF[O5_V2] | OFF[O6_V0] |
+-----+           | IGNORED      | IGNORED      | IGNORED

```

**Table 4-24. [PST] Section Fields**

Field	Description
</tb/CPU_PST>	PST that you define in the UPF file, followed by the tabular representation of the port states.
</tb/pd1.primary_pst>	PST converted from the add_power_state commands in your UPF file. The identifier for a converted PST is “_pst” suffix. Refer to the <a href="#">“Power Domain/Supply Sets/Power State Group [APS]”</a> section for details.
IGNORED	Identifier for a PST state that is undetermined or unreachable.

## Composed PSTs [CPST]

The [CPST] section contains information about all the composed PSTs created by the simulator for boundary analysis. If the composed PST has more than eight columns (default), no tables are shown, instead a message and a list of columns of the composed PST is shown.

Use the -pa\_maxpstcol <integer> argument with the vopt command to change the default number of columns.

## Format

```

5. Composed PSTs [CPST] :
1. Composed PST ID: [CP3]
Composed PST Table ( PSTs ) :
+-----+-----+-----+
| [CP3] | pd_ao.primary_pst[P0] | pd_vh.primary_pst[P1] |
+-----+-----+-----+
| [CP3_CS0] | <OFF_2[P0_S1]> | <OFF_2[P1_S1]> |
| [CP3_CS1] | SLE[P0_S2] | OFF_1[P1_S0] |
+-----+-----+-----+
Expanded Composed PST Table ( Supplies ) :
+-----+-----+-----+
| [CP4] | vdd_ao[O0] | vdd_v1[O2] | vss[O3] | ln3[O4] | pd_ao.primary[O5] |
+-----+-----+-----+
| [CP4_CS0] | <* [O0_V4]> | <* [O2_V4]> | <pOFF[O3_V2]> | {0}[O4_V1] | OFF[O5_V2]

```

**Table 4-25. [CPST] Section Fields**

Field	Description
Composed PST ID	Identifier for the composed PST, enclosed in brackets ([]).
Composed PST Table (PSTs)	Composed PST table in terms of individual PSTs used during composition. The columns show the names of individual PSTs, and the rows show the states on the PSTs that can coexist.
Expanded Composed PST Table (Supplies)	Expanded composed PST table in terms of individual supplies. The columns show the names of supplies used in each PST, and the rows show the port states added on those supplies.

## Duplicate PSTs [DUPLICATE\_PSTS]

The [DUPLICATE\_PSTS] section contains information about the PSTs that are identified as duplicates of other PSTs. A PST is considered a duplicate when it refers to the same or equivalent headers and exactly the same number of states, which results in a note (vopt-9880).

## Format

```

5. Duplicate PSTs [DUPLICATE_PSTS] :
1. PST: /tb/DUT/softIP_Vmem_pst
Duplicate PST(s) :
1. /tb/DUT/TOP1/eagle_mid/eagle_Vmem_pst
2. /tb/DUT/TOP1/king_mid/kf_Vmem_pst

```

**Table 4-26. [DUPLICATE\_PSTS] Section Fields**

Field	Description
PST	Name of the PST
Duplicate PST(s)	Name(s) of the duplicate PST(s)

## Undetermined States [UNDETERMINED\_STATES]

The [UNDETERMINED\_STATES] section contains information about the undetermined states identified during analysis. There are two types of undetermined states:

- **Undetermined port state** — Occurs when a state in a PST contains a reference to a port state that is not present in any other PST present in the same scope (horizontal composition). This results in a warning (vopt-9882).
- **Undetermined PST state** — Occurs when a higher-level PST contains reference to port states that are not present in lower-level PSTs (vertical composition). This results in a warning (vopt-9882).

### Format for Undetermined Port State

```
6. Undetermined States [UNDETERMINED_STATES]
  1. State: S3, PST: /tb/top_inst/PDTOP_pst
    Port state(Comb): { {FULL_ON,1.10,1.30}, {FULL_ON} }
    Incomplete PST(s): /tb/clus1/pd_pst[P4], /tb/clus1/pd_pst[P10]
```

### Format for Undetermined PST State

```
6. Undetermined States [UNDETERMINED_STATES]
  1. Power State: S3, Object: PD: /tb/top_inst/PDTOP
    Mapped PST states that are affected: All
    1. State: S3, PST: /tb/top_inst/PDTOP_pst
      Port state(Comb): { {FULL_ON,1.10,1.30}, {FULL_ON} }
      Incomplete PST(s): /tb/clus1/pd_pst[P4], /tb/clus1/pd_pst[P10]
```

**Table 4-27. [UNDETERMINED\_STATES] Section Fields**

Field	Description
Power State	Name of the power state
Object	Name of the UPF object
SS or PD	Type of the UPF object: <ul style="list-style-type: none"><li>• SS — Supply set</li><li>• PD — Power domain</li></ul>
Mapped PST states that are affected	PST states that are affected by the undetermined states: <ul style="list-style-type: none"><li>• All</li><li>• Partial</li></ul>
State	Name of the undetermined PST state
PST	Name of the PST
Port state (Comb)	Port states that caused the undetermined states in other PSTs
Incomplete PST(s)	PSTs that go into an undetermined state when a specified PST state occurs

## Unreachable States [UNREACHABLE\_STATES]

The [UNREACHABLE\_STATES] section contains information about the unreachable states identified during boundary analysis. There are two types of unreachable states:

- **Unreachable port state** — Occurs when states defined for a port are not referred to in any PSTs that you have defined. This results in a warning (vopt-9883).
- **Unreachable PST state** — Occurs when a lower-level PST state refers to port states that are not referred to in any top-level PST states. This results in a warning (vopt-9883).

### Format for Unreachable Port state

```
6. Unreachable States [UNREACHABLE_STATES] :
  1. Power State: V1_OFF, Supply: /tb/V1
```

### Format for Unreachable PST state

```
6. Unreachable States [UNREACHABLE_STATES] :
  1. Power State: OFF[A4_S0], Object: SS:/tb/pd_adv.default_retention[A4]
    Mapped PST states that are affected: All
    1. State: OFF_2[P8_S1], PST: /tb/pd_adv.default_retention_pst[P8]
      Port state(Comb): OFF[O14_V2], {OFF} [O1_V1]
      Incomplete PST(s): /tb/pd_adv_pst[P7], /tb/pd_cpu_pst[P4], /tb
      pd_12_pst[P10], /tb/pd_clus1top_pst[P2]
```

**Table 4-28. [UNREACHABLE\_STATES] Section Fields**

Field	Description
Power State	Name of the power state
Supply	Supply of the power state
Object	Name of the UPF object
SS or PD	Type of the UPF object: <ul style="list-style-type: none"> <li>• SS — Supply set</li> <li>• PD — Power domain</li> </ul>
Mapped PST states that are affected	PST states that are affected by the unreachable states: <ul style="list-style-type: none"> <li>• All</li> <li>• Partial</li> </ul>
State	Name of the unreachable PST state.
PST	Name of the PST
Port state (Comb)	Unreachable port state that caused the unreachable states in other PSTs
Incomplete PST(s)	PSTs that go into an unreachable state when a specified PST state occurs

## Example of the PST Analysis Report

```
-----  
----- QuestaSim Power Aware PST Analysis Report File -----  
-----  
... <Header information removed> ...  
-----  
  
1. Power Domain Boundary Analysis [BOUNDARY_ANALYSIS] :  
1. pd1 -> pd3 [PD2_to_PD1] :  
    Isolation: Not Required  
    Level shifter: Required for threshold: 0.20 V  
        Maximum voltage difference: 0.40 V  
        Shift Direction: low_to_high  
    Level shifter: Required  
        Maximum voltage difference: 0.40 V  
        Shift Direction: low_to_high  
    Details:  
        Analysis between supplies:  
            Source Supply: V1, drives pd1.primary.power  
            Sink Supply: V3, drives pd3.primary.power  
            PST used for analysis: Composed PST ID: [CP3]  
+-----+-----+  
| State(s)          | V1      | V3      |  
+-----+-----+  
| { [CP3_CS1] , [CP3_CS2] } | V1_ON  | V3_OFF |  
| { [CP3_CS0] }     | V1_ON  | V3_ON  |  
+-----+-----+  
...  
<additional [BOUNDARY_ANALYSIS] information>  
...  
-----  
2. Source Objects Referred in PSTs [OBJ] :  
1. /tb/top_inst/I2/vddA  
1. *  
2. { FULL_ON, 0.70, 0.90 } => vddA = { FULL_ON, 0.70, 0.90 }  
3. { FULL_ON } => vddA = { FULL_ON }  
4. { OFF } => vddA = { OFF }  
2. /tb/top_inst/I2/vssA  
1. *  
2. { FULL_ON } => vssA = { FULL_ON }  
3. { OFF } => vssA = { OFF }  
3. /tb/top_inst/PDTOP.SSH1  
1. ON => PDTOP.SSH1 = { ON }  
2. OFF => PDTOP.SSH1 = { OFF }  
...  
<additional [OBJ] information>  
...  
-----  
3. Power Domain/Supply Sets/Power State Group [APS] :  
1. PD:/tb/pd_ao[A3]  
    Converted PST: /tb/pd_ao_pst[DP0]  
    Power States:  
        1. RUN[A3_S1], File: /src/logicExpr2/test.upf(28).  
           Logic Expr:  
               (pd_ao.primary == ON)  
    Mapped PST states:  
        1. RUN[P0_S1]
```

```

2. SHD[A3_S0], File: /src/logicExpr2/test.upf(28).
    Logic Expr:
        (pd_ao.primary == OFF)
    Mapped PST states:
        1. SHD[P0_S0]
2. SS:/tb/pd_ao.primary[A1]
    Converted PST: /tb/pd_ao.primary_pst[P0]
    Power States:
        1. ON[A1_S1], File: /src/logicExpr2/test.upf(20).
            Logic Expr:
                (ln3 == 1)
            Mapped PST states:
                1. ON[P0_S1]
        2. OFF[A1_S0], File: /src/logicExpr2/test.upf(20).
            Logic Expr:
                (ln3 == 0)
            Mapped PST states:
                1. OFF[P0_S0]
...
<additional [APS] information>
...
-----
4. PSTs [PST]:
1. /tb/top_inst/PDTop_pst
    Unable to print table. Number of columns exceed the limit(8).
    List of column headers:
    1. vddA
    2. vssA
    3. PDTop.SSH1
    4. vddB
    5. vssB
    6. PDTop.SSH2
    7. PDTop.primary
    8. PD1.primary
    9. PD.SSAH
    10. PD.SSBH
    11. vddB_int
    12. PD.SSBH_SW
    13. PDTop.primary.power
    14. PDTop.primary.ground
    15. PD1.primary.power
    16. PD1.primary.ground

2. /tb/top_inst/PD1.primary_pst
+-----+-----+-----+
| PD1.primary_pst | PD1.primary | PD1.primary.power | PD1.primary.ground |
+-----+-----+-----+
| ON           | ON         | {FULL_ON}       | {FULL_ON}       |
+-----+-----+-----+
...
<additional [PST] information>
...
-----
5. Composed PSTs [CPST]:
1. Composed PST ID: [CP3]
    Composed PST Table ( PSTs ):
+-----+-----+
| [CP3]   | P12  | P23  |

```

# Power Aware Reports

## PST Analysis Report

```
+-----+-----+
| [CP3_CS0] | S1   | S2   |
| [CP3_CS1] | S1   | S1   |
| [CP3_CS2] | S0   | S0   |
+-----+-----+
Expanded Composed PST Table ( Supplies ) :
+-----+-----+
| [CP3]      | V1     | V2     | V3     |
+-----+-----+
| [CP3_CS0]  | V1_ON  | V2_ON  | V3_ON  |
| [CP3_CS1]  | V1_ON  | V2_ON  | V3_OFF |
| [CP3_CS2]  | V1_ON  | V2_OFF | V3_OFF |
+-----+-----+
...
<additional [CPST] information>
...
-----
6. Duplicate PSTs [DUPLICATE_PSTS] :
  1. PST: /tb/pd_ao.primary_pst[P0]
  Duplicate PST(s):
    1. /tb/pd_ao_pst[DP0]
  2. PST: /tb/pd_vl.primary_pst[P1]
  Duplicate PST(s):
    1. /tb/pd_vl_pst[DP1]
...
<additional [DUPLICATE_PSTS] information>
...
-----
7. Undetermined States [UNDETERMINED_STATES] :
  1. Power State: S3, Object: PD:/tb/top_inst/PDTOP
    Mapped PST states that are affected: All
    1. State: S3, PST: /tb/top_inst/PDTOP_pst
      Port state(Comb): {{FULL_ON,1.10,1.30}, {FULL_ON}},
      {{FULL_ON,0.70,0.90}, {FULL_ON}}
      Incomplete PST(s): /tb/top_inst/PDTOP.SSH2_pst, /tb/top_inst/PDTOP.SSH1_pst
  2. Power State: S7, Object: PD:/tb/top_inst/PDTOP
    Mapped PST states that are affected: All
    1. State: S7_2, PST: /tb/top_inst/PDTOP_pst
      Port state(Comb): {{FULL_ON,0.70,0.90}, {FULL_ON}}
      Incomplete PST(s): /tb/top_inst/PDTOP.SSH1_pst
    2. State: S7_1, PST: /tb/top_inst/PDTOP_pst
      Port state(Comb): {{FULL_ON,0.70,0.90}, {FULL_ON}}
      Incomplete PST(s): /tb/top_inst/PDTOP.SSH1_pst
...
<additional [UNDETERMINED_STATES] information>
...
-----
8. Unreachable States [UNREACHABLE_STATES] :
  1. Power State: S2, Object: PD:/tb/top_inst/I2/PD
    Mapped PST states that are affected: All
    1. State: S2, PST: /tb/top_inst/I2/PD_pst
      Port state(Comb): {{FULL_ON,1.10,1.30}, {FULL_ON}},
      {{FULL_ON,0.70,0.90}, {FULL_ON}}
      Incomplete PST(s): /tb/top_inst/PDTOP.SSH2_pst, /tb/top_inst/PDTOP.SSH1_pst
  2. Power State: S3, Object: PD:/tb/top_inst/I2/PD
    Mapped PST states that are affected: All
```

```
1. State: S3_2, PST: /tb/top_inst/I2/PD_pst
   Port state(Comb): { {FULL_ON, 0.70, 0.90}, {FULL_ON} }
   Incomplete PST(s): /tb/top_inst/PDTOP.SSH1_pst
2. State: S3_1, PST: /tb/top_inst/I2/PD_pst
   Port state(Comb): { {FULL_ON, 0.70, 0.90}, {FULL_ON} }
   Incomplete PST(s): /tb/top_inst/PDTOP.SSH1_pst
...
<additional [UNREACHABLE_STATES] information>
...
```

---

## Related Topics

[Generating Reports for Power Aware](#)

[Power State Tables](#)

## Source Sink Path Report

Filename: *report.srcsink.txt*

vopt argument: -pa\_genrpt=srcsink

The source sink path report contains information related to the source sink-paths that are analyzed for static checks.

## Format

---

```
|| SOURCE PD : pd_aon - SINK PD : pd ||

[PATH_15]
----- [Src Supply ( /tb/TOP/pd_aon )]
[DRIVER Buffer : /tb/TOP/clk_window]
[ Supply ( Supplies(power : /tb/TOP/VDD,ground : /tb/TOP/GND) ), ]
[ Attribute : report_srcsink/src/test.upf(26), ]
[RS Buffer : /tb/TOP/sub_in2/out2 [0:31]]
[Rtl Cell : /tb/TOP/lisininst3_UPF_LS]
[ Cand Port ( /tb/TOP/mid2/in2_bot ), ]
[ Strtgy ( ls_PD_mid2 ), Mask ( 1 ), ]
[ Input Port ( data ), ]
[ Output Port ( out ), ]
[ Src ( /tb/TOP/PD_tb ) ]
[ Sink ( -- ) ]
[ Supply ( Supplies(power : /tb/VDD_1_net,ground : /tb/GND_1_net) ), ]
[Rtl Wrapper Cell]
[ Cand Port ( /tb/TOP/inst2/in1 ), ]
[ Strtgy ( ISO2 ), ( LWR ), Mask ( 1111 ), ]
[ Rtl Inst ( /tb/TOP/G_1[1]/iso_bit_0 ), ]
[ Rtl Inst ( /tb/TOP/G_1[1]/iso_bit_1 ), ]
[ Rtl Inst ( /tb/TOP/G_1[1]/iso_bit_2 ), ]
[ Rtl Inst ( /tb/TOP/G_1[1]/iso_bit_3 ), ]
[PD Buffer : /tb/TOP/inst_ff/CP [0]]
[ Supply ( Supplies(power : /tb/QPA_NET,ground : /tb/QPA_NET) ), ]
[ Attribute : ./src/test02/ret_cell.lib(20), ]
[Port : /tb/TOP/itf_inst1[1]/c]
[ ( Input ) ]
[UFP Cell : /tb/TOP/itf_inst1[1]/c]
[ Cand Port ( /tb/TOP/sub_out1/in4 ), ]
[ level shifter ( LS1 ), (LWR) ]
[ Src ( Supplies(power : /tb/VDD_1_net,ground : /tb/GND_1_net) ) ]
[ Sink ( Supplies(power : /tb/VDD_2_net,ground : /tb/GND_2_net) ) ]
[RECEIVER Buffer : /tb/TOP/clk]
[ Supply ( Supplies(power : /tb/TOP/VDD,ground : /tb/TOP/GND) ), ]
[ Attribute : report_srcsink/src/test.upf(27), ]
[Sink Supply ( /tb/TOP/pd )]
```

**Table 4-29. Source Sink Report Fields**

Field	Description
SOURCE or SINK PD	Name of the source or sink power domain.
PATH_<integer>	Unique path-id tag that specifies a source-sink path. The <a href="#">Static Check Report</a> contains the path-id tag for each static check.
Src Supply	Name of the source power domain or supply set.
DRIVER Buffer	Buffer cell corresponding to the driver_supply specified in the set_port_attributes command.

**Table 4-29. Source Sink Report Fields (cont.)**

Field	Description
RS Buffer	Buffer cell corresponding to the set_related_supply_net command, or the repeater_supply specified in the set_repeater command.
Rtl Cell	Power Aware cell present in the design.
Cand Port	Candidate port of UPF strategy.
Rtl Wrapper Cell	Collection of ISO, LS, and ELS cells found in the source-sink path.
PD Buffer	Buffer cell corresponding to the power-down function (a liberty attribute) or related_power/ground_pin.
Port	HDL port lying on the power domain boundary.
UPF Cell	Simulator inserted Power Aware cell due to the UPF strategy.
LWR	Lower extent.
RECEIVER Buffer	Buffer cell corresponding to the receiver_supply specified in the set_port_attributes command.
Sink Supply	Name of the sink power domain or supply set.

## Static Check Report

Filename: *report.static.txt* and *report.static.tdb*

vopt argument: -pa\_checks=s

The static check report contains information related to the static checks performed on your design.

A database file (*report.static.tdb*) is also generated, which enables you to use the Results Analysis window to view the static check results.

### Restriction

 The database file (*report.static.tdb*) is not available for ModelSim SE.

The static check report contains the following sections:

- [Static Checks Summary Report](#)
- [Domain Wise Static Checks Detailed Report](#)

For more information, refer to the “[Example of the Static Check Report](#)”.

### Static Checks Summary Report

The Static Checks Summary Report section contains the summary of the static checks performed on your design.

#### Format

Check-ID	Count	Severity	Description
ISO_NOT_ANALYZED	17	Warning	Not analyzed isolation cells
LS_MISSING	3	Warning	Missing level shifters

**Table 4-30. Static Checks Summary Report Section Fields**

Field	Description
Check-ID	Mnemonic of the static check
Count	Count of the cells corresponding to the static check
Severity	Severity of the message that is displayed when the static check is performed
Description	Description of the static check

#### Domain Wise Static Checks Detailed Report

The Domain Wise Static Checks Detailed Report section contains the details of the static checks performed on your design, which is structured according to the source-sink power domains.

#### Format

```

1. Source power domain: /tb/PD_1 to Sink power domain: /tb/PD_1.
Total 1 Missing level shifters [Total Crossings:1, Shared Crossings:0]
Total 2 Not required isolation cells [Total Crossings:1, Shared
Crossings:0]
1.1. Source port: /tb/M1/y [LowConn] to Sink port: /tb/M2/a [LowConn],
width:1
Total 1 Missing level shifters [Total Crossings:1, Shared Crossings:0]
Total 1 Not required isolation cells [Total Crossings:1, Shared
Crossings:0]
1.1.1. Inferred type: LS_MISSING, count: 1
    Driver: /tb/M1/y [ LowConn ] [ Supplies(VDD_2,GND_2) ] ->
    Receiver: /tb/M2/a [ LowConn ]
    Possible reason: 'Level shifter is required from
    (Supplies(VDD_2,GND_2)) => (/tb/PD_1) and neither level shifter
    strategy nor level shifter cell is present in design'
    Analysis link: [SS4_to_PD2]
    Path-ID link: [PATH_15]

```

**Table 4-31. Domain Wise Static Checks Detailed Report Section Fields**

Field	Description
Source-Sink power domain	Source-sink power domain of the static check. The lines that follow the Source-Sink power domain field contain the summary of the static checks performed on the source-sink power domain.

**Table 4-31. Domain Wise Static Checks Detailed Report Section Fields (cont.)**

Field	Description
Source-Sink port	Source-sink port on which the static checks are performed. The lines that follow the Source-Sink port field contain the summary of the static checks performed on the source-sink port.
Inferred type	Mnemonic of the static check performed on the source-sink port. The lines that follow the Inferred type field contain the detail description of the static checks. The Path-ID link field contains a unique path-id tag, whose details are present in the <a href="#">Source Sink Path Report</a> .

## Example of the Static Check Report

```
-----  
----- QuestaSim Power Aware Static Checks Report File -----  
      <... Header information removed ...>  
-----  
+=====+  
||          Static Checks Summary report ||  
+=====+  
|Check-ID      Count  Severity   Description|  
|-----|  
|ISO_NOT_ANALYZED    17     Warning  Not analyzed isolation cells|  
|-----|  
|LS_MISSING        10     Warning  Missing level shifters|  
|-----|  
|LS_VALID          17     Note    Valid level shifters|  
|-----|  
|ISO_MISSING        14     Warning  Missing isolation cells|  
|-----|  
|ISO_VALID          17     Note    Valid isolation cells|  
|-----|  
+=====+  
  
+=====+  
||          Domain wise static Checks Detailed report ||  
+=====+  
  
-----  
1. Source power domain: /tb/TOP/PD_tb to Sink power domain: /tb/TOP/  
PD_mid1.  
Total 2 Missing level shifters [Total Crossings: 2, Shared Crossings: 0]  
Total 17 Valid isolation cells [Total Crossings: 17, Shared Crossings: 0]  
  
    1.1. Source port: /tb/TOP/isoinst1_0/data [LowConn] to Sink port: /tb/  
TOP/mid1/in1_bot[0] [LowConn], width:1  
        Total 1 Missing level shifters [Total Crossings: 1, Shared  
Crossings: 0]  
        Total 1 Valid isolation cells [Total Crossings: 1, Shared Crossings:  
0]  
            1.1.1. Inferred type: LS_MISSING, count: 1  
                Possible reason: 'Level shifter is required from (/tb/TOP/PD_tb)  
=> (/tb/TOP/PD_mid1) and neither level shifter strategy nor level shifter  
cell is present in design'  
                Analysis link: [PD2_to_PD4]  
            1.1.2. Inferred type: ISO_VALID, count: 1  
                Candidate Port: /tb/TOP/mid1/in1_bot[0], Isolation Cell: /tb/  
TOP/isoinst1_0(ISO_AND), Strategy: iso_PD_mid1, Domain: /tb/TOP/PD_mid1  
                Possible reason: 'Isolation is required and is present from (/  
tb/TOP/PD_tb) => (/tb/TOP/PD_mid1)'  
                Analysis link: [PD2_to_PD4]  
        1.2. Source port: /tb/TOP/isoinst1_1/data [LowConn] to Sink port: /tb/  
TOP/mid1/in1_bot[1] [LowConn], width:1  
        Total 1 Missing level shifters [Total Crossings: 1, Shared  
Crossings: 0]  
        Total 1 Valid isolation cells [Total Crossings: 1, Shared Crossings:  
0]  
            1.2.1. Inferred type: LS_MISSING, count: 1  
                Possible reason: 'Level shifter is required from (/tb/TOP/PD_tb)  
=> (/tb/TOP/PD_mid1) and neither level shifter strategy nor level shifter
```

```

cell is present in design'
    Analysis link: [PD2_to_PD4]

    1.2.2. Inferred type: ISO_VALID, count: 1
        Candidate Port: /tb/TOP/mid1/in1_bot[1], Isolation Cell: /tb/
TOP/isoinst1_1(ISO_AND), Strategy: iso_PD_mid1, Domain: /tb/TOP/PD_mid1
        Possible reason: 'Isolation is required and is present from (/
tb/TOP/PD_tb) => (/tb/TOP/PD_mid1)'
        Analysis link: [PD2_to_PD4]

    1.3. Source port: /tb/TOP/isoinst1_2/data [LowConn] to Sink port: /tb/
TOP/mid1/in1_bot[2] [LowConn], width:1
        Total 1 Valid isolation cells [Total Crossings: 1, Shared Crossings:
0]
        1.3.1. Inferred type: ISO_VALID, Strategy status: Missing, count: 1
            Isolation Cell: /tb/TOP/isoinst1_2(ISO_AND),
            Possible reason: 'Isolation is required and is present from (/
tb/TOP/PD_tb) => (/tb/TOP/PD_mid1)'
            Analysis link: [PD2_to_PD4]

    1.4. Source port: /tb/TOP/isoinst1_3/data [LowConn] to Sink port: /tb/
TOP/mid1/in1_bot[3] [LowConn], width:1Total 1 Valid isolation cells [Total
Crossings: 1, Shared Crossings: 0]

        1.4.1. Inferred type: ISO_VALID, Strategy status: Missing, count: 1
            Isolation Cell: /tb/TOP/isoinst1_3(ISO_AND),
            Possible reason: 'Isolation is required and is present from (/
tb/TOP/PD_tb) => (/tb/TOP/PD_mid1)'
            Analysis link: [PD2_to_PD4]

    1.5. Source port: /tb/TOP/isoinst1_4/data [LowConn] to Sink port: /tb/
TOP/mid1/in1_bot[4] [LowConn], width:1
        Total 1 Valid isolation cells [Total Crossings: 1, Shared Crossings:
0]
        1.5.1. Inferred type: ISO_VALID, Strategy status: Missing, count: 1
            Isolation Cell: /tb/TOP/isoinst1_4(ISO_AND),
            Possible reason: 'Isolation is required and is present from (/
tb/TOP/PD_tb) => (/tb/TOP/PD_mid1)'
            Analysis link: [PD2_to_PD4]
        ...

```

## Related Topics

- [Generating Reports for Power Aware](#)
- [Power Aware Static Check Display](#)

# Supply Network Initialization Report

Filename: *report.supplynetworkinit.txt*

vopt argument: -pa\_gen rpt

The supply network initialization report contains initial value of the UPF input supply port, output supply port (if the port has no driver), default isolation and retention supply set of a

power domain, and switch supply set of a power domain. These initial values are set by the simulator.

## Example of the Supply Network Initialization Report

```
-----  
----- ModelSim Power Aware Supply Network Initialization Report File -----  
-----  
/top_v1/QPA_AON_DUMMY_NET = = '{FULL_ON,0}  
/top_v1/VDD_comb = = '{OFF,0}  
/top_v1/PD_1.default_isolation.power = = '{FULL_ON,0}  
/top_v1/PD_1.default_isolation.ground = = '{FULL_ON,0}  
/top_v1/PD_1.default_retention.power = = '{FULL_ON,0}  
/top_v1/PD_1.default_retention.ground = = '{FULL_ON,0}  
/top_v1/PD_1_SW.supply.power = = '{FULL_ON,0}  
/top_v1/PD_1_SW.supply.ground = = '{FULL_ON,0}  
/top_v1/PD_0.default_isolation.power = = '{FULL_ON,0}  
/top_v1/PD_0.default_isolation.ground = = '{FULL_ON,0}  
/top_v1/PD_0.default_retention.power = = '{FULL_ON,0}  
/top_v1/PD_0.default_retention.ground = = '{FULL_ON,0}  
/top_v1/PD_0_SW.supply.power = = '{FULL_ON,0}  
/top_v1/PD_0_SW.supply.ground = = '{FULL_ON,0}  
/top_v1/PD_1_GNET = = '{FULL_ON,0}  
/top_v1/PD_1_VNET = = '{FULL_ON,0}  
/top_v1/PD_0_GNET = = '{FULL_ON,0}  
/top_v1/PD_0_VNET = = '{FULL_ON,0}
```

## Related Topics

[Generating Reports for Power Aware](#)

## Unassociated Cell Report

Filename: *report.unassociated.cells.txt*

vopt argument: None

The unassociated cell report contains a list of Power Aware cells that are not associated with any UPF strategy.

---

### Note

 The unassociated cell report is generated only when there is at least one Power Aware cell that is not associated with any UPF strategy.

---

### Format

```
PA switch cell Module : Switch2
Verification Model File : ./src/chip.v(64)
Liberty Model File : src/switch.lib(2)

Power Cells :
sw2_1, File: ./src/chip.v(41)
```

**Table 4-32. Unassociated Cell Report Fields**

Field	Description
PA switch cell Module	Name of the unassociated cell module
Verification Model File	HDL file where you define the unassociated cell module, preceded by its path, and the line number
Liberty Model File	Liberty file where you define the unassociated cell module, preceded by its path, and the line number
Power Cells	Instance of the unassociated cell module
File	HDL file where you specify the module instance, preceded by its path, and the line number

## Example of the Unassociated Cell Report

```
-----
-----QuestaSim Report File for PA cells which are unassociated with any
UPF strategy-----
-----
PA switch cell Module : Switch1
Verification Model File : ./src/chip.v(37)

Power Cells :
    sw1_2, File: ./src/chip.v(31)

PA switch cell Module : Switch2
Verification Model File : ./src/chip.v(64)
Liberty Model File : src/switch.lib(2)

Power Cells :
    sw2_1, File: ./src/chip.v(41)
    sw2_5, File: ./src/chip.v(43)
...
```

## Related Topics

[Generating Reports for Power Aware](#)



# Chapter 5

## Power Aware Checks

---

Power Aware simulation performs a variety of checks on your design that validate the power intent defined in the UPF file. These checks ensure that the power intent protocol violations do not occur because of a power event sequence or the power architecture.

<b>Power Aware Checks Overview .....</b>	<b>144</b>
Isolation Checks .....	146
Level Shifter Checks.....	146
Path Analysis Checks .....	147
Retention Checks .....	147
Back-to-Back Checks .....	147
Power State Composition .....	148
<b>Power Aware Checks Command Reference .....</b>	<b>154</b>
Static RTL Checks .....	155
Static GLS Checks .....	163
Dynamic Checks.....	179
Quick Reference of Static RTL and Dynamic Checks .....	193
<b>Power Aware Checks Control .....</b>	<b>196</b>
Controlling Checks With the pa_checks Command.....	197
Controlling Checks With the pa msg Command .....	199
Precedence Order of Arguments.....	201
Pathname Convention in Arguments .....	202

## Power Aware Checks Overview

---

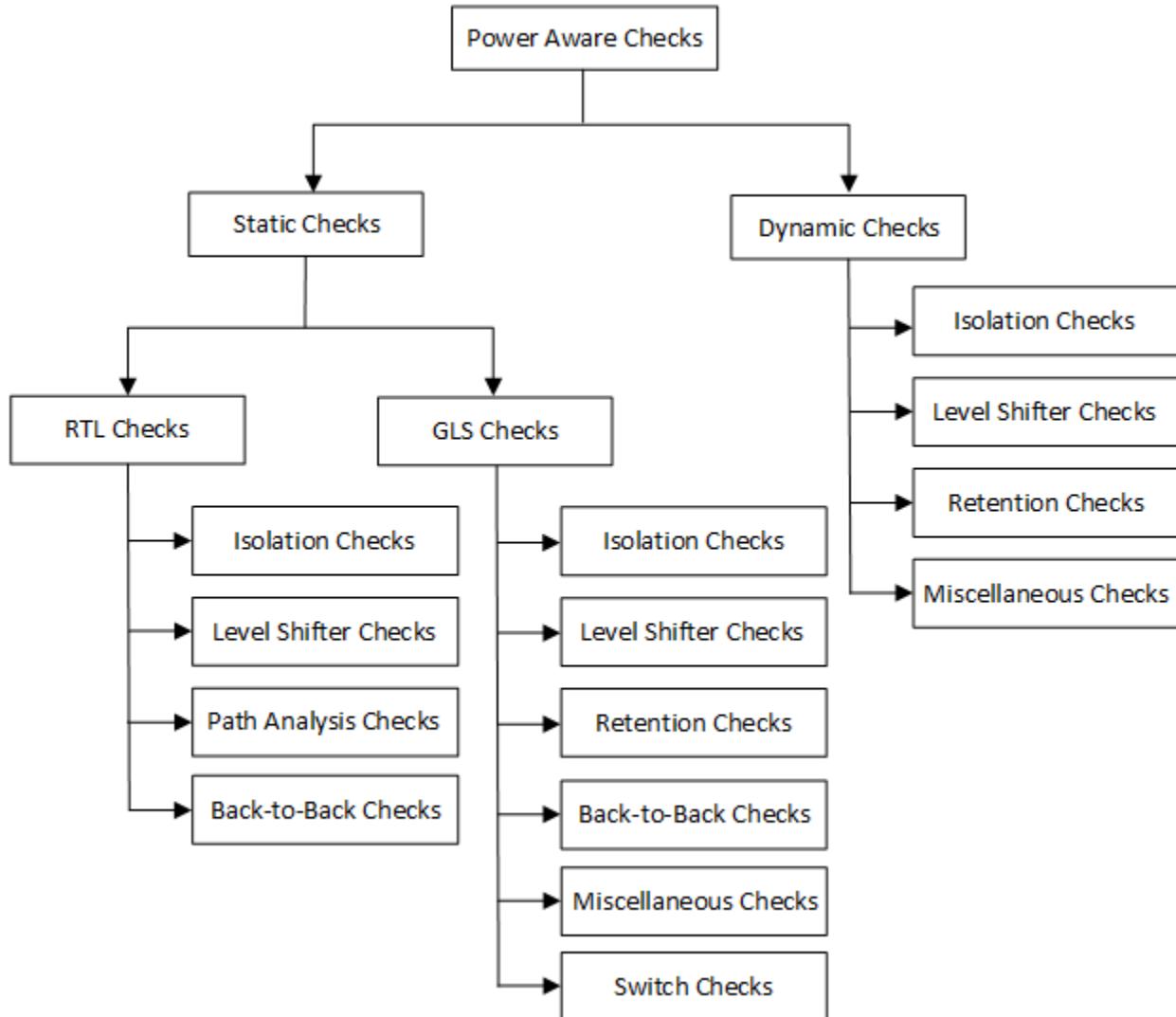
Use various Power Aware checks on your design to ensure that it is compliant to the power intent defined in the UPF file. Power Aware checks use the power intent definition to report any missing or inconsistent power-related information, saving your time in simulation and synthesis debugging.

There are two categories of Power Aware checks:

- **Static Check** — Static checks identify architectural issues in the design, such as missing isolation or level shifter cells. A static check verifies the design for the defined power intent without running the actual simulation. Because a simulation run is not required, you only need to run vopt to perform static checks.  
Static checks are categorized into static RTL and static GLS checks. Use static RTL checks on the RTL designs and static GLS checks on the gate-level designs.
- **Dynamic Check** — Dynamic checks identify behavioral issues in the design, such as incorrect control sequencing for powering up or powering down portions of the design. A dynamic check verifies the design for the defined power intent during the actual simulation. You need to run Power Aware simulation to perform dynamic checks.

Figure 5-1 shows the categories of Power Aware checks.

**Figure 5-1. Power Aware Checks**



Use the `pa_checks` and `pa msg` commands for granular control of Power Aware checks. These commands selectively enable or disable specific Power Aware checks for a specific design or UPF object. For more information, refer to “[Power Aware Checks Control](#)”.

<b>Isolation Checks</b> .....	<a href="#">146</a>
<b>Level Shifter Checks</b> .....	<a href="#">146</a>
<b>Path Analysis Checks</b> .....	<a href="#">147</a>
<b>Retention Checks</b> .....	<a href="#">147</a>
<b>Back-to-Back Checks</b> .....	<a href="#">147</a>
<b>Power State Composition</b> .....	<a href="#">148</a>

## Isolation Checks

Use isolation checks to report any missing or inconsistent isolation cells in your design. An isolation cell is present in the design if your UPF file contains a `set_isolation`-instance command, which defines that cell in the design as an isolation cell.

If you have a power domain crossing, isolation cells have the following requirements:

- An isolation cell is *statically* required if the source domain can be off when the sink domain is on.
- An isolation cell is *dynamically* required if the source domain is off when the sink domain is on.

### Related Topics

[Static RTL Isolation Checks](#)

[Static GLS Isolation Checks](#)

[Dynamic Isolation Checks](#)

## Level Shifter Checks

Use level shifter checks to report any missing or inconsistent level shifter cells in your design. A level shifter cell is present in the design if your UPF file contains a `set_level_shifter`-instance command, which defines that cell in the design as a level shifter cell.

If you have a power domain crossing, level shifter cells have the following requirements:

- A level shifter cell is *statically* required if the source and sink domains can be both powered on at the same time, and the difference between the maximum voltages exceeds a certain threshold voltage.
- A level shifter cell is *dynamically* required if the source and sink domains are both powered on at the same time, and the difference between the maximum voltages exceeds a certain threshold voltage.

A level shifter cell has either of the following direction:

- **high\_to\_low** — If the maximum voltage powering the source domain is *higher* than the maximum voltage powering the sink domain.
- **low\_to\_high** — If the maximum voltage powering the source domain is *higher* than the maximum voltage powering the sink domain.

### Related Topics

[Static RTL Level Shifter Checks](#)

- [Static GLS Level Shifter Checks](#)
- [Dynamic Level Shifter Checks](#)

## Path Analysis Checks

Use path analysis checks to report not analyzed/unanalyzed and good paths for isolation and level shifter requirements. Path analysis checks are not reported by default, because there are many not analyzed/unanalyzed and good paths for a power domain crossing.

During path analysis checks, the Questa SIM simulator reports the following paths:

- **Not analyzed/Unanalyzed path** — The power domain crossing is not analyzed for the isolation and level shifter requirements because of insufficient power state table (PST) information on either the source or sink supply.
- **Good path** — The power domain crossing is analyzed for the isolation and level shifter requirements, and isolation and level shifter cells are not required for the crossing.

### Related Topics

- [Static RTL Path Analysis Checks](#)

## Retention Checks

Use retention checks to report various inconsistent retention conditions in your design. A retention cell is present in the design if your UPF file contains a set\_retention -instance command, which defines that cell in the design as a retention cell.

If you have to retain a state element in a power domain, these are the requirements:

- A retention cell is *statically* required if the power domain can toggle between off and on states.
- A retention cell is *dynamically* required if the power domain toggles between off and on states.

### Related Topics

- [Static GLS Retention Checks](#)

## Back-to-Back Checks

Use back-to-back checks to report back-to-back cells in your design. Replace the back-to-back cells with a single cell.

During back-to-back checks, the Questa SIM simulator reports the following back-to-back cells:

- Isolation – Level shifter
- Level shifter – Isolation
- Isolation – Isolation
- Level shifter – Level shifter

## Related Topics

[Static RTL Back-to-Back Checks](#)

[Static GLS Back-to-Back Checks](#)

# Power State Composition

States of ports and nets carry voltage and on/off information, which means combinations of power states of supply ports and/or nets are important in identifying the requirement of isolation cells and level shifters on boundaries between two power domains. Because a power state added on one domain/supply set can reference other power states added on another domain/supply set, Power Aware simulation can statically infer such dependencies. These dependencies help in determining the combinations of power states of supply nets and/or ports.

You can add composite power states to your design as follows:

- Supply net — power state information (state and voltage level) is defined by the `supply_net_type` declaration of the HDL cell (see “[Explicit Connections to HDL Ports](#)”).
- Supply port — power state information (state and voltage level) is defined by the `supply_net_type` declaration of the HDL cell (see “[Explicit Connections to HDL Ports](#)”).
- Supply set — composed of two or more supply nets, so its power state is specified in terms of those supply nets.
- Power domain — power state is determined by the state of supply sets associated with the domain.

Power Aware simulation performs level shifter and isolation cell analysis/checks from information provided with `add_power_state` UPF commands and also reports the power domain state dependencies in report files.

To know whether an isolation or level-shifting is required on a boundary between power domains, a static analysis is required to determine state dependencies between these power domains (or supply sets associated with these domains). The dependency information is extracted from supply/logic expression (mentioned in `add_power_state` command) of various power states from power domains or supply sets.

A valid combination of two power states of domains/supply sets is equivalent to a valid combination of power states of certain supply ports/supply nets (state and voltages). Power Aware simulation uses this net state and voltage information to determine whether isolation or level-shifting is required or not.

---

**Note**

 For the cases where Power Aware simulation is statically not able to determine whether the state combination of two power domain/supply sets is invalid (or these power states cannot co-exist at any point of time), Power Aware simulation assumes it to be a valid state combination.

---

In the following example for static analysis, Power Aware simulation assumes that PD1\_ON and PD2\_ON may co-exist at some point of time.

```
add_power_state PD1 -state PD1_ON
{-supply_expr {VDD1 == FULL_ON && GND1 == FULL_ON} }

add_power_state PD2 -state PD2_ON
{-supply_expr {VDD2 == FULL_ON && GND2 == FULL_ON} }
```

This information corresponds that (VDD1 is FULL\_ON, GND1 is FULL\_ON, VDD2 is FULL\_ON, GND2 is FULL\_ON) may exist at some point of time.

---

**Note**

 For better static analysis, it is recommended to include the state information of primary nets of power domain in supply expression of add\_power\_state for power domain (or its associated supply set).

Also, to determine the state relation between two power domains/supply sets, Power Aware simulation uses the information specified in power states of those two power domains/supply sets.

---

For example:

```
create_supply_net VDD ...
create_supply_net GND ...
set_domain_supply_net PD1 -primary_power_net VDD -primary_ground_net GND
add_power_state PD1 -state PD1_ON {-logic_expr {PD2 == PD2_OFF}
-supply_expr {VDD == FULL_ON && GND == FULL_ON} }
```

## State Dependency Determination with add\_power\_state Options

You can use the following options of the UPF add\_power\_state command to determine whether power states of different power domains can co-exist:

- -logic\_expr
- -supply\_expr

The following examples show various ways of using these options.

### Example 5-1. Dependency Specified with add\_power\_state -logic\_expr — Case 1

In this example, from logic\_expr of PD1\_S1 Power Aware simulation determines that state PD1\_S1 and PD2\_S2 cannot co-exist.

```
add_power_state PD1 -state PD1_S1 {-logic_expr {PD2 != PD2_S2}
    -supply_expr {VDD1 == FULL_ON && GND1 == FULL_ON} }

add_power_state PD2 -state PD2_S2 {-supply_expr {VDD1 == FULL_ON &&
    GND1 == FULL_ON}}
```

### Example 5-2. Dependency Specified with add\_power\_state -logic\_expr — Case 2

In this example, from logic\_expr of PD1\_S1 and PD2\_S2 Power Aware simulation determines that state PD1\_S1 and PD2\_S2 cannot co-exist, as one requires ctrl to be 1 and other requires it to be 0.

```
add_power_state PD1 -state PD1_S1 {-logic_expr {ctrl} -supply_expr
    {VDD1 == FULL_ON && GND1 == FULL_ON} }

add_power_state PD2 -state PD2_S2 {-logic_expr {!ctrl} -supply_expr
    {VDD1 == FULL_ON && GND1 == FULL_ON}}
```

### Example 5-3. Dependency Specified with add\_power\_state -logic\_expr — Case 3

In this example, from logic\_expr of PD1\_S1, PD3\_S3 Power Aware simulation determines that state PD1\_S1 and PD2\_S2 cannot co-exist.

```

add_power_state PD1 -state PD1_S1 {-logic_expr {PD3 == PD3_S3}
    -supply_expr {VDD1 == FULL_ON && GND1 == FULL_ON} }

add_power_state PD2 -state PD2_S2 {-logic_expr {!ctrl} -supply_expr
    {VDD1 == FULL_ON && GND1 == FULL_ON} }

add_power_state PD3 -state PD3_S3 {-logic_expr {PD2 != PD2_S2}
    -supply_expr {VDD3 == FULL_ON && GND3 == FULL_ON} }

```

#### **Example 5-4. Dependency Specified with add\_power\_state -logic\_expr — Case 4**

In this example, from logic\_expr of PD1\_S1 and PD2\_S2 Power Aware simulation determines that state PD1\_S1 and PD2\_S2 cannot co-exist, as one requires ctrl to be 1 and other requires it to be 0.

```

add_power_state PD1 -state PD1_S1 {-logic_expr {ctrl == 1} -supply_expr
    {VDD1 == FULL_ON && GND1 == FULL_ON} }

add_power_state PD2 -state PD2_S2 {-logic_expr {!ctrl} -supply_expr
    {VDD1 == FULL_ON && GND1 == FULL_ON} }

```

#### **Example 5-5. Dependency Specified with add\_power\_state -supply\_expr — Case 1**

In this example, from supply\_expr of PD1\_S1 and PD2\_S2 (VDD state) Power Aware simulation determines that state PD1\_S1 and PD2\_S2 cannot co-exist.

```

add_power_state PD1 -state PD1_S1 {-supply_expr {VDD1 == FULL_ON &&
    GND1 == FULL_ON && VDD == FULL_ON} }

add_power_state PD2 -state PD2_S2 {-supply_expr {VDD1 == FULL_ON &&
    GND1 == FULL_ON && VDD == OFF} }

```

#### **Example 5-6. Dependency Specified with add\_power\_state -supply\_expr — Case 2**

In this example, from supply\_expr of PD1\_S1 and PD2\_S2 (VDD state), Power Aware simulation determines that state PD1\_S1 and PD2\_S2 cannot co-exist.

```

add_power_state PD1 -state PD1_S1 {-supply_expr {VDD1 == FULL_ON &&
    GND1 == FULL_ON && VDD == FULL_ON} }

add_power_state PD2 -state PD2_S2 {-supply_expr {VDD1 == FULL_ON &&
    GND1 == FULL_ON && VDD != FULL_ON} }

```

**Example 5-7. Dependency Specified with add\_power\_state -supply\_expr — Case 3**

In this example, from supply\_expr of PD1\_S1 and PD2\_S2 (VDD voltage range), Power Aware simulation determines that state PD1\_S1 and PD2\_S2 cannot co-exist.

```
add_power_state PD1 -state PD1_S1 {-supply_expr {VDD1 == FULL_ON &&
    GND1 == FULL_ON && VDD == '{FULL_ON, 0.8, 1.4}'}}
add_power_state PD2 -state PD2_S2 {-supply_expr {VDD1 == FULL_ON &&
    GND1 == FULL_ON && VDD == '{FULL_ON, 1.6, 2.4}'}}
```

**Power State Reporting**

Power Aware simulation provides the following types of information in the report files:

- Information about the power state added on power domain/supply sets.
- Power State dependencies between connected power domains.

### Example Report (Excerpt)

```

Power state info of Power domain : 'PD_tb'.
1. Power state 'PD_tb_normal', File: src/testcase4/test.upf(15).
   Supply Expression : (((tb_pow)==('FULL_ON , 3.30
}))&&((tb_gnd)==('FULL_ON , 1.00 )))).
   Function States :
1. Ground(tb_gnd) == {FULL_ON,1.00}, Power(tb_pow) ==
{FULL_ON,3.30}.

Power state info of Power domain : 'PD_mid2'.
1. Power state 'PD_mid2_off', File: src/testcase4/test.upf(67).
   Supply Expression : (((mid2_MAIN_NET)==('OFF
}))&&((mid2_GND_NET)==('OFF )))).
   Function States :
1. Ground(mid2_GND_NET) == {OFF}, Power(mid2_MAIN_NET) ==
{OFF}.

2. Power state 'PD_mid2_normal', File: src/testcase4/test.upf(66).
   Supply Expression : (((mid2_MAIN_NET)==('FULL_ON , 4.20
)))&&((mid2_GND_NET)==('FULL_ON , 1.30 )))).
   Function States :
1. Ground(mid2_GND_NET) == {FULL_ON,1.30}, Power(mid2_MAIN_NET)
== {FULL_ON,4.20}.

```

Power State Combinations for connected domains :

Power Domain 'PD_mid1'	Power Domain 'PD_wrapper1'
PD_mid1_off	PD_wrapper1_normal
PD_mid1_normal	PD_wrapper1_normal
Power Domain 'PD_mid1'	Power Domain 'PD_wrapper2'
PD_mid1_off	PD_wrapper2_normal
PD_mid1_normal	PD_wrapper2_normal
Power Domain 'PD_mid1'	Power Domain 'PD_mid3'
PD_mid1_off	PD_mid3_off
PD_mid1_off	PD_mid3_normal
PD_mid1_normal	PD_mid3_off
PD_mid1_normal	PD_mid3_normal
Power Domain 'PD_wrapper1'	Power Domain 'PD_mid3'
PD_wrapper1_normal	PD_mid3_off
PD_wrapper1_normal	PD_mid3_normal

# Power Aware Checks Command Reference

---

This section summarizes the arguments to the `vopt` command that you use to enable static RTL, static GLS, and dynamic checks.

**Table 5-1. Power Aware Checks Command Reference**

Check	Usage Syntax	Description
Static RTL Checks	<code>vopt -pa_checks=s</code>	Enables all static RTL isolation and level shifter checks.
Static GLS Checks	<code>vopt -pa_glschecks=s</code>	Enables all static GLS checks, except static GLS back-to-back cell checks. You have to explicitly enable static GLS back-to-back cell checks.
Dynamic Checks	<code>vopt -pa_checks=d</code>	Enables all dynamic checks. Alternatively, specifying <code>-pa_checks</code> without any argument also enables all dynamic checks.

---

**Note**

 To specify more than one Power Aware check, use a plus sign (+) to separate multiple values:

---

```
vopt -pa_checks=s+d
```

---

Refer to the “[vopt](#)” section of the *Questa SIM Command Reference Manual* for details on the usage of the `vopt` command.

<b>Static RTL Checks</b> .....	<a href="#">155</a>
<b>Static GLS Checks</b> .....	<a href="#">163</a>
<b>Dynamic Checks</b> .....	<a href="#">179</a>
<b>Quick Reference of Static RTL and Dynamic Checks</b> .....	<a href="#">193</a>

## Static RTL Checks

Static RTL checks validate your RTL design with the power intent defined in the UPF file. For static RTL checks, the Questa SIM simulator analyzes any specified power state tables (PSTs), power states added on power domains (by an add\_power\_state command), and supply sets in the UPF file. The purpose of this analysis is to detect the power domain relative ON/OFF condition and the relative operating voltages.

Use various static RTL checks (such as isolation, level shifter, path analysis, and back-to-back) on your RTL design.

To enable all static RTL checks, specify the argument -pa\_checks=s to the vopt command. This enables all static RTL checks, except static RTL path analysis checks.

When you enable all static RTL checks, if state dependencies between two connected power domains are not present in the PST/add\_power\_state, then the Questa SIM simulator cannot determine power domain relative ON/OFF states statically. In this case, static RTL checks report strategies as *Not Analyzed*. If state dependencies between two connected power domains are present in the PST/add\_power\_state, then the Questa SIM simulator performs all static RTL checks.

Static RTL checks generate a vopt log file, and a detailed report file (*report.static.txt*):

- **vopt log file** — Summary of the static RTL check results.
- **Report file** — Detailed results of the static RTL check.

**Table 5-2. Static RTL Checks**

Check	Description
Static RTL Isolation Checks	Use static RTL isolation checks to report any missing or inconsistent isolation cells in your RTL design.
Static RTL Level Shifter Checks	Use static RTL level shifter checks to report any missing or inconsistent level shifter cells in your RTL design.
Static RTL Path Analysis Checks	Use static RTL path analysis checks to report any not analyzed path or good paths in your RTL design.
Static RTL Back-to-Back Checks	Use static RTL back-to-back checks to report any back-to-back cells in your RTL design.

## Static RTL Isolation Checks

Use static RTL isolation checks to report any missing or inconsistent isolation cells in your RTL design.

To enable static RTL isolation checks, specify a value to the vopt -pa\_checks command as shown in [Table 5-3](#). All static RTL isolation check results are written to the report file, *report.static.txt*.

**Table 5-3. Static RTL Isolation Checks**

Check	Usage Syntax	Description Mnemonic Example Message
Missing isolation cell	vopt -pa_checks=smi	Reports missing isolation cell when an isolation cell is required for the power domain crossing, but an isolation cell is not present in the design, and the tool did not infer one.  Mnemonic: ISO_MISSING  Warning: (vopt-9750) [UPF_ISO_STATIC_CHK] Found Total 1 Missing isolation cells.
Not required isolation cell	vopt -pa_checks=sri	Reports not required isolation cell when an isolation cell is not required for the power domain crossing, but either an isolation cell is present in the design or the tool infers one.  Mnemonic: ISO_NOT_REQUIRED  Warning: (vopt-9750) [UPF_ISO_STATIC_CHK] Found Total 3 Not required isolation cells.
Incorrect isolation cell	vopt -pa_checks=ssi	Reports incorrect isolation cell when an isolation cell is required for the power domain crossing, but you specify the <b>set_isolation -no_isolation</b> command in the UPF file.  Mnemonics: ISO_INCORRECT ISO_INCORRECT_SUPPLY  Warning: (vopt-9750) [UPF_ISO_STATIC_CHK] Found Total 3 Incorrect isolation cells.
Valid isolation cell	vopt -pa_checks=svi	Reports valid isolation cell when an isolation cell is required for the power domain crossing, and either an isolation cell is present in the design or the tool infers one.  Mnemonic: ISO_VALID  Note: (vopt-9750) [UPF_ISO_STATIC_CHK] Found Total 1 Valid isolation cells.

**Table 5-3. Static RTL Isolation Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
Not analyzed isolation cell	vopt -pa_checks=sni	Reports not analyzed isolation cell when the power state table (PST) information is not sufficient to analyze whether an isolation cell is required or not for the power domain crossing.  Mnemonic: ISO_NOT_ANALYZED  Warning: (vopt-9750) [UPF_ISO_STATIC_CHK] Found Total 1 Not analyzed isolation cells.
Not inserted isolation cell	vopt -pa_checks=sdi	Reports not inserted isolation cell when you specify the isolation strategy with the <b>set_isolation -no_isolation</b> command in the UPF file. However, an isolation cell may or may not be required for the power domain crossing.  Mnemonic: ISO_NOT_INSERTED  Warning: (vopt-9750) [UPF_ISO_STATIC_CHK] Found Total 1 Not inserted isolation cells.
All static RTL isolation check	vopt -pa_checks=si	Performs all static RTL isolation checks (smi, sri, sii, svi, sni, and sdi).

## Related Topics

[Isolation Checks](#)

[Static Check Report](#)

## Static RTL Level Shifter Checks

Use static RTL level shifter checks to report any missing or inconsistent level shifter cells in your RTL design.

To enable static RTL level shifter checks, specify a value to the vopt -pa\_checks command as shown in [Table 5-4](#). All static RTL isolation check results are written to the report file, *report.static.txt*.

**Table 5-4. Static RTL Level Shifter Checks**

Check	Usage Syntax	Description Mnemonic Example Message
Missing level shifter cell	vopt -pa_checks=sml	<p>Reports missing level shifter cell when a level shifter cell is required for the power domain crossing, but a level shifter cell is not present in the design, and the simulator did not infer one.</p> <p>Mnemonic: LS_MISSING</p> <p>Warning: (vopt-9693)  [UPF_LS_STATIC_CHK] Found Total 1 Missing level shifters.</p>
Not required level shifter cell	vopt -pa_checks=srl	<p>Reports not required level shifter cell when a level shifter cell is not required for the power domain crossing, but either a level shifter cell is present in the design or the simulator infers one.</p> <p>Mnemonic: LS_REDUNDANT</p> <p>Warning: (vopt-9693)  [UPF_LS_STATIC_CHK] Found Total 1 Not Required level shifters.</p>
Incorrect level shifter cell	vopt -pa_checks=sil	<p>Reports incorrect level shifter cell when the direction of the level shifter cell specified in the UPF file does not match with the direction as per the voltage difference of the power domain crossing.</p> <p>For example, the simulator reports an incorrect level shifter cell if the power domain crossing requires a low_to_high level shifter cell and you specify the level shifter strategy in the UPF file as high_to_low.</p> <p>Mnemonic: LS_INCORRECT</p> <p>Warning: (vopt-9693)  [UPF_LS_STATIC_CHK] Found Total 1 Incorrect level shifters.</p>
Valid level shifter cell	vopt -pa_checks=svl	<p>Reports valid level shifter cell when a level shifter cell is required for the power domain crossing, and either a level shifter cell is present in the design or the simulator infers one.</p> <p>Mnemonic: LS_VALID</p> <p>Note: (vopt-9693) [UPF_LS_STATIC_CHK] Found Total 2 Valid level shifters</p>

**Table 5-4. Static RTL Level Shifter Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
Not analyzed level shifter cell	vopt -pa_checks=snl	Reports not analyzed level shifter cell when the power state table (PST) information is not sufficient to analyze whether a level shifter cell is required or not for the power domain crossing. <b>Mnemonic:</b> LS_NOT_ANALYZED <b>Warning:</b> (vopt-9693) [UPF_LS_STATIC_CHK] Found Total 1 Not analyzed level shifters.
Not inserted level shifter cell	vopt -pa_checks=sdl	Reports not inserted level shifter cell when you specify the level shifter strategy with the <b>set_level_shifter -no_shift</b> command in the UPF file. However, a level shifter cell may or may not be required for the power domain crossing. <b>Mnemonic:</b> LS_NOT_INSERTED <b>Warning:</b> (vopt-9693) [UPF_LS_STATIC_CHK] Found Total 1 Not inserted level shifters.
All static RTL level shifter checks	vopt -pa_checks=sl	Performs all static RTL level shifter checks (sml, srl, sil, svl, snl, and sdl).

## Related Topics

[Level Shifter Checks](#)

[Static Check Report](#)

## Static RTL Path Analysis Checks

Use static RTL path analysis checks to report any not analyzed path or good paths in your RTL design.

To enable static RTL path analysis checks, specify a value to the vopt -pa\_checks command as shown in [Table 5-5](#).

To write the results of static RTL path analysis check to the report file, *report.static.txt*, use the -pa\_enable=reportcrossings argument with the vopt command.

**Table 5-5. Static RTL Path Analysis Checks**

Check	Usage Syntax	Description Mnemonic Example Message
Not analyzed path for level shifter requirement	vopt -pa_checks=snpl	<p>Reports not analyzed path for level shifter requirement when the power domain crossing is not analyzed for the level shifter requirements because of insufficient PST information on either the source or sink supplies.</p> <p>Mnemonic:  <b>LS_PATH_NOT_ANALYZED</b></p> <p>Warning: (vopt-9939) [ UPF_LS_STATIC_CHK ] Found Total 35 Not analyzed path for LS requirement.</p>
Good path with no level shifter requirement	vopt -pa_checks=scpl	<p>Reports good path with no level shifter requirement when the power domain crossing is analyzed for the level shifter requirements, and no level shifter cell is required for the crossing.</p> <p>Mnemonic: <b>LS_PATH_GOOD</b></p> <p>Note: (vopt-9938) [ UPF_LS_STATIC_CHK ] Found Total 3 Valid Path with no LS requirement.</p>
Not analyzed path for isolation requirement	vopt -pa_checks=snpi	<p>Reports not analyzed path for isolation requirement when the power domain crossing is not analyzed for the isolation requirements because of insufficient PST information on either the source or sink supplies.</p> <p>Mnemonic: <b>ISO_PATH_NOT_ANA LYZED</b></p> <p>Warning: (vopt-9939) [ UPF_ISO_STATIC_CHK ] Found Total 35 Not analyzed path for ISO requirement.</p>

**Table 5-5. Static RTL Path Analysis Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
Good path with no isolation requirement	<code>vopt -pa_checks=scpi</code>	Reports good path with no isolation requirement when the power domain crossing is analyzed for the isolation requirements, and no isolation cell is required for the crossing.  Mnemonic: ISO_PATH_GOOD  Note: (vopt-9938) [ UPF_ISO_STATIC_CHK ] Found Total 3 Valid Path with no ISO requirement.
Not analyzed path for isolation and level shifter requirement	<code>vopt -pa_checks=snpl+snpi</code>	Reports not analyzed path for isolation and level shifter requirements when the power domain crossing is not analyzed for the isolation and level shifter requirements because of insufficient PST information on either the source or sink supplies.  Mnemonic: PATH_NOT_ANALYZED  Warning: (vopt-9939) [ UPF_STATIC_CHK ] Found Total 35 Not analyzed path for ISO/LS requirement.
Good path with no isolation and level shifter requirement	<code>vopt -pa_checks=scpl+scpi</code>	Reports good path with no isolation and level shifter requirements when the power domain crossing is analyzed for the isolation and level shifter requirements, and no isolation and level shifter cell are required for the crossing.  Mnemonic: PATH_GOOD  Note: (vopt-9938) [ UPF_STATIC_CHK ] Found Total 64 Valid Path with no ISO/LS requirement.

## Related Topics

[Path Analysis Checks](#)

[Static Check Report](#)

## Static RTL Back-to-Back Checks

Use static RTL back-to-back checks to report any back-to-back cells in your RTL design.

To enable static RTL back-to-back checks, specify a value to the `vopt -pa_checks` command as shown in [Table 5-6](#). All static RTL back-to-back check results are written to the report file, `report.static.txt`.

**Table 5-6. Static RTL Back-to-Back Checks**

Check	Usage Syntax	Description Mnemonic Example Message
Back-to-back isolation cells with same control signals	<code>vopt -pa_checks=s+b2b</code>	Reports back-to-back isolation cells with the same control signal for a power domain crossing.  Mnemonic: ISO_B2B_CTRL_SAME  Warning: (vopt-9956) [UPF_ISO_STATIC_CHK] Found Total 3 B2B ISO cells with same ctrl signals.
Back-to-back isolation cells with different control signals	<code>vopt -pa_checks=s+b2b</code>	Reports back-to-back isolation cells with different control signals for a power domain crossing.  Mnemonic: ISO_B2B_CTRL_DIFF  Warning: (vopt-9955) [UPF_ISO_STATIC_CHK] Found Total 4 B2B ISO cells with different ctrl signals.
Back-to-back isolation cells with unknown control signals	<code>vopt -pa_checks=s+b2b</code>	Reports back-to-back isolation cells with unknown control signals for a power domain crossing.  Mnemonic: ISO_B2B_CTRL_UNK  Warning: (vopt-9973) [UPF_ISO_STATIC_CHK] Found Total 3 B2B ISO cells.

## Related Topics

[Back-to-Back Checks](#)

[Static Check Report](#)

## Static GLS Checks

Static GLS checks validate your gate-level design with the power intent defined in the UPF file. For static GLS checks, the simulator analyzes any specified power state tables (PSTs), power states added on power domains (by an add\_power\_state command), and supply sets in the UPF file. The purpose of this analysis is to detect the power domain relative ON/OFF condition and the relative operating voltages.

Use various static GLS checks (isolation, level shifter, retention, switch, back-to-back, and miscellaneous) on your gate-level design.

**Note**

 Your gate-level design can either be gate-level netlists out of synthesis, or fully PG netlists out of place-and-route.

For static GLS checks, perform the following steps:

1. Specify an existing cell in the design as an isolation or a level shifter cell. This specification enables the simulator to identify that cell in the design.
2. Use the following argument with the vopt command to enable all static GLS checks:

```
vopt -pa_glschecks=s
```

This enables all static GLS checks, except static GLS back-to-back cell checks.

When you enable all static GLS checks, if state dependencies between two connected power domains are not present in the PST/add\_power\_state, then the Questa SIM simulator cannot determine power domain relative ON/OFF states statically. In this case, static GLS checks report strategies as *Not Analyzed* or *Unanalyzed*. If state dependencies between two connected power domains are present in the PST/add\_power\_state, then the Questa SIM simulator performs all static GLS checks.

Static GLS checks generate a vopt log file, and a detailed report file (*report.static.txt*):

- **vopt log file** — Summary of the static GLS check results.
- **Report file** — Detailed results of the static GLS check.

**Table 5-7. Static GLS Checks**

Check	Description
Static GLS Isolation Checks	Use static GLS isolation checks to report any missing or inconsistent isolation cells in your gate-level design.
Static GLS Level Shifter Checks	Use static GLS level shifter checks to report any missing or inconsistent level shifter cells in your gate-level design.

**Table 5-7. Static GLS Checks (cont.)**

Check	Description
<a href="#">Static GLS Retention Checks</a>	Use static GLS retention checks to report any inconsistent retention condition in your gate-level design.
<a href="#">Static GLS Back-to-Back Checks</a>	Use static GLS back-to-back checks to report any back-to-back cells in your gate-level design.
<a href="#">Static GLS Miscellaneous Checks</a>	Use static GLS miscellaneous checks to report any missing liberty attribute in your gate-level design.
<a href="#">Static GLS Switch Checks</a>	Use static GLS switch checks to report any missing or inconsistent switch cells in your gate-level design.

## Static GLS Isolation Checks

Use static GLS isolation checks to report any missing or inconsistent isolation cells in your gate-level design.

Specify an existing cell in the design as an isolation cell by any of the following methods. This type of isolation cell is recognized for static and dynamic isolation checks.

### Method 1

Specify the liberty attribute, `is_isolation_cell`, as part of the module definition to identify the isolation cell. Set the `isolation_cell_enable_pin = "TRUE"` on the cell enable pin.

For example:

```
(* is_isolation_cell = 1 *)
module ISO_AND(
    (*pg_type = "primary_power"*) input logic pwr_rail,
    (*pg_type = "primary_ground"*) input logic gnd_rail,
    (*isolation_cell_enable_pin = "TRUE"*) input logic en,
    (*isolation_cell_data_pin = "TRUE"*) input data,
    output logic out);
    assign out = (data & ~en);
endmodule
```

### Method 2

Add pragma information to the instantiation generated by the synthesis application to identify the isolation cell.

For example:

```
ISOLOD1BWP isoinst1(.I(o1), .Z(w1), .ISO(ctrl)); //synopsys isolation_upf
iso_PD_mid1+PD_mid1
```

`iso_PD_mid1` is the UPF strategy for the isolation cell, and `PD_mid1` is the power domain for the strategy specified.

### Method 3

Assign an isolation prefix or suffix string to the existing isolation cell, as specified by the UPF name\_format command.

For example:

```
ISO isoinst2_UPF_ISO(.I(o1), .Z(w1), .ISO(ctrl));
```

When you enable static GLS checks, the Questa SIM simulator performs the static GLS isolation checks listed in [Table 5-8](#). All static GLS isolation check results are written to the report file, *report.static.txt*.

**Table 5-8. Static GLS Isolation Checks**

Check	Description <b>Mnemonic</b> <b>Example Message</b>
Valid isolation cell and strategy	<p>Reports valid isolation cell and strategy when an isolation cell is required, and an isolation cell and strategy are present for the power domain crossing.</p> <p>Mnemonic: ISO_VALID_CELL_WITH_STRATEGY</p> <p>Note: (vopt-9452) [GLS_STATIC_CHK] Found Total 7 Valid Isolation cell and strategy.</p>
Valid isolation cell with no strategy	<p>Reports valid isolation cell with no strategy when an isolation cell is required, and an isolation cell is present in the design for the power domain crossing. However, the corresponding isolation strategy is not specified in the UPF file.</p> <p>Mnemonic: ISO_CELL_WITHOUT_STRATEGY</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 4 Isolation cell with NO strategy.</p>
Valid isolation strategy with no cell	<p>Reports valid isolation strategy with no cell when an isolation cell is required, and an isolation strategy is specified in the UPF file for the power domain crossing. However, the corresponding isolation cell is not present in the design and is inferred instead.</p> <p>Mnemonic: ISO_VALID_STRATEGY_NO_CELL</p> <p>Note: (vopt-9452) [GLS_STATIC_CHK] Found Total 2 Isolation strategy with NO cell.</p>

**Table 5-8. Static GLS Isolation Checks (cont.)**

Check	Description Mnemonic Example Message
Isolation cell with incorrect location	<p>Reports incorrect location of the isolation cell when an isolation cell is required, and an isolation cell is present in the design for the power domain crossing. However, the cell location does not match with the location specified in the UPF file.</p> <p>Mnemonic: ISO_INCORRECT_LOCATION</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 5 Isolation cell with incorrect location.</p>
Missing isolation cell	<p>Reports missing isolation cell when an isolation cell is required for the power domain crossing. However, an isolation cell is not present in the design nor did the simulator infer one.</p> <p>Mnemonic: ISO_MISSING_CELL</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 6 Missing Isolation cell.</p>
Not required isolation cell	<p>Reports not required isolation cell when an isolation cell is not required for the power domain crossing. However, either an isolation cell is present in the design or the simulator infers one.</p> <p>Mnemonic: ISO_NOT_REQUIRED</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 2 Not required Isolation cell.</p>
Not required isolation cell and strategy	<p>Reports not required isolation cell and strategy when an isolation cell is not required, but an isolation cell and strategy are present for the power domain crossing.</p> <p>Mnemonic: ISO_NOT_REQUIRED_CELL_AND_STRATEGY</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 4 Not required Isolation cell and strategy.</p>
Incorrect isolation strategy	<p>Reports incorrect isolation strategy when an isolation cell is required for the power domain crossing, but you specify the isolation strategy with <b>set_isolation -no_isolation</b> in the UPF file.</p> <p>Mnemonic: ISO_INCORRECT_STRATEGY</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 4 Isolation cell with NO strategy.</p>

**Table 5-8. Static GLS Isolation Checks (cont.)**

Check	Description Mnemonic Example Message
Valid isolation cell with control reaching data	<p>Reports valid isolation cell with control reaching data when isolation enable signals for the UPF strategy reaches the data pin of the cell.</p> <p>Mnemonic: ISO_CTRL_REACH_DATA</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 4 Isolation cell with control reaching data.</p>
Valid isolation cell with unreachable control	<p>Reports valid isolation cell with unreachable control when isolation enable signals for the UPF strategy does not reach the enable pin of the cell.</p> <p>Mnemonic: ISO_CTRL_UNREACHABLE</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 3 Isolation cell with unreachable control.</p>
Unanalyzed isolation cell	<p>Reports unanalyzed isolation cell when either of the following condition is met:</p> <ul style="list-style-type: none"> <li>The source or sink power domain lies in a hierarchy that is not specified in any power domain.</li> <li>PST information is insufficient for isolation analysis of the power domains.</li> </ul> <p>Mnemonic: ISO_UNANALYZED_CELL</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 8 Unanalyzed Isolation cell.</p>
Parallel isolation cell	<p>Reports parallel isolation cell on diverging signal in the design.</p> <p>Mnemonic: ISO_PARALLEL_CELLS</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 8 Parallel Isolation cells on diverging signal.</p>
Mismatching isolation clamp	<p>Reports mismatching isolation clamps when the clamp-value of the isolation cell does not match with the corresponding UPF strategy.</p> <p>Mnemonic: ISO_MISMATCHING_CLAMP</p> <p>Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 7 Mismatching clamp value.</p>

**Table 5-8. Static GLS Isolation Checks (cont.)**

Check	Description Mnemonic Example Message
Not required isolation cell for internally isolated IP pin	Reports not required isolation cell for the internally isolated IP pin when an isolation cell is present for an internally isolated IP pin. The internally isolated IP pin is specified in the Liberty file using the Liberty attribute <code>is_isolated</code> .  Mnemonic: ISO_REDUNDANT_ISOLATED_IP_PIN  Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 1 Not required Iso cell for internally isolated IP pin.
Isolation control signals not generated from always-on region	Reports isolation control signals that are not generated from the always-on region of the design.  Mnemonic: ISO_CTRL_NOT_ALWYS_ON  Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 4 ISO ctrl signal NOT generated from ALWAYSON region.
Unconnected or floating isolation control signals	Reports isolation control signal pins that are unconnected or floating in the design.  Mnemonic: ISO_CTRL_UNCONNECTED  Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 3 ISO ctrl signal NOT driven.
Isolation control signals driven by constants	Reports isolation control signal pins that are driven by constants in the design.  Mnemonic: ISO_CTRL_CONST_DRIVER  Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 3 ISO ctrl signal has CONST driver.

## Related Topics

[Isolation Checks](#)

[Static Check Report](#)

## Static GLS Level Shifter Checks

Use static GLS level shifter checks to report any missing or inconsistent level shifter cells in your gate-level design.

Specify an existing cell in the design as a level shifter cell by any of the following methods. This type of level shifter cell is recognized for static and dynamic level shifter checks.

## Method 1

Specify the liberty attribute, is\_level\_shifter, as part of the module definition to identify the level shifter cell.

For example:

```
(* is_level_shifter = 1 *)
module ls_buf(
    (*pg_type = "primary_power"*) input logic pwr_rail,
    (*pg_type = "primary_ground"*) input logic gnd_rail,
    (* level_shifter_data_pin = 1 *)input data,
    output logic out);
    assign out = (data);
endmodule
```

## Method 2

Assign a level-shifter prefix or suffix string to the existing level shifter cell, as specified by the UPF name\_format command.

For example:

```
LVLHLD1BWP lsinst2_UPF_LS(.I(w2), .Z(w4));
```

When you enable static GLS checks, the Questa SIM simulator performs the static GLS level shifter checks listed in [Table 5-9](#). All static GLS level shifter check results are written to the report file, *report.static.txt*.

**Table 5-9. GLS Static Level Shifter Checks**

Check	Description Mnemonic Example Message
Valid level shifter cell and strategy	Reports valid level shifter cell and strategy when a level shifter cell is required, and a level shifter cell and strategy of the required direction are present for the power domain crossing.  Mnemonic: LS_VALID_CELL_WITH_STRATEGY  Note: (vopt-9451) [GLS_STATIC_CHK] Found Total 7 Valid Level-Shifter cell and strategy.
Valid level shifter cell with no strategy	Reports valid level shifter cell with no strategy when a level shifter cell is required, and a level shifter cell of the required direction is present for the power domain crossing. However, the corresponding strategy is not specified in the UPF file.  Mnemonic: LS_CELL_WITHOUT_STRATEGY  Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 4 Level-Shifter cell with NO strategy.

**Table 5-9. GLS Static Level Shifter Checks (cont.)**

Check	Description Mnemonic Example Message
Valid level shifter strategy with no cell	<p>Reports valid level shifter strategy with no cell when a level shifter cell is required, and a level shifter strategy is specified in the UPF file for the power domain crossing. However, the corresponding level shifter cell is not present in the design and is inferred instead.</p> <p>Mnemonic: LS_VALID_STRATEGY_NO_CELL</p> <p>Note: (vopt-9451) [GLS_STATIC_CHK] Found Total 5 Level-Shifter strategy with NO cell.</p>
Level shifter cell with incorrect location	<p>Reports incorrect location of the level shifter cell when a level shifter cell is required, and a level shifter cell of the required direction is present for the power domain crossing. However, the cell location does not match with the location specified in the UPF file.</p> <p>Mnemonic: LS_INCORRECT_LOCATION</p> <p>Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 5 Level-Shifter cell with incorrect location.</p>
Level shifter cell with mismatching strategy	<p>Reports incorrect direction of level shifter cell when a level shifter cell is required, and a level shifter cell is present for the power domain crossing. However, the direction of level shifter cell mismatches with the one inferred from level shifter strategy specified in the UPF file. For example:</p> <p>Level shifter rule = LH  Inferred rule = high_to_low</p> <p>Mnemonic: LS_TYPE_MISMATCH</p> <p>Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 2 Level-Shifter cell with mismatching strategy.</p>

**Table 5-9. GLS Static Level Shifter Checks (cont.)**

Check	Description Mnemonic Example Message
Not required level shifter cell	<p>Reports not required level shifter cell for the power domain crossing when either of the following condition is met:</p> <ul style="list-style-type: none"> <li>• The source to sink power domain/supply set does not require a level shifter cell.</li> <li>• Source and sink supply are same.</li> </ul> <p>However, either a level shifter cell is present in the design or the tool infers one.</p> <p>Mnemonic: LS_NOT_REQUIRED_CELL</p> <p>Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 2 Not required Level-Shifter cell.</p>
Not required level shifter cell and strategy	<p>Reports not required level shifter cell and strategy for the power domain crossing when either of the following condition is met:</p> <ul style="list-style-type: none"> <li>• Source to sink power domain or supply set does not require a level shifter cell.</li> <li>• Source and sink supply are same.</li> </ul> <p>However, a level shifter cell and strategy is present for the power domain crossing.</p> <p>Mnemonic: LS_NOT_REQUIRED_CELL_AND_STRATEGY</p> <p>Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 2 Not required Level-Shifter cell and strategy.</p>
Missing level shifter cell	<p>Reports missing level shifter cell when a level shifter cell is required for the power domain crossing. However, neither a level shifter cell is present in the design nor did the tool infer one.</p> <p>Mnemonic: LS_MISSING_CELL</p> <p>Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 2 Missing Level-Shifter cell.</p>

**Table 5-9. GLS Static Level Shifter Checks (cont.)**

Check	Description Mnemonic Example Message
Not required level shifter cell and strategy (threshold)	<p>Reports not required level shifter cell and strategy when a level shifter cell is not required as the absolute voltage difference between the source and sink power domain/supply set is not greater than the threshold. However, a level shifter cell and strategy are present for the power domain crossing.</p> <p>Mnemonic: LS_REDUNDANT_THRESHOLD</p> <p>Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 5 Redundant Level-Shifter strategy (threshold-based).</p>
Not required level shifter strategy (threshold)	<p>Reports not required level shifter strategy when a level shifter cell is not required as the absolute voltage difference between the source and sink power domain/supply set is not greater than the threshold. However, a level shifter strategy is specified in the UPF file.</p> <p>Mnemonic: LS_REDUNDANT_THRESHOLD_STRATEGY</p> <p>Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 2 Redundant Level-Shifter strategy (threshold-based).</p>
Not required level shifter cell (threshold)	<p>Reports not required level shifter cell when a level shifter cell is not required as the absolute voltage difference between the source and sink power domain/supply set is not greater than the threshold. However, a level shifter cell is present in the design.</p> <p>Mnemonic: LS_REDUNDANT_THRESHOLD_CELL</p> <p>Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 5 Redundant Level-Shifter cell (threshold-based).</p>
Not analyzed level shifter cell	<p>Reports unanalyzed level shifter cell when either of the following condition is met:</p> <ul style="list-style-type: none"> <li>• The source or sink power domain lies in the hierarchy that is not specified in any power domain</li> <li>• PST information is insufficient for level shifter analysis of the power domains.</li> </ul> <p>Mnemonic: LS_UNANALYZED_CELL</p> <p>Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 1 Unanalyzed Level-Shifter cell.</p>

**Table 5-9. GLS Static Level Shifter Checks (cont.)**

<b>Check</b>	<b>Description</b> <b>Mnemonic</b> <b>Example Message</b>
Parallel level shifter cell	Reports parallel level shifter cells on diverging signal in the design.  Mnemonic: LS_PARALLEL_CELLS  Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 4 Parallel Level-Shifter Cells on diverging signal.

## Related Topics

[Level Shifter Checks](#)

[Static Check Report](#)

## Static GLS Retention Checks

Use static GLS retention checks to report any inconsistent retention condition in your gate-level design.

The Questa SIM simulator identifies the retention cells in the gate-level design under the following conditions:

- The gate-level design has retention cells instantiated.
- The UPF file conveys the signals on which retention is specified in the strategy for the set\_retention command.

For example:

```
set_retention ... -elements { bus[0] q_1 }
```

- The retention cell instantiated in the design has the liberty attribute retention\_cell.
- The retention cell instantiated in the design has a retention/save/restore pin as defined in the UPF file.
- There is no logic (other than isolation, level shifter, or buffer) between the retention cell and the signals specified by the set\_retention -elements option.

When you enable static GLS checks, the Questa SIM simulator performs the static GLS retention checks listed in [Table 5-10](#). All static GLS retention check results are written to the report file, *report.static.txt*.

**Table 5-10. Static GLS Retention Checks**

Check	Description Mnemonic Example Message
Valid retention cell with no strategy	<p>Reports valid retention cell with no retention strategy when the retention cell is not associated with any retention strategy.</p> <p>Mnemonic: RET_NO_STRATEGY_FOR_CELL</p> <p>Warning: (vopt-9453) [GLS_STATIC_CHK] Found Total 4 Retention cell with NO strategy.</p>
Valid retention strategy with no cell	<p>Reports valid retention strategy with no cell when retention strategy is specified in the UPF file. However, the corresponding retention cell is not present in the design.</p> <p>Mnemonic: RET_NO_CELL_FOR_STRATEGY</p> <p>Warning: (vopt-9453) [GLS_STATIC_CHK] Found Total 4 Retention strategy without cell.</p>
Retention cell with no strategy in the element list	<p>Reports retention cells that are not in the strategy element lists.</p> <p>Mnemonic: RET_NO_STRATEGY_ELEM_FOR_CELL</p> <p>Warning: (vopt-9453) [GLS_STATIC_CHK] Found Total 4 Retention cell not in strategy elements list.</p>
Retention control signals not generated from always-on region	<p>Reports retention control signals that are not generated from the always-on region in the design.</p> <p>Mnemonic: RET_CTRL_NOT_ALWYS_ON</p> <p>Warning: (vopt-9453) [GLS_STATIC_CHK] Found Total 4 Retention ctrl signal NOT generated from ALWAYSON region.</p>
Unconnected or floating retention control signals	<p>Reports retention control signals that are unconnected or floating in the design.</p> <p>Mnemonic: RET_CTRL_UNCONNECTED</p> <p>Warning: (vopt-9453) [GLS_STATIC_CHK] Found Total 4 Retention ctrl signal NOT driven.</p>
Retention control signals driven by constant	<p>Reports retention control signals that are driven by constants in the design.</p> <p>Mnemonic: RET_CTRL_CONST_DRIVER</p> <p>Warning: (vopt-9453) [GLS_STATIC_CHK] Found Total 4 Retention ctrl signal has CONST driver.</p>

## Related Topics

[Retention Checks](#)

[Static Check Report](#)

## Static GLS Back-to-Back Checks

Use static GLS back-to-back checks to report any back-to-back cells in your gate-level design.

To enable static GLS back-to-back checks, specify a value to the vopt -pa\_glschecks command as shown in [Table 5-11](#). All static GLS back-to-back check results are written to the report file, *report.static.txt*.

**Table 5-11. Static GLS Back-to-Back Checks**

Check	Usage Syntax	Description Mnemonic Example Message
Back-to-back isolation and level shifter cells	vopt -pa_glschecks=s+els	Reports back-to-back ISO-LS cells present in the design. They can be replaced with an ELS cell.  Mnemonic: ISO_LS_B2B_CELLS  Warning: (vopt-9450) [GLS_STATIC_CHK] Found Total 1 B2B ISO-LS cells. ELS cell suggested.
Back-to-back level shifter and isolation cells	vopt -pa_glschecks=s+els	Reports back-to-back LS-ISO cells present in the design. They can be replaced with an ELS cell.  Mnemonic: LS_ISO_B2B_CELLS  Warning: (vopt-9450) [GLS_STATIC_CHK] Found Total 4 B2B LS-ISO cells. ELS cell suggested.
Back-to-back level shifter cells	vopt -pa_glschecks=s+b2b	Reports back-to-back LS cells present in the design. They can be replaced with a single LS cell.  Mnemonic: LS_B2B_CELLS  Warning: (vopt-9451) [GLS_STATIC_CHK] Found Total 2 B2B LS cells. single cell suggested.

**Table 5-11. Static GLS Back-to-Back Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
Back-to-back isolation cells with same control signals	vopt -pa_glschecks=s+b2b	Reports back-to-back isolation cells with same control signal present in the design.  Mnemonic: ISO_B2B_CELLS_CTRL_SAME  Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 2 B2B ISO cells with same ctrl signals.
Back-to-back isolation cells with different control signals	vopt -pa_glschecks=s+b2b	Reports back-to-back isolation cells with different control signals present in the design.  Mnemonic: ISO_B2B_CELLS_CTRL_SAME  Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 3 B2B ISO cells with diff ctrl signals.
Back-to-back isolation cells with unknown control signals	vopt -pa_glschecks=s+b2b	Reports back-to-back isolation cells with unknown control signals present in the design.  Mnemonic: ISO_B2B_CELLS_CTRL_UNK  Warning: (vopt-9452) [GLS_STATIC_CHK] Found Total 4 B2B ISO cells.

## Related Topics

[Static Check Report](#)

## Static GLS Miscellaneous Checks

Use static GLS miscellaneous checks to report any missing liberty attribute in your gate-level design.

When you enable static GLS checks, the Questa SIM simulator performs the static GLS miscellaneous checks listed in [Table 5-12](#). The static GLS miscellaneous check results are written to the report file, *report.static.txt*.

**Table 5-12. Miscellaneous Checks**

Check	Description Mnemonic Example Message
Missing liberty attribute	Reports missing liberty attributes for the design cells. Mnemonic: CELL_MISSING_LIB_ATT  Error (suppressible): (vopt-9455) [GLS_STATIC_CHK] Attribute 'related_ground_pin' not defined for port 'set' of instance '/top2/top_tb/ type_1_inst_2'.

## Related Topics

[Static Check Report](#)

## Static GLS Switch Checks

Use static GLS switch checks to report any missing or inconsistent switch cells in your gate-level design.

When you enable static GLS checks, the Questa SIM simulator performs the static GLS switch checks listed in [Table 5-13](#). All static GLS switch check results are written to the report file, *report.static.txt*.

**Table 5-13. Static GLS Switch Checks**

Check	Description Mnemonic Example Message
Switch control signals not generated from the always-on region	Reports switch control signals that are not generated from the always-on region of the design. Mnemonic: SWTCH_CTRL_NOT_ALWYS_ON  Warning: (vopt-9454) [GLS_STATIC_CHK] Found Total 2 SWITCH ctrl signal NOT generated from ALWAYSON region.
Unconnected or floating switch control signals	Reports switch control signals that are unconnected or floating in the design. Mnemonic: SWTCH_CTRL_UNCONNECTED  Warning: (vopt-9454) [GLS_STATIC_CHK] Found Total 3 SWITCH ctrl signal NOT driven.

**Table 5-13. Static GLS Switch Checks (cont.)**

<b>Check</b>	<b>Description</b> <b>Mnemonic</b> <b>Example Message</b>
Switch control signals driven by constant	Reports switch control signals that are driven by constants in the design. Mnemonic: SWTCH_CTRL_CONST_DRIVER Warning: (vopt-9454) [GLS_STATIC_CHK] Found Total 2 SWITCH ctrl signal has CONST driver.

## Related Topics

[Static Check Report](#)

## Dynamic Checks

During Power Aware simulation, dynamic checks validate your design with the power intent defined in the UPF file.

Use various dynamic checks (isolation, level shifter, retention, and miscellaneous) on your design.

To enable all dynamic checks, specify the argument `-pa_checks=d` to the `vopt` command. Add the `-pa_checksoption=assertionhierpath` argument to your `vopt` command to display the full hierarchical path of any failing check in the dynamic report.

**Table 5-14. Dynamic Checks**

Check	Description
Dynamic Isolation Checks	Use dynamic isolation checks to report any missing or inconsistent isolation cells in your design.
Dynamic Level Shifter Checks	Use dynamic level shifter checks to report any missing or inconsistent level shifter cells in your design.
Dynamic Retention Checks	Use dynamic retention checks to report any inconsistent retention condition in your design.
Dynamic Miscellaneous Checks	Specify a value to the <code>vopt -pa_checks</code> command to enable dynamic miscellaneous checks.

## Dynamic Isolation Checks

Use dynamic isolation checks to report any missing or inconsistent isolation cells in your design.

To enable dynamic isolation checks, specify a value to the `vopt -pa_checks` command as shown in [Table 5-15](#). These checks trigger error messages when a particular isolation strategy fails or a noise is detected in isolating a particular hierarchy. All dynamic isolation check results are written to the transcript window.

**Table 5-15. Dynamic Isolation Checks**

Check	Usage Syntax	Description Mnemonic Example Message
Isolation clamp value	vopt -pa_checks=icp	Flags violation if the clamp value of the isolation cell is different from the clamp value specified in the UPF file during the active isolation period.  Mnemonic: QPA_ISO_CLAMP_CHK  Error: (vsim-8930) QPA_ISO_CLAMP_CHK: Time: 40 ns, Isolated port for isolation cell (strategy: iso_PD_mid2_2) on port '/tb/TOP/mid2/in2_bot' having value (x) is different from clamp value (1) during isolation period. # File: test.upf, Line: 125, Power Domain:PD_mid2
Isolation disable protocol	vopt -pa_checks=idp	Flags violation if the isolation control signal is disabled when the source domain is OFF and sink domain is ON.  Mnemonic: QPA_ISO_DIS_PG  Error: (vsim-8919) QPA_ISO_DIS_PG: Time: 250 ns, Isolation control is disabled during power shut OFF (0) for the following: # Port: /tb/TOP/mid3/in2_bot[3:2]. # File: test.upf, Line: 113, Power Domain:PD_mid3
Isolation enable protocol	vopt -pa_checks=iep	Flags violation if the isolation control signal is not enabled when the source domain is OFF and sink domain is ON.  Mnemonic: QPA_ISO_EN_PSO  Error: (vsim-8918) QPA_ISO_EN_PSO: Time: 358 ns, Isolation control (0) is not enabled when power is switched OFF for the following: # Port: /tb_25/FA4_inst/ FA_inst2/d. # File: test.upf, Line: 91, Power Domain:PD_mid1

**Table 5-15. Dynamic Isolation Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
Isolation disable protocol check for COA states	vopt -pa_checks=idpcoa	<p>Flags violation if the isolation control signal is disabled when the source domain is in any of the following CORRUPT simstates, and the sink domain is not in a CORRUPT simstate:</p> <ul style="list-style-type: none"> <li>• CORRUPT</li> <li>• CORRUPT_ON_ACTIVITY</li> <li>• CORRUPT_STATE_ON_CHANGE</li> <li>• CORRUPT_STATE_ON_ACTIVITY</li> </ul> <p>Mnemonic: QPA_ISO_DIS_PG</p> <pre>Error: (vsim-8919) QPA_ISO_DIS_PG: Time: 60 ns, Isolation control is disabled during driving supply CORRUPT_ON_ACTIVITY simstate (1) for the following: # Port: /tb/TOP/mid1/out1_bot. # File: test.upf, Line: 62, Power Domain:PD_mid1</pre>
Isolation enable protocol check for COA states	vopt -pa_checks=iepcoa	<p>Flags violation if the isolation control signal is not enabled when a source domain is in any of the following CORRUPT simstates, and the sink domain is not in a CORRUPT simstate:</p> <ul style="list-style-type: none"> <li>• CORRUPT</li> <li>• CORRUPT_ON_ACTIVITY</li> <li>• CORRUPT_STATE_ON_CHANGE</li> <li>• CORRUPT_STATE_ON_ACTIVITY</li> </ul> <p>Mnemonic: QPA_ISO_EN_PSO</p> <pre>Error: (vsim-8918) QPA_ISO_EN_PSO: Time: 20 ns, Isolation control (x) is not enabled when driving supply is in CORRUPT_ON_ACTIVITY simstate for the following: # Port: /tb/TOP/mid1/out1_bot. # File: test.upf, Line: 62, Power Domain:PD_mid1</pre>

**Table 5-15. Dynamic Isolation Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
Isolation functionality	vopt -pa_checks=ifc	<p>Flags violation if the value at the output of an isolation cell is different from that at its input during the inactive isolation period.</p> <p>Mnemonic: QPA_ISO_FUNC_CHK</p> <pre>Error: (vsim-8931) QPA_ISO_FUNC_CHK: Time: 100 ns, Isolated port for isolation cell (strategy: iso_PD_mid2) on port '/tb/TOP/mid2/in2_bot' having value (x) is different from port value (1) during non-isolation period. # File: test.upf, Line: 111, Power Domain:PD_mid2</pre>
Isolation redundant activity	vopt -pa_checks=ira	<p>Flags violation if there is no requirement of isolation (such as the source domain is not in an OFF or CORRUPT state when the sink domain is in an ON state), and the isolation control signal is active.</p> <p>Mnemonic: QPA_ISO_REDUNDANT_ACT</p> <pre>Error: (vsim-8934) QPA_ISO_REDUNDANT_ACT: Time: 430 ns, Redundant activity on isolation control signal, for crossing {(PD_mid1) =&gt; (PD_mid2)}. # Port: /tb/TOP/mid1/out1_bot. # File: test.upf, Line: 44, Power Domain:PD_mid1</pre>
Isolation race	vopt -pa_checks=irc	<p>Flags violation if the value on the isolated port changes when its isolation control signal is changing state (high_to_low or low_to_high).</p> <p>Mnemonic: QPA_ISO_PORT_TOGGLE</p> <pre>Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port for isolation cell (strategy: iso_PD_mid1) on port '/tb/TOP/mid1/out2_bot' toggled when its control signal is activated. # File: test.upf, Line: 70, Power Domain:PD_mid1</pre>

**Table 5-15. Dynamic Isolation Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
Isolation toggle	vopt -pa_checks=it	Flags violation if the value on the isolated port changes during the active isolation period.  Mnemonic: QPA_ISO_ON_ACT  Error: (vsim-8939) QPA_ISO_ON_ACT: Time: 300 ns, Isolated port for isolation cell (strategy: iso_PD_mid1) on port '/tb/TOP/mid1/out2_bot[1]' toggled when its control signal was active. # File: test.upf, Line: 95, Power Domain:PD_mid1
Missing isolation cell	vopt -pa_checks=umi	Flags violation if the isolation strategy is not specified for a power domain crossing whose source domain is ON and sink domain is OFF.  Mnemonic: QPA_UPF_MISSING_ISO_CHK  Error: (vsim-8929) QPA_UPF_MISSING_ISO_CHK: Time: 130 ns, Missing isolation cell for domain boundary, PD_mid1 => PD_wrapper2 for following: # Source port : /tb/TOP/mid1/out1_bot [ connected mask: "1" ] [ LowConn ] -> Sink port: /tb/TOP/mid2(wrapper1(wrapper2/in1 [ connected mask: "1" ] [ LowConn ] # File: test.upf, Line: 25, Power Domain:PD_mid1
All dynamic isolation checks	vopt -pa_checks=i	Enables the following dynamic isolation checks: iep, iepcoa, idp, idpcoa, irc, ira, icp, and ifc.

## Related Topics

[Isolation Checks](#)

[Dynamic UPF Report](#)

## Dynamic Level Shifter Checks

Use dynamic level shifter checks to report any missing or inconsistent level shifter cells in your design.

To enable dynamic level shifter checks, specify a value to the vopt -pa\_checks command as shown in [Table 5-16](#). All dynamic level shifter check results are written to the transcript window.

**Table 5-16. Dynamic Level Shifter Checks**

Check	Usage Syntax	Description Mnemonic Example Message
Missing level shifter	vopt -pa_checks=uml	<p>Flags violation if a level shifter strategy is not specified in the UPF file when a level shifter cell is required for the power domain crossing.</p> <p>Also, strategy specified with set_level_shift -no_shift in the UPF file are checked for any missing level shifter cells.</p> <p>Mnemonic: QPA_UPF_MISSING_LS_CHK</p> <pre># ** Error: (vsim-8915) QPA_UPF_MISSING_LS_CHK: Missing level shifters for domain boundary, pd_aon ( Operating Voltage: 2.000000 V ) =&gt; pd_top ( Operating Voltage: 1.000000 V ) for the following: #   Source port : /tb/TOP/bot4/out1_bot [ LowConn ] -&gt; Sink port: /tb/TOP/bot4/out1_bot [ HighConn ] #   Time: 90 ns  Scope: mspa_top.mspa_upf_top.mspa0_pd_tb.mspa3_pd_t op.mspa5_pd_aon. QPA_UPF_MISSING_LS_CHK_0 File: src/inc_ls_dyn_check1/top.upf Line: 4</pre>
Incorrect level shifter	vopt -pa_checks=uil	<p>Flags violation if the direction of level shifter cell specified in the UPF file does not match with the direction as per the voltage difference of the power domain crossing.</p> <p>For example, the simulator reports an incorrect level shifter cell if the power domain crossing requires a low_to_high level shifter cell and you specify the level shifter strategy in the UPF file as high_to_low.</p> <p>Mnemonic: QPA_UPF_INCORRECT_LS_CHK</p> <pre># ** Error: (vsim-8917) QPA_UPF_INCORRECT_LS_CHK: Shift mismatch for level shifter(my_ls_bot3, rule: low_to_high, Domain: pd_aon ) at domain boundary, pd_aon ( Operating Voltage: 2.000000 V ) =&gt; pd_top (Operating Voltage: 1.000000 V ) for following: #   Source port : /tb/TOP/bot3/out1_bot [ LowConn ] -&gt; Sink port: /tb/TOP/bot3/out1_bot [ HighConn ] #   Time: 90 ns  Scope: mspa_top.mspa_upf_top.msp a0_pd_tb.mspa3_pd_top.mspa5_pd_aon. my_ls_bot3. QPA_UPF_INCORRECT_LS_CHK_1 File: src/inc_ls_dyn_check1/top.upf Line:4</pre>

**Table 5-16. Dynamic Level Shifter Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
All dynamic level shifter checks	vopt -pa_checks=ul	Enables all dynamic level shifter checks.

## Operating Voltage for Dynamic Checking

In some cases, the dynamic checks report the operating voltage of one of the domains as 0. This happens when you have not changed the voltage on the primary power and ground pin of the domain and you are operating with default unknown voltage levels. It is recommended to change the operating voltages of such domains during simulation. Change the operating voltages by using the supply\_on or supply\_off commands defined in the UPF SystemVerilog package.

## Related Topics

[Level Shifter Checks](#)

[Dynamic UPF Report](#)

## Dynamic Retention Checks

Use dynamic retention checks to report any inconsistent retention condition in your design.

To enable dynamic retention checks, specify a value to the vopt -pa\_checks command as shown in [Table 5-17](#). All dynamic retention check results are written to the transcript window.

**Table 5-17. Dynamic Retention Checks**

Check	Usage Syntax	Description Mnemonic Example Message
Power off	vopt -pa_checks=rop	<p><b>Master-Slave Configuration</b> — Flags violation if the retention condition is not asserted when the power is switched off.</p> <p>Mnemonic: QPA_RET_OFF_PSO</p> <pre>Error: (vsim-8903) QPA_RET_OFF_PSO: Time: 243 ns, Retention condition (0) for the following retention elements in scope '/tb/top_vh' of power domain 'pd' is not asserted during power shut down: q_regvh. # File: ./src/rtc_chk_all/test.upf, Line: 10, Power Domain:pd</pre> <p><b>Balloon-Latch Configuration</b> — Flags violation if the retention condition is not asserted when the power is switched off.</p> <p>This check is not activated if the asynchronous set or reset is <i>active</i>.</p> <p>Mnemonic: QPA_RET_OFF_PSO</p> <pre>Error: (vsim-8903) QPA_RET_OFF_PSO: Time: 35 ns, Retention control (x) for the following retention elements of power domain 'PD' is not asserted during power shut down: #/tb/top/mid1/bot_latch1. # File: ./src/rtc_chk_all/test.upf, Line: 2, Power Domain:PD</pre>

**Table 5-17. Dynamic Retention Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
Power on	vopt -pa_checks=rpo	<p><b>Master-Slave Configuration</b> — Flags violation if there is an error in the sequence of triggering of the retention condition and power signal. For retention to succeed, the retention condition should be high at power down and power up. However, this check is triggered when that is not the case.</p> <p>Mnemonic: QPA_RET_PD_OFF</p> <pre>Error: (vsim-8904) QPA_RET_PD_OFF: Time: 183 ns, Power for domain: 'pd' is not ON (0) when retention is disabled for retention elements in scope '/tb/top_vh': q_regvh. # File: ./src/rtc_chk_all/test.upf, Line: 2, Power Domain:pd</pre> <p><b>Balloon-Latch Configuration</b> — Flags violation if there is an error in the sequence of triggering of the retention condition and power signal. For retention to succeed, the power should be high. However, this check is triggered when that is not the case.</p> <p>This check is not activated if the asynchronous set or reset is <i>active</i>.</p> <p>Mnemonic: QPA_RET_PD_OFF</p> <pre>Error: (vsim-8904) QPA_RET_PD_OFF: Time: 85 ns, Power for domain: 'PD' is not ON (0) when retention is enabled for retention elements: # /tb/top/mid1/bot_latch1. # File: ./src/rtc_chk_all/test.upf, Line: 2, Power Domain:PD</pre>

**Table 5-17. Dynamic Retention Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
Clock/latch enable	vopt -pa_checks=rcs	<p><b>Balloon-Latch Configuration</b> — Flags violation if the clock or latch enable is not at a certain value when the save and restore events take place. If a latch is enabled and can change its value, triggering retention can potentially cause race conditions in the stored value. This is also a check against such conditions.</p> <p>This check is for balloon-latch configuration only; it does not apply to master-slave (slave-alive) retention.</p> <p>Mnemonic: QPA_RET_CLK_STATE</p> <pre>Error: (vsim-8905) QPA_RET_CLK_STATE: Time: 85 ns, LatchEn is not at proper level: 'LOW' (1) for the retention element(s) of type: AHRLA of power domain: PD. # /tb/top/mid1/bot_latch1. # File: ./src/rtc_chk_all/test.upf, Line: 2, Power Domain:PD</pre> <pre>Error: (vsim-8905) QPA_RET_CLK_STATE: Time: 207 ns, Clock is not at proper level: 'LOW' (1) for the retention element(s) of type: CLRFF of power domain: PD. # /tb/top/mid1/bot_ff. # File: ./src/rtc_chk_all/test.upf, Line: 3, Power Domain:PD</pre>
Clock toggle	vopt -pa_checks=rsa	<p>Flags violation if the clock toggles during the retention period.</p> <p>Mnemonic: QPA_RET_SEQ_ACT</p> <pre>Error: (vsim-PA-8906) QPA_RET_SEQ_ACT: Time: 208 ns, clock toggled during retention period for retention element(s) : # /tb/top/mid1/bot_ff. # File: ./src/rtc_chk_all/test.upf, Line: 15, Power Domain:PD</pre>

**Table 5-17. Dynamic Retention Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Message
Retention condition off at power enable	vopt -pa_checks=rtcon	Flags violation if the retention condition is off when the power is enabled.  Mnemonic: QPA_RET_ON_RTC  Error: (vsim-8953) QPA_RET_ON_RTC: Time: 155 ns, Retention condition is not ON when power is enabled for retention strategy: 'pd_retention' # File: ./src/rtc_chk_all/test.upf, Line: 57, Power Domain:pd
Retention condition off at power disable	vopt -pa_checks=rtcoff	Flags violation if the retention condition is off when the power is disabled.  Mnemonic: QPA_RET_OFF_RTC  Error: (vsim-8954) QPA_RET_OFF_RTC: Time: 171 ns, Retention condition is not ON when power is disabled for retention strategy: 'pd_retention' # File: ./src/rtc_chk_all/test.upf, Line: 57, Power Domain:pd
Retention condition toggle	vopt -pa_checks=rtctog	Flags violation if the retention condition toggles during power down. Toggling of the retention condition during power down corrupts the retained value.  Mnemonic: QPA_RET_RTC_TOGGLE  Error: (vsim-8955) QPA_RET_RTC_TOGGLE: Time: 231 ns, Retention condition toggled during power down period for retention strategy: 'pd_retention' # File: ./src/rtc_chk_all/test.upf, Line: 57, Power Domain:pd
All retention condition checks	vopt -pa_checks=rtc	Enables the following retention condition checks: <ul style="list-style-type: none"><li>• -pa_checks=rtcon</li><li>• -pa_checks=rtcoff</li><li>• -pa_checks=rtctog</li></ul>
All dynamic retention checks	vopt -pa_checks=r	Enables all dynamic retention checks (rop, rpo, rcs, rsa, rtcon, rtcoff, and rtctog).

## Related Topics

[Retention Checks](#)

[Dynamic UPF Report](#)

## Dynamic Miscellaneous Checks

Specify a value to the vopt -pa\_checks command to enable dynamic miscellaneous checks.

[Table 5-18](#) describes each dynamic miscellaneous check. All dynamic miscellaneous check results are written to the transcript window.

**Table 5-18. Dynamic Miscellaneous Checks**

Check	Usage Syntax	Description Mnemonic Example Messages
Toggle	vopt -pa_checks=t	Flags violation if the input to a power domain toggles when the power domain is turned off.  Mnemonic: QPA_PD_OFF_ACT, QPA_PD_BIAS_ACT  # ** Error: (vsim-PA-8908) QPA_PD_OFF_ACT: Time: 36 ns, /tb/top/clk toggled during power down of power domain: PD  # ** Error: (vsim-PA-8907) QPA_PD_BIAS_ACT : Time: 140 ns, testbench.top_entity.deliver2.clk toggled during bias of power domain: PD_1
Control signal corruption	vopt -pa_checks=cp	Flags violation when the power signal to any power domain gets corrupted.  This check does not flag a violation when both the source and sink power domain supplies are off.  For UPF, this check is applicable to control ports of a switch, isolation enable signal of an isolation strategy, and retention save and restore signals of a retention strategy.  Mnemonic: QPA_CTRL_SIG_CRPT  # ** Error: (vsim-8901) QPA_CTRL_SIG_CRPT: Time: 240 ns, Control Signal '/tb/ pg_array[1]' is corrupted. (Current Value: x) #File: test.upf, Line: 29, Power Domain:PD_TOP
Power domain status	vopt -pa_checks=p	Reports the power domains that are switched on or off.  Mnemonic: QPA_PD_STATUS_INFO  ** Note: (vsim-8902) QPA_PD_STATUS_INFO: Time: 0 ns, Power domain 'PD_ALU' is BIASED down. (CORRUPT_ALL_ON_ACT)

**Table 5-18. Dynamic Miscellaneous Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Messages
Illegal or undefined state	vopt -pa_checks=pis	<p>Flags violation if there is an undefined or illegal state on a PST or supply port.</p> <p>A state on a PST is undefined or illegal if the state of nets or ports is not a valid combination as related to the power state table in the UPF file.</p> <p>A state on a supply port is undefined or illegal if it is not defined with add_port_state for the supply port.</p> <p>Mnemonic: <b>QPA_UPF_ILLEGAL_STATE_REACHED</b></p> <pre># ** Error: (vsim-8933) QPA_UPF_ILLEGAL_STATE_REACHED: Time: 80 ns, PST 'eth_pci_pst' reached an undefined state. # {tb_pow = (tb_nom : 4.800000 V), mid1_MAIN_NET = (undefined : 4.000000 V), mid2_MAIN_NET = (NOM2 : 5.300000 V)} # File: ./src/illegal_pst_2/test.upf, Line: 79, PST state:UNDEFINED</pre>
Glitch detection	vopt -pa_checks=ugc	<p>Reports any spurious spikes (glitches) on control lines so that it does not cause false switching of control ports of various control logic (such as isolation, power switch, and retention). Use the <b>pa msg -glitch_window</b> command to specify the maximum allowed time window of the glitch.</p> <p>Mnemonic: <b>QPA_CTRL_SIG_GLITCH</b></p> <pre>** Warning: (vsim-PA-8921) QPA_CTRL_SIG_GLITCH : Time: 192 ns, Glitch (* -&gt; 0 -&gt; 1 ) detected for signal(/tb/power1) acting as switch control for (/tb/PD_INV1,sw)</pre>

**Table 5-18. Dynamic Miscellaneous Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Messages
Non-retention register reset	vopt -pa_checks=npu	<ul style="list-style-type: none"> <li>Default behavior: Flags violation if the non-retention registers are not reset when the power domain containing them is powered up. Also generates the report file <i>report.nretsyncff.txt</i>. Mnemonic: QPA_NRET_ASYNCFF           <pre>** Error: (vsim-8912) QPA_NRET_ASYNCFF: Time: 12 ns, Asynchronous (set/reset) control for the following flop(s) of power domain 'PD1' is not asserted at power up: # /tb/top_inst/out1. # File: test.upf, Line: 4, Power Domain:PD1</pre> </li> <li>When you define the attribute <code>qpa_attr_inactive_reset_duration</code> and/or <code>qpa_attr_active_reset_duration</code> in the <code>set_design_attributes</code> command: Mnemonic: QPA_NRET_ASYNCFF           <ol style="list-style-type: none"> <li>Flags violation if the asynchronous reset is not asserted within <code>Tmax</code> time units after power-up. See “<a href="#">“qpa_attr_inactive_reset_duration”</a>”.               <pre># ** Error: (vsim-8912) QPA_NRET_ASYNCFF: Time: 16 ns, Asynchronous(set/reset) control for the following flop(s) in scope '/tb/ top_inst' of power domain 'PD0' is not asserted within 4 time-units after power up: # out2, # File: test.upf, Line:4, Power Domain:PD0</pre> </li> <li>Flags violation if the asynchronous reset is not asserted for minimum <code>Twidht</code> time units. See “<a href="#">“qpa_attr_active_reset_duration”</a>”.               <pre># ** Note: (vsim-8912) QPA_NRET_ASYNCFF: Time: 19 ns, Asynchronous(set/reset) control for the following flop(s) in scope '/tb/ top_inst' of power domain 'PD0' is deasserted within 3 time-units after getting asserted: # out2, # File: ./src/npu_chk3/test.upf, Line: 4, Power Domain:PD0</pre> </li> </ol> </li> </ul>

**Table 5-18. Dynamic Miscellaneous Checks (cont.)**

Check	Usage Syntax	Description Mnemonic Example Messages
Power on	vopt -pa_checks=upc	Flags violation if the isolation or retention supplies are switched off during the active isolation or retention period.  Mnemonic: QPA_UPF_PG_CHK  Error: (vsim-8920) QPA_UPF_PG_CHK: Time: 0 ns, Power for Isolation strategy: 'ISO_FA4_1_3' of power domain: 'PD_FA4_1' is switched OFF during isolation. # File: test.upf, Line: 121, Power Domain:PD_FA4_1

## Related Topics

[Dynamic UPF Report](#)

# Quick Reference of Static RTL and Dynamic Checks

The quick reference table provides values of the -pa\_checks argument for enabling various static RTL and dynamic checks.

**Table 5-19. Argument Values for Static RTL and Dynamic Checks**

Checks	Static RTL Value	Dynamic Value
<b>Isolation Cell Checks</b>		
Missing isolation cell check	-pa_checks=smi	-pa_checks=umi
Not required isolation cell	-pa_checks=sri	
Incorrect isolation cell	-pa_checks=sii	
Valid isolation cell	-pa_checks=svi	
Not analyzed isolation cell	-pa_checks=sni	
Not inserted isolation cell	-pa_checks=mdi	
Isolation clamp value		-pa_checks=icp
Isolation disable protocol		-pa_checks=idp
Isolation enable protocol		-pa_checks=iep
Isolation disable protocol check for COA states		-pa_checks=idpcoa

**Table 5-19. Argument Values for Static RTL and Dynamic Checks (cont.)**

Checks	Static RTL Value	Dynamic Value
Isolation enable protocol check for COA states		-pa_checks=iepcoa
Isolation functionality		-pa_checks=ifc
Isolation redundant activity		-pa_checks=ira
Isolation race		-pa_checks=irc
Isolation toggle		-pa_checks=it
<b>Level Shifter Cell Checks</b>		
Missing level shifter cell	-pa_checks=sml	-pa_checks=uml
Not required level shifter cell	-pa_checks=srl	
Incorrect level shifter cell	-pa_checks=sil	-pa_checks=uil
Valid level shifter cell	-pa_checks=svl	
Not analyzed level shifter cell	-pa_checks=snl	
Not inserted level shifter cell	-pa_checks=sdl	
<b>Retention Cell Checks</b>		
Power off		-pa_checks=rop
Power on		-pa_checks=rpo
Clock/latch enable		-pa_checks=rcs
Retention condition off at retention enable		-pa_checks=rtcon
Retention condition off at retention disable		-pa_checks=rtcoff
Retention condition toggle		-pa_checks=rtctog
<b>Path Analysis Checks</b>		
Not analyzed path for level shifter requirement	-pa_checks=snpl	
Good path with no level shifter requirement	-pa_checks=scpl	
Not analyzed path for isolation requirement	-pa_checks=snpi	
Good path with no isolation requirement	-pa_checks=scpi	

**Table 5-19. Argument Values for Static RTL and Dynamic Checks (cont.)**

Checks	Static RTL Value	Dynamic Value
Not analyzed path for isolation and level shifter requirement	-pa_checks=snpl+snpi	
Good path with no isolation and level shifter requirement	-pa_checks=scpl+scpi	
<b>Back-to-Back Cells Check</b>		
Back-to-back isolation cells	-pa_checks=s+b2b	
<b>Miscellaneous Checks</b>		
Toggle		-pa_checks=t
Control signal corruption		-pa_checks=cp
Power domain status		-pa_checks=p
Illegal or undefined state		-pa_checks=pis
Glitch detection		-pa_checks=ugc
Non-retention register reset		-pa_checks=npu
Power on		-pa_checks=upc

## Power Aware Checks Control

Power Aware checks may require granular control than is provided by the values of the vopt -pa\_checks command or the arguments of the pa msg command. For granular control of the checks, create a Tcl configuration file that contains the pa\_checks or the pa msg commands, and then use this file to selectively enable or disable specific checks for a design or UPF object (such as isolation, level shifter, and retention).

The following examples demonstrate scenarios when you require granular control of Power Aware checks:

- You want to enable all Power Aware checks, except for the missing isolation cell check. To achieve this, you must specify each individual value for the -pa\_checks argument, except -pa\_checks=umi. There is no alternative using the existing values of the vopt -pa\_checks command.
- You want to enable a particular check for all design and UPF objects, except for a few. For example, you cannot use the vopt -pa\_checks command to enable the retention checks for all retention-strategies, domain, supply-set, design-element, and models, except for a selected one or a set of selected ones. Similarly, you cannot specify exceptions for the source, sink, PST name, or switch name.
- You want to control the Power Aware checks based on certain expressions becoming *true*. For example, you want to enable the retention checks only when the retention state control signal is high.

Use the following commands for granular control of checks:

- **pa\_checks** — Specify the pa\_checks commands in a Tcl configuration file, and use this file with the vopt -pa\_tclfile command. The pa\_checks commands in the Tcl configuration file selectively enable or disable specific static RTL and dynamic checks for a specific design or UPF object.

If there is a conflict between two or more pa\_checks commands, the simulator resolves the priority of commands based on the precedence of arguments of the pa\_checks command. Refer to “[pa\\_checks](#)” for more information on the arguments of the pa\_checks command.

- **pa msg** — Specify the pa msg commands in a Tcl configuration file, and use this file at the VSIM prompt. The pa msg commands in the Tcl configuration file selectively enable or disable specific dynamic checks for a specific design or UPF object.

If there is a conflict between two or more pa msg commands, the simulator resolves the priority of commands based on the precedence of arguments of the pa msg command. Refer to “[Command to Control Dynamic Checks](#)” for more information on the arguments of the pa msg command.

**Controlling Checks With the pa\_checks Command.....** [197](#)

Controlling Checks With the pa msg Command .....	<a href="#">199</a>
Precedence Order of Arguments .....	<a href="#">201</a>
Pathname Convention in Arguments.....	<a href="#">202</a>

## Controlling Checks With the pa\_checks Command

Enable granular control of static RTL and dynamic checks by using a Tcl configuration file, containing the pa\_checks commands, with the vopt command.

### Procedure

1. Create a Tcl configuration file that contains one or more pa\_checks commands.
2. Include the -pa\_tclfile argument in the vopt command (instead the -pa\_checks argument) to invoke the Tcl configuration file.

This applies the pa\_checks commands to your Power Aware design (the same as by specifying vopt -pa\_checks <values>).

For example, the following vopt command specifies a Tcl configuration file, named *chks\_config*, which contains the pa\_checks commands:

```
vopt -pa_upf ./test.upf tb -o tbout -pa_tclfile ./chks_config
```

3. Save the configuration of static RTL and dynamic checks performed by the Questa SIM simulator, using the following command in a UPF file or a Tcl configuration file:

```
save_checks_config <file_name>
```

### Examples

The following example compares the error messages reported from the Power Aware checks with no controls to Power Aware checks with controls.

## Power Aware Checks With No Controls

When you do not control the Power Aware checks, the following error messages appear in the transcript window:

```
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_1) on port '/tb/TOP/midi1/
out1_bot[4]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_1) on port '/tb/TOP/midi1/
out1_bot[2]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_2) on port '/tb/TOP/midi1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_1) on port '/tb/TOP/midi1/
out1_bot[4]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_1) on port '/tb/TOP/midi1/
out1_bot[2]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns,
Isolated port for isolation cell (strategy: iso_PD_midi_2) on port '/tb/
TOP/midi1/out2_bot[4:5]' toggled when its control signal is activated.
```

## Power Aware Checks With Controls

Assume you are interested in the error messages of the QPA\_ISO\_PORT\_TOGGLE check that corresponds only to the following element:

```
/tb/TOP/midi1/out2_bot
```

To decrease the reported error messages to those of interest, use the following vopt command with the -pa\_tclfile argument to specify a Tcl configuration file, *chks\_config*:

```
vopt -pa_upf ./test.upf tb -o tbout -pa_tclfile ./chks_config
```

where the *chks\_config* file contains the following pa\_checks commands:

```
pa_checks -disable -checkIds {all} set_scope /tb/TOP
pa_checks -enable -checkIds {QPA_ISO_PORT_TOGGLE} -elements {midi1/
out2_bot}
```

These pa\_checks commands disable all Power Aware checks, except the QPA\_ISO\_PORT\_TOGGLE check with the element {mid1/out2\_bot}.

When you control Power Aware checks, the desired error messages appear in the transcript window:

```
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_2) on port '/tb/TOP/midi1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_2) on port '/tb/TOP/midi1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 110 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_2) on port '/tb/TOP/midi1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_midi
```

## Related Topics

[pa\\_checks](#)

[save\\_checks\\_config](#)

# Controlling Checks With the pa msg Command

Enable granular control of dynamic checks by invoking a Tcl configuration file, containing the pa msg commands, at the VSIM prompt. You can also enter one or more pa msg commands directly at the VSIM prompt.

## Procedure

1. Create a Tcl configuration file that contains one or more pa msg commands.
2. Invoke the Tcl configuration file at the VSIM prompt.

This applies the pa msg commands to the Power Aware simulation.

For example, the following command specifies a Tcl configuration file, named *chks\_config*, which contains the pa msg commands:

```
VSIM 1> do ./chks_config
```

You can also enter the pa msg commands directly at the VSIM prompt.

For example:

```
VSIM 1> pa msg -enable -all
VSIM 2> pa msg -disable -checkIDs {iepcoa idpcoa}
```

## Examples

The following example compares the error messages reported from Power Aware checks with no controls to Power Aware checks with controls.

## Power Aware Checks With No Controls

When you do not control Power Aware checks, the following error messages appear in the transcript window:

```
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_1) on port '/tb/TOP/midi1/
out1_bot[4]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_1) on port '/tb/TOP/midi1/
out1_bot[2]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_2) on port '/tb/TOP/midi1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_1) on port '/tb/TOP/midi1/
out1_bot[4]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_1) on port '/tb/TOP/midi1/
out1_bot[2]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns,
Isolated port for isolation cell (strategy: iso_PD_midi_2) on port '/tb/
TOP/midi1/out2_bot[4:5]' toggled when its control signal is activated.
```

## Power Aware Checks With Controls

Assume you are interested in the error messages of the QPA\_ISO\_PORT\_TOGGLE check that corresponds only to the following element:

```
/tb/TOP/midi1/out2_bot
```

To decrease the reported error messages to those of interest, invoke a Tcl configuration file,*chks\_config*, at the VSIM prompt:

```
VSIM 1> do ./chks_config
```

where the *chks\_config* file contains the following pa msg commands:

```
pa msg -disable -checkIds {d}
pa msg -enable -checkIds {QPA_ISO_PORT_TOGGLE} -elements {midi1/out2_bot} -
scope /tb/TOP
```

These pa msg commands disable all Power Aware checks, except the QPA\_ISO\_PORT\_TOGGLE check with the element {mid1/out2\_bot}.

When you control Power Aware checks, the desired error messages appear in the transcript window:

```
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_2) on port '/tb/TOP/midi1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_2) on port '/tb/TOP/midi1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_midi
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 110 ns, Isolated port
for isolation cell (strategy: iso_PD_midi_2) on port '/tb/TOP/midi1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_midi
```

## Related Topics

[Command to Control Dynamic Checks](#)

# Precedence Order of Arguments

If a Tcl configuration file contains multiple pa\_checks or pa msg commands that apply to the same check, then the simulator resolves the priority of the commands based on the precedence of arguments.

The precedence order of the arguments is as follows:

1. Source Sink Ports — For crossing-based checks, either of the source or sink port matches with the port(s) specified in the -elements argument.
2. Source Sink Domain/Supply — For crossing-based checks, either of the source or sink domain/supply matches with the domain/supply specified in the -source or the -sink argument.
3. Strategy — A strategy matches with the strategy specified in the -strategies argument.
4. Domain — A power domain matches with the domain specified in the -domains argument.
5. Source Sink Scopes — For crossing based checks, either of the source or sink instance hierarchies lie under hierarchy specified in the -elements argument.

For non-crossing based checks, the parent scope lies under hierarchy specified in the -elements argument.

If the -transitive argument is TRUE, then the simulator searches for hierarchies recursively inside the specified hierarchy; otherwise, it looks in the specified hierarchy only.

6. Modules — For crossing-based checks, the module of either the source or sink matches with the module specified in the -models arguments.

For non-crossing based checks, the module of a parent scope matches with the module specified in the -models arguments.

## Example

Assume you include the following two commands in the same Tcl configuration file:

- `pa msg -enable -checkIds {umi} -sources {PD1}`  
where you enable the dynamic missing isolation cell check, which originates from the PD1 power domain.
- `pa msg -disable -checkIds {umi} -elements {top/inst1} -transitive TRUE`  
where you disable the dynamic missing isolation cell check if either the source/sink scope lies inside the top/inst1 hierarchy.

In this case, the simulator enables the dynamic missing isolation check because the precedence of source sink supply is higher than the source sink scopes. Hence, if there is a missing isolation cell from the PD1-to-PD2 crossing, then the simulator reports it, because it matches the specified -sources argument (in the first command)—even if it lies inside the hierarchy top/inst, for which the second command disable the checks.

For crossings that do not start with PD1, the simulator applies the second command according to the precedence order.

## Related Topics

[pa\\_checks](#)

[Command to Control Dynamic Checks](#)

# Pathname Convention in Arguments

Use either the absolute or relative pathname to specify the design or UPF objects in the arguments of the commands: pa\_checks and pa msg.

- Absolute pathname — If the pathname starts with a forward slash (/), it is treated as an absolute pathname.

Example with the pa\_checks command:

```
pa_checks ... -elements {/dut/inst1/bot/mid1}
```

Example with the pa msg command:

```
pa msg ... -elements {/tb/TOP}
```

- Relative pathname — If the pathname does not start with a forward slash (/), it is treated as a relative pathname.

When using the pa\_checks or pa msg command, relative pathnames are established with respect to the active scope of this command. By default, the active scope is the design with which the command is invoked.

Change the active scope of a command by using the set\_scope UPF command at the vopt command line and the -scope argument to the pa msg command at the VSIM prompt.

For example, to specify a pathname relative to the design scope of inst1 at the vopt command line:

```
set_scope inst1
pa_checks ... -elements {bot/mid1}
```

where bot/mid1 is searched inside /dut inst1 scope. The behavior of the set\_scope command is same as it is in the UPF file.

For example, to specify a pathname relative to the design scope of tb at the VSIM prompt:

```
pa msg ... -elements {TOP} -scope /tb
```

where TOP is searched inside the scope of /tb.

## Related Topics

[pa\\_checks](#)

[Command to Control Dynamic Checks](#)

[set\\_scope](#)



# Chapter 6

## Power Aware Coverage

---

Analyze Power Aware coverage to verify that the regression test suites are adequately testing the Power Aware elements of your design.

<b>Power Aware Coverage Overview .....</b>	<b>205</b>
<b>Power Aware Coverage Flow .....</b>	<b>206</b>
Power Aware Coverage Collection.....	208
Generating UCDB and Test Plan Files.....	210
Power Aware Coverage Analysis .....	212
<b>Coverage Support of UPF Objects.....</b>	<b>223</b>

## Power Aware Coverage Overview

Power Aware coverage contains information about all dynamic checks, and power states and transitions of various UPF objects.

- **Dynamic Check Coverage** — Lets you know whether a particular dynamic check is performed during Power Aware simulation. A particular dynamic check is performed when the test vectors provide the correct stimuli. For example, you want to check the missing isolation cells in all power domain crossings. You enable the check with the `-pa_checks=umi` argument to `vopt`. To trigger the check during simulation, your test vector should create a stimuli such that the source is OFF and sink is ON. If the test vectors do not create this behavior, then the missing isolation cell check is not performed during simulation. In this case, verify that there are no missing isolation cells in the design.
- **Power State and Transition Coverage** — Lets you know whether all power states and transitions of various UPF objects, such as power domain, supply set, and power switch are covered during Power Aware simulation. See “[Coverage Support of UPF Objects](#)”.

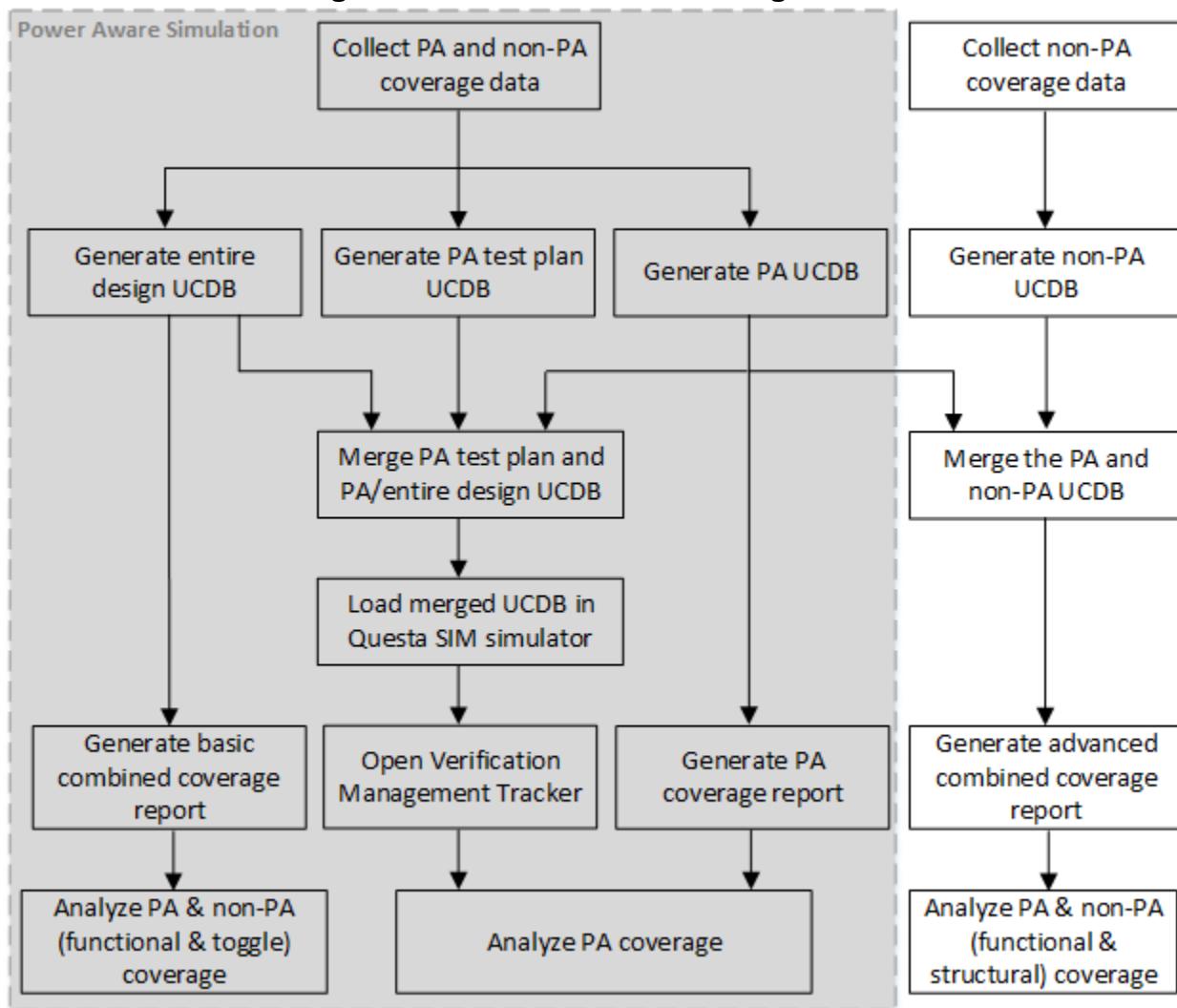
## Power Aware Coverage Flow

Configure a Power Aware simulation to collect Power Aware and non-Power Aware coverage data.

Use the collected coverage data to generate the following:

- **Power Aware Test Plan** — Contains information about Power Aware coverage of your design. View the test plan in an XML format or in the Verification Management Tracker window of the Questa SIM simulator.
- **Power Aware Coverage Report** — Contains information about Power Aware coverage of your design.
- **Basic Combined Coverage Report** — Contains information about Power Aware and non-Power Aware coverage of your design. Non-Power Aware coverage contains information about functional and toggle coverage.
- **Advanced Combined Coverage Report** — Contains information about Power Aware and non-Power Aware coverage of your design. Non-Power Aware coverage contains information about functional coverage, and all structural coverage - statement, expression, FSM, branch, and toggle.

**Figure 6-1. Power Aware Coverage Flow**



The installation directory provides a coverage example on an ALU design at the following location:

`<install_dir>/examples/pa_sim/example_five/`

For more information on the example, view the *README* file provided in the directory.

<b>Power Aware Coverage Collection.....</b>	<b>208</b>
<b>Generating UCDB and Test Plan Files .....</b>	<b>210</b>
<b>Power Aware Coverage Analysis .....</b>	<b>212</b>

## Power Aware Coverage Collection

The vopt and vsim commands provide arguments to collect Power Aware coverage data. If you have existing scripts to collect coverage data, alter them for Power Aware simulation on your design and test suite.

**Collecting Coverage Data of Dynamic Checks . . . . .** **208**

**Collecting Coverage Data of Power States and Transitions . . . . .** **209**

### Collecting Coverage Data of Dynamic Checks

Configure the standard Power Aware simulation flow to collect coverage data of dynamic checks.

#### Procedure

1. Compile using your existing command.
2. Optimize using your existing vopt command with the following arguments:
  - **-pa\_coverage=checks** — Collects Power Aware coverage data of dynamic checks.  
To enable a specific dynamic check, specify values to the -pa\_checks argument of the vopt command. See “[Dynamic Checks](#)”.
  - **-pa\_enable=autotestplan** — (optional) Enables generation of the Power Aware test plan.  
To enable hierarchical view of the power domains in the test plan, use the following argument:  
`-pa_enable=autotestplan+pdhieritestplan`

---

#### Tip

 To collect non-Power Aware coverage data, use the +cover argument with the vopt command. Use the non-Power Aware coverage data to generate the advanced combined coverage report.

---

3. Simulate using your existing vsim command with the following arguments:
  - **-unattemptedimmed** — (optional) Enables you to view any unattempted dynamic checks in the coverage data.

---

#### Note

 By default, a dynamic check is marked as covered when the check is attempted and it never fails (fail count = 0).

---

## Related Topics

- [Analyzing Coverage Using the Power Aware Test Plan](#)
- [Generating the Advanced Combined Coverage Report](#)

# Collecting Coverage Data of Power States and Transitions

Configure the standard Power Aware simulation flow to collect coverage data of power states and transitions of UPF objects, such as power domains, ports, and switch.

## Prerequisites

- Include power state information in your UPF file.

## Procedure

1. Compile using your existing command.
2. Optimize using your existing vopt command with the following arguments:
  - **-pa\_coverage** — Collects Power Aware coverage data of all dynamic checks, and power states and transitions of various UPF objects.
    - Specify a value to the -pa\_coverage argument to selectively collect coverage data. If you do not specify a value to the -pa\_coverage argument, then all values, except implicitportnet and undeftrans, are enabled.

For example:

```
vopt -pa_coverage=switch
vopt -pa_coverage=iso
```

For more information on the values of the -pa\_coverage argument, see “[vopt](#)” command in the *Questa SIM Command Reference Manual*.

---

### **Tip**

 For finer control on the coverage data, see “[describe\\_state\\_transition](#)” and “[describe\\_state\\_cross\\_coverage](#)”.

---

- To specify more than one value to the -pa\_coverage argument, use the plus sign (+) operator between the values.

For example:

```
vopt -pa_coverage=switch+iso
```

- To enable complete cross coverage, specify the following argument with your vopt command. See “[Cross Coverage of Power States](#)”.

```
vopt -pa_coverage=powerstate+crossdontcare
```

- **-pa\_enable=autotestplan** — (optional) Enables generation of the Power Aware test plan.

To enable hierarchical view of the power domains in the test plan, specify the following argument with your vopt command:

```
vopt -pa_enable=autotestplan+pdhieritestplan
```

---

**Tip**

 To collect non-Power Aware coverage data, use the **+cover** argument with the vopt command. Use the non-Power Aware coverage data to generate the advanced combined coverage report.

---

3. Simulate using your existing vsim command.

## Related Topics

[Analyzing Coverage Using the Power Aware Test Plan](#)

[Power State and Transition Concepts](#)

[Generating the Advanced Combined Coverage Report](#)

# Generating UCDB and Test Plan Files

Save the collected coverage data to Unified Coverage Database (UCDB) and test plan files to analyze coverage during post-simulation.

## Prerequisites

- Collect Power Aware coverage data of dynamic checks, and power states and transitions of all UPF objects. See “[Collecting Coverage Data of Power States and Transitions](#)”.
- Collect non-Power Aware coverage data. See “[Usage Flow for Code Coverage Collection](#)” in the *Questa SIM User’s Manual*.
- Enable generation of test plan. See “[Power Aware Coverage Collection](#)”.

## Procedure

1. Generate a Power Aware UCDB file, for example, *pa.ucdb*:

```
coverage save -pa pa.ucdb
```

2. Generate the entire design UCDB file, for example, *all.ucdb*, which contains Power Aware and non-Power Aware information.

```
coverage save all.ucdb
```

---

**Note**

 Non-Power Aware information contains functional and toggle coverage details only.

---

3. Generate the Power Aware test plan UCDB file, *QuestaPowerAwareTestplan.ucdb*. The test plan file is generated at the *pa\_reports* directory. See “[pa autotestplan](#)” in the *Questa SIM Command Reference Manual*.

```
pa autotestplan [-format] [-filename]
```

## Related Topics

[Power Aware Coverage Collection](#)

[Analyzing Coverage Using the Power Aware Test Plan](#)

[Questa Testplan Creation Using OpenOffice](#)

[Questa Testplan Creation Using Excel Add-In](#)

## Power Aware Coverage Analysis

Use the coverage reports and test plan to analyze Power Aware and non-Power Aware coverage of your design.

<b>Generating the Power Aware Coverage Report .....</b>	<b>212</b>
<b>Analyzing Coverage Using the Power Aware Test Plan .....</b>	<b>214</b>
<b>Generating the Basic Combined Coverage Report.....</b>	<b>220</b>
<b>Generating the Advanced Combined Coverage Report.....</b>	<b>221</b>

### Generating the Power Aware Coverage Report

Generate the Power Aware coverage report either during live-simulation or post-simulation to analyze Power Aware coverage. Power Aware coverage contains information about dynamic checks coverage, and power states and transitions coverage.

#### Prerequisites

- Generate the Power Aware UCDB file, for example, *pa.ucdb*. See step 1 in the “[Generating UCDB and Test Plan Files](#)” section.

#### Procedure

1. (Optional) Load the Power Aware UCDB file in the Questa SIM simulator:

```
vsim -viewcov pa.ucdb
```

---

#### Note



You do not need to load the UCDB file during live-simulation.

---

2. (Optional) Use the coverage exclude command to exclude a power state or transition from the coverage report. See “[coverage exclude](#)” in the *Questa SIM Command Reference Manual*.

State exclusion:

```
coverage exclude -scope /top/pd_alu -pstate PD_SYS1 SLEEP
```

Transition exclusion:

```
coverage exclude -scope /top/pd_alu -ptrans PD_SYS2 * -> slp
```

3. Generate the Power Aware coverage report. See “[coverage report](#)” in the *Questa SIM Command Reference Manual*.

```
coverage report -pa [-details] [-verbose] [-assert]  
[-file <filename>] [-xml -file <filename>] [-html]
```

Alternatively, to generate the report during post-simulation, use the vcover report command at the command line.

For example:

```
vcover report -pa pa.ucdb
vcover report -pa pa.ucdb -verbose
vcover report -pa pa.ucdb -html
```

---

**Note**

 You can also generate the report using the **Tools > Coverage Report** menu item.

---

4. (Optional) Use the following command to list the Power Aware objects that are excluded from the coverage statistics:

```
coverage report -excluded -pa
```

## Results

The following section shows a Power Aware coverage report that is generated using the coverage report or vcover report command without any optional argument.

## Power Aware Coverage

### Power Aware Coverage Analysis

---

```
# Coverage Report Summary Data by PA state machine instance
=====
# === Instance: /alu_tester/dut/in1_reg/q_UPF_ISO
# === Design Unit: work.msp_a_iso_chk_cell_vec
#
# =====
#   Enabled Coverage      Active      Hits      Misses % Covered
#   -----      -----      -----      ----- -----
#   Assertions           5          3          2       60.00
#
# =====
# === Instance: /alu_tester/dut/in1_reg
# === Design Unit: work.register
#
# =====
#   Enabled Coverage      Active      Hits      Misses % Covered
#   -----      -----      -----      ----- -----
#   Assertions           1          0          1       0.00
#
# =====
# === Instance: /alu_tester/dut/in2_reg/q_UPF_ISO
# === Design Unit: work.msp_a_iso_chk_cell_vec
#
# =====
#   Enabled Coverage      Active      Hits      Misses % Covered
#   -----      -----      -----      ----- -----
#   Assertions           5          3          2       60.00
#
# =====
# === Instance: /alu_tester/dut/in2_reg
# === Design Unit: work.register
#
# =====
#   Enabled Coverage      Active      Hits      Misses % Covered
#   -----      -----      -----      ----- -----
#   Assertions           1          0          1       0.00

# TOTAL POWER STATE COVERAGE: 56.88% POWER STATE COVERAGE TYPES: 46

# POWER AWARE CHECKS COVERAGE SUMMARY

# TOTAL ASSERTION COVERAGE: 69.44% ASSERTIONS: 36

# Total Coverage By Instance (filtered view): 63.16%
```

## Analyzing Coverage Using the Power Aware Test Plan

Use the Power Aware test plan either in an XML file, or in the Verification Management Tracker window of the Questa SIM simulator to analyze Power Aware coverage. The test plan contains information about dynamic checks coverage, and power states and transitions coverage.

---

### Restriction

---

 ModelSim SE does not support the test plan generation.

---

## Prerequisites

- Generate a Power Aware UCDB file, for example, *pa.ucdb*. See step 1 of the “[Generating UCDB and Test Plan Files](#)” section.

- Generate the entire design UCDB file, for example, *all.ucdb*. See step of the [2 “Generating UCDB and Test Plan Files” section](#).
- Generate the Power Aware test plan UCDB file, *QuestaPowerAwaretestplan.ucdb*. See step [3](#) of the “[Generating UCDB and Test Plan Files](#)” section.

## Procedure

1. Merge the Power Aware test plan UCDB file, and the Power Aware or entire design UCDB file, to generate the merged UCDB file at the command line:

```
vcover merge merged.ucdb pa_reports/QuestaPowerAwareTestplan.ucdb  
pa.ucdb
```

```
vcover merge merged.ucdb pa_reports/QuestaPowerAwareTestplan.ucdb  
all.ucdb
```

2. Load the merged UCDB file, *merged.ucdb*, in the Questa SIM simulator:

```
vsim -viewcov merged.ucdb
```

3. Choose **View > Verification Management > Tracker** to open the Verification Management Tracker window and analyze Power Aware coverage.

---

### Tip

 You can modify the Power Aware test plan XML file, use the `xml2ucdb` command to generate the modified test plan UCDB file, and follow the steps from [1](#) to [3](#) to view the modified test plan in the Verification Management Tracker window.

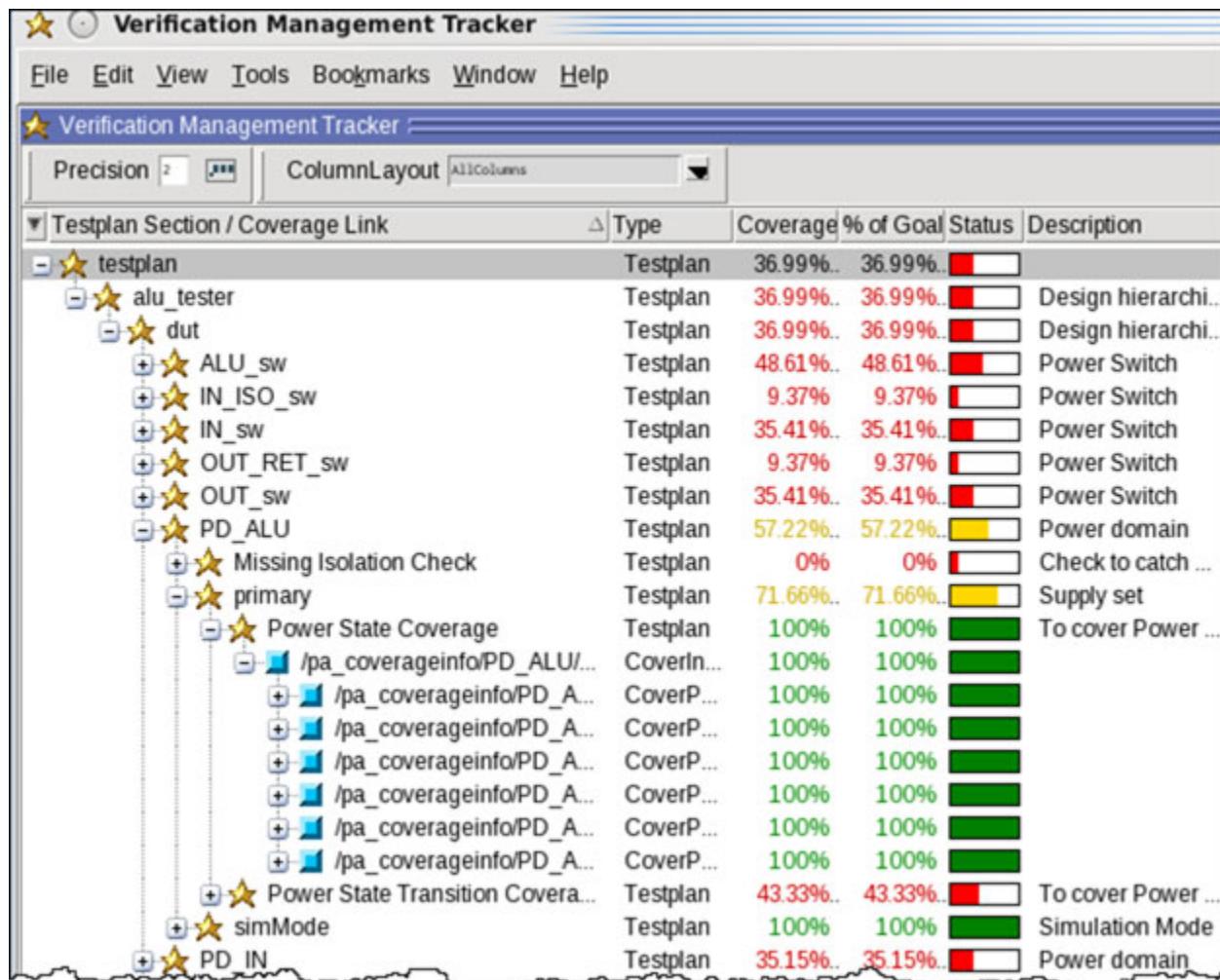
---

## Results

The power domains in the test plan are listed in either of the following ways:

- **Flat View** — By default, the Questa SIM simulator generates the flat view of the test plan, where all power domains are listed in a flat hierarchy.

**Figure 6-2. Power Aware Test Plan (Flat View) in the Verification Management Tracker Window**

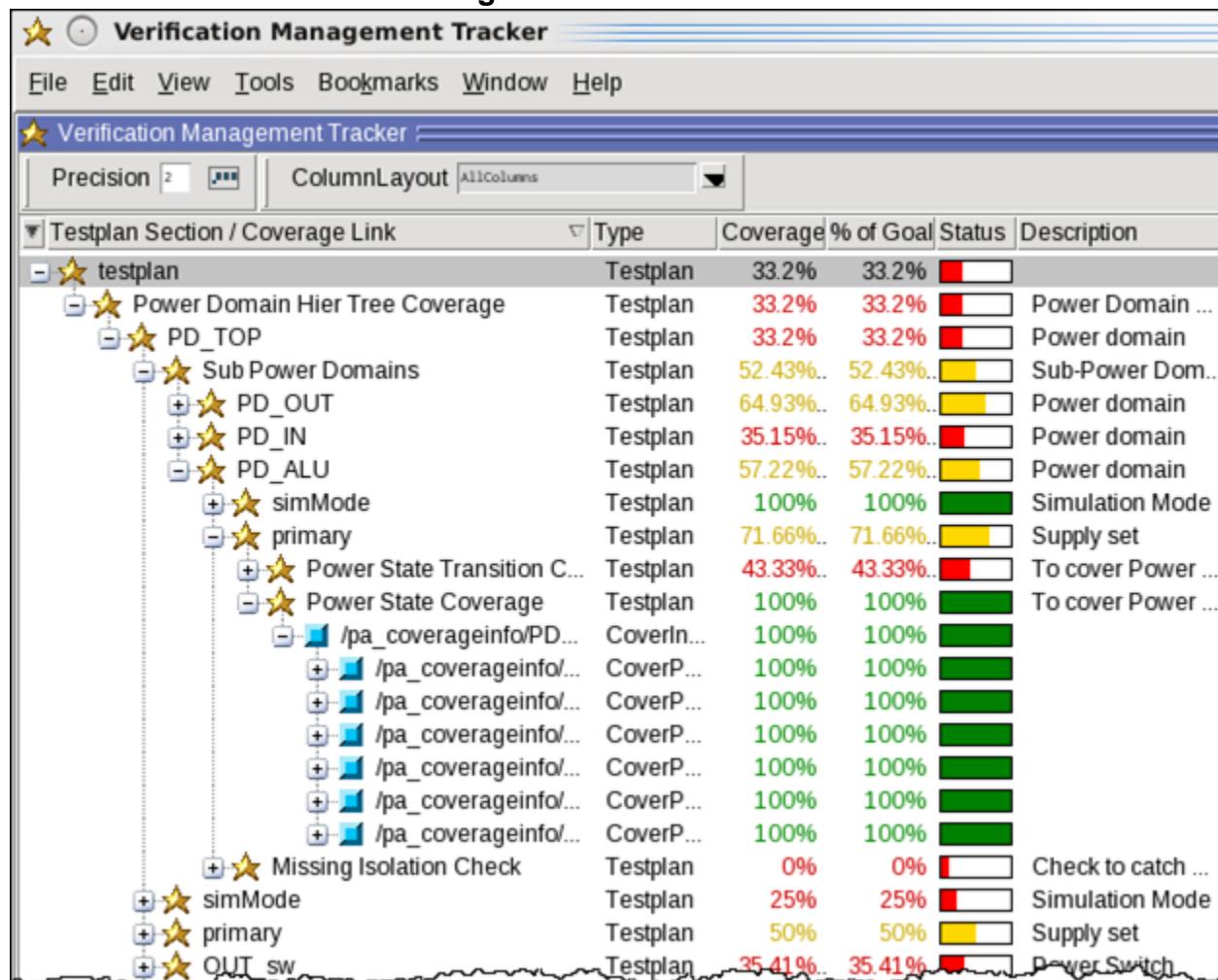


**Figure 6-3. Power Aware Test Plan (Flat View) in an XML Format**

#	Section	Description	Link	Type	Weight	Goal
1	alu tester	Design hierarchical scope			1	100
11	dut	Design hierarchical scope			1	100
111	OUT RET sw	Power Switch			1	100
1111	Power State Coverage	To cover Power States	OUT RET sw STATE COVERAGE.Valu tester/dut/QPA CTRL CRPT CHK 1	COVERG	1	100
1112	Power State Transition	To cover Power State Transitions	OUT RET sw TRANSITION COVERAGE.Valu tester/dut/QPA CTRL CRPT CHK 1	COVERG	1	100
1113	Power Switch Control	Check to catch corruption on	Valu tester/dut/QPA CTRL CRPT CHK 2	ASSERTI	1	100
1114	ctrl p	Control Port			1	100
1114.1	Power State Coverage	To cover Power States	ctrl p STATE COVERAGE.Valu tester/dut/p	COVERG	1	100
1114.2	Power State Transition	To cover Power State Transitions	ctrl p TRANSITION COVERAGE.Valu tester/dut/p	COVERG	1	100
112	OUT sw	Power Switch			1	100
1121	Power State Coverage	To cover Power States	OUT sw STATE COVERAGE.Valu tester/dut/p	COVERG	1	100
1122	Power State Transition	To cover Power State Transitions	OUT sw TRANSITION COVERAGE.Valu tester/dut/p	COVERG	1	100
1123	Power Switch Control	Check to catch corruption on	Valu tester/dut/QPA CTRL CRPT CHK 3	ASSERTI	1	100
1124	ctrl p	Control Port			1	100
1124.1	Power State Coverage	To cover Power States	ctrl p STATE COVERAGE.Valu tester/dut/p	COVERG	1	100
1124.2	Power State Transition	To cover Power State Transitions	ctrl p TRANSITION COVERAGE.Valu tester/dut/p	COVERG	1	100
113	ALU sw	Power Switch			1	100
1131	Power State Coverage	To cover Power States	ALU sw STATE COVERAGE.Valu tester/dut/p	COVERG	1	100
1132	Power State Transition	To cover Power State Transitions	ALU sw TRANSITION COVERAGE.Valu tester/dut/p	COVERG	1	100
1133	Power Switch Control	Check to catch corruption on	Valu tester/dut/QPA CTRL CRPT CHK 4	ASSERTI	1	100
1134	ctrl high	Control Port			1	100
1134.1	Power State Coverage	To cover Power States	ctrl high STATE COVERAGE.Valu tester/dut/p	COVERG	1	100
1134.2	Power State Transition	To cover Power State Transitions	ctrl high TRANSITION COVERAGE.Valu tester/dut/p	COVERG	1	100
1135	ctrl moderate	Control Port			1	100
1135.1	Power State Coverage	To cover Power States	ctrl moderate STATE COVERAGE.Valu tester/dut/p	COVERG	1	100
1135.2	Power State Transition	To cover Power State Transitions	ctrl moderate TRANSITION COVERAGE.Valu tester/dut/p	COVERG	1	100
1136	ctrl low	Control Port			1	100
1136.1	Power State Coverage	To cover Power States	ctrl low STATE COVERAGE.Valu tester/dut/p	COVERG	1	100

- Hierarchical View** — In the Hierarchical view, the test plan is based on your UPF hierarchy to aid you in navigation and for correlating results. For more information on generating the hierarchical view of the test plan, see step 2 of the “[Collecting Coverage Data of Power States and Transitions](#)” section.

**Figure 6-4. Power Aware Test Plan (Hierarchical View) in the Verification Management Tracker Window**



**Figure 6-5. Power Aware Test Plan (Hierarchical View) in an XML Format**

#	Section	Description	Link	Type	Weight	Goal
1	Power Domain Hier Tree	Power Domain Hier Tree Coverage			1	100
1.0	PD_TOP	Power domain			1	100
1.0.1	simMode	Simulation Mode			1	100
1.0.1.1	SimMode State	To cover Simulation States	simMode STATE COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.1.2	SimMode State Transition	To cover Simulation State Transitions	simMode TRANSITION COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.2	primary	Supply set			1	100
1.0.2.1	Power State Coverage	To cover Power States	primary STATE COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.2.2	Power State Transition	To cover Power State Transitions	primary TRANSITION COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.3	OUT_RET_sw	Power Switch			1	100
1.0.3.1	Power State Coverage	To cover Power States	OUT_RET_sw STATE COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.3.2	Power State Transition	To cover Power State Transitions	OUT_RET_sw TRANSITION COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.3.3	Power Switch Control	Check to catch corruption on power	Value tester/dut/QPA CTRL CRPT CHK 2	ASSERTION	1	100
1.0.3.4	ctrl_p	Control Port			1	100
1.0.3.4.1	Power State Coverage	To cover Power States	ctrl_p STATE COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.3.4.2	Power State Transition	To cover Power State Transitions	ctrl_p TRANSITION COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.4	OUT_sw	Power Switch			1	100
1.0.4.1	Power State Coverage	To cover Power States	OUT_sw STATE COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.4.2	Power State Transition	To cover Power State Transitions	OUT_sw TRANSITION COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.4.3	Power Switch Control	Check to catch corruption on power	Value tester/dut/QPA CTRL CRPT CHK 3	ASSERTION	1	100
1.0.4.4	ctrl_p	Control Port			1	100
1.0.4.4.1	Power State Coverage	To cover Power States	ctrl_p STATE COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.4.4.2	Power State Transition	To cover Power State Transitions	ctrl_p TRANSITION COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.5	ALU_sw	Power Switch			1	100
1.0.5.1	Power State Coverage	To cover Power States	ALU_sw STATE COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.5.2	Power State Transition	To cover Power State Transitions	ALU_sw TRANSITION COVERAGE Value tester/dut	COVERGROUP	1	100
1.0.5.3	Power Switch Control	Check to catch corruption on power		ASSERTION	1	100
1.0.5.4	ctrl_high	Control Port			1	100
1.0.5.4.1	Power State Coverage	To cover Power States	ctrl_high STATE COVERAGE Value tester/dut	COVERGROUP	1	100

**pa\_coverageinfo** — The simulator generates a Power Aware coverage scope, *pa\_coverageinfo*, which represents all power state and transition coverage data inside the related design instance. The coverage scope contains information related to the supply sets, power domains, ports, nets and power state tables. The name conflicts of *pa\_coverageinfo* are resolved by adding a counter to the end of the name (*pa\_coverageinfo\_<n>*) for the second and any further occurrences.

**Undefined Power State** — During simulation, when none of the named states (including predefined states) of a state object is active, the Questa SIM simulator creates a state named *Undefined*, which becomes active. This undefined state is added by the Questa SIM simulator and is representative of all the power states that are missing in the UPF, while adding states on a particular UPF objects, such as power domain, supply set, supply port, and PST.

## Related Topics

[Questa Testplan Creation Using OpenOffice](#)

[Questa Testplan Creation Using Excel Add-In](#)

## Generating the Basic Combined Coverage Report

Generate the basic combined coverage report either during live-simulation or post-simulation to analyze Power Aware coverage and non-Power Aware coverage. Power Aware coverage contains information about dynamic checks coverage, and power states and transitions coverage. Non-Power Aware coverage contains information about functional and toggle coverage.

### Prerequisites

- Generate the entire design UCDB file, for example, *all.ucdb*. See step 2 in the “[Generating UCDB and Test Plan Files](#)” section.

### Procedure

1. (Optional) Load the entire design UCDB file in the Questa SIM simulator:

```
vsim -viewcov all.ucdb
```

---

#### Note

---

 You do not need to load the UCDB file during live-simulation.

---

2. (Optional) Use the coverage exclude command to exclude a power state or transition from the coverage report. See “[coverage exclude](#)” in the *Questa SIM Command Reference Manual*.

State exclusion:

```
coverage exclude -scope /top/pd_alu -pstate PD_SYS1 SLEEP
```

Transition exclusion:

```
coverage exclude -scope /top/pd_alu -ptrans PD_SYS2 * -> slp
```

3. Generate the basic combined coverage report. See “[coverage report](#)” in the *Questa SIM Command Reference Manual*.

```
coverage report -pacombined [-details] [-verbose] [-assert]  
[-file <filename>] [-xml -file <filename>] [-html]
```

Alternatively, to generate the report during post-simulation, use the vcover report command at the command line:

```
vcover report -pacombined all.ucdb  
vcover report -pacombined all.ucdb -verbose  
vcover report -pacombined all.ucdb -html
```

---

#### Note

---

 You can also generate the report using the **Tools > Coverage Report** menu item.

---

4. (Optional) Use the following command to list the objects that are excluded from the coverage statistics:

```
coverage report -excluded
```

## Results

The following section shows a basic combined coverage report that is generated using the coverage report or vcover report command without any optional argument.

```
# Coverage Report Summary Data by instance
# =====
# === Instance: /alu_tester/dut/alu_inst/and_gate
# === Design Unit: work.and_cell
# =====
#   Enabled Coverage      Active      Hits      Misses % Covered
#   -----      -----      -----      ----- -----
#   Toggle Bins          24          10          14      41.66
# =====
# === Instance: /alu_tester/dut/alu_inst/or_gate
# === Design Unit: work.or_cell
# =====
#   Enabled Coverage      Active      Hits      Misses % Covered
#   -----      -----      -----      ----- -----
#   Toggle Bins          24          14          10      58.33
# =====
# === Instance: /alu_tester/dut/alu_inst/xor_gate
# === Design Unit: work.xor_cell
# =====
#   Enabled Coverage      Active      Hits      Misses % Covered
#   -----      -----      -----      ----- -----
#   Toggle Bins          24          12          12      50.00
# =====
# === Instance: /alu_tester/dut/alu_inst/xnor_gate
# === Design Unit: work.xnor_cell
# =====
#   Enabled Coverage      Active      Hits      Misses % Covered
#   -----      -----      -----      ----- -----
#   Toggle Bins          24          12          12      50.00
#
# TOTAL POWER STATE COVERAGE: 56.88% POWER STATE COVERAGE TYPES: 46
#
# POWER AWARE CHECKs COVERAGE SUMMARY
#
# TOTAL ASSERTION COVERAGE: 69.44% ASSERTIONS: 36
#
# Total Coverage By Instance (filtered view): 55.15%
```

## Generating the Advanced Combined Coverage Report

Generate the advanced combined coverage report during post-simulation to analyze Power Aware and non-Power Aware coverage. Power Aware coverage contains information about dynamic checks coverage, and power states and transitions coverage. Non-Power Aware

coverage contains information about functional and all structural coverage - statement, expression, FSM, branch, and toggle.

## Prerequisites

- Complete the non-Power Aware simulation. See “[Usage Flow for Code Coverage Collection](#)” in the *Questa SIM User’s Manual*.
- Generate a non-Power Aware UCDB file, for example, *nonpa.ucdb*. See “[Saving Code Coverage in the UCDB](#)” in the *Questa SIM User’s Manual*.
- Complete the Power Aware simulation. See “[Power Aware Simulation](#)”.
- Generate a Power Aware UCDB file, for example, *pa.ucdb*. See step 1 in the “[Generating UCDB and Test Plan Files](#)” section.

## Procedure

1. Merge the Power Aware UCDB file and the non-Power Aware UCDB file to generate the merged UCDB file at the command line:

```
vcover merge merged.ucdb pa.ucdb nonpa.ucdb
```

2. Generate the advanced combined coverage report. See “[vcover report](#)” in the *Questa SIM Command Reference Manual*.

```
vcover report -pacombined merged.ucdb [-details] [-verbose]  
[-assert] [-file <filename>] [-xml -file <filename>] [-html]
```

## Results

The following section shows an advanced combined coverage report that is generated using the vcover report command without any optional argument.

```

# Coverage Report Summary Data by instance
# =====
# Instance: /alu_tester/dut/alu_inst/and_gate
# === Design Unit: work.and_cell
# =====
# Enabled Coverage      Active      Hits      Misses % Covered
# -----      -----      -----      ----- -----
# Stmts          1          1          0    100.00
# Toggle Bins   24         10         14    41.66
# =====
# Instance: /alu_tester/dut/alu_inst/or_gate
# === Design Unit: work.or_cell
# =====
# Enabled Coverage      Active      Hits      Misses % Covered
# -----      -----      -----      ----- -----
# Stmts          1          1          0    100.00
# Toggle Bins   24         14         10    58.33
# =====
# Instance: /alu_tester/dut/alu_inst/xor_gate
# === Design Unit: work.xor_cell
# =====
# Enabled Coverage      Active      Hits      Misses % Covered
# -----      -----      -----      ----- -----
# Stmts          1          1          0    100.00
# Toggle Bins   24         12         12    50.00
# =====
# === Instance: /alu_tester
# === Design Unit: work.alu_tester
# =====
# Enabled Coverage      Active      Hits      Misses % Covered
# -----      -----      -----      ----- -----
# Stmts         110        110         0    100.00
# FEC Expression Terms 2          2          0    100.00
# Toggle Bins   118        31         87    26.27

# TOTAL POWER STATE COVERAGE: 56.88% POWER STATE COVERAGE TYPES: 46

# POWER AWARE CHECKs COVERAGE SUMMARY

# TOTAL ASSERTION COVERAGE: 69.44% ASSERTIONS: 36

# Total Coverage By Instance (filtered view): 73.71%

```

## Coverage Support of UPF Objects

The Questa SIM simulator supports coverage of power states and transitions of various UPF objects, such as ports and nets, PSTs, supply sets, power domains, power switches, isolation strategies, and retention strategies.

The simulator supports coverage of the following UPF objects:

- [Ports and Nets](#)
- [PSTs](#)

- Power Domains
- Supply Sets
- Power Switches
- Isolation Strategies
- Retention Strategies

Refer to the coverage reports in the “[Power Aware Coverage Analysis](#)” section for details on the coverage of various UPF objects.

The simulator also supports cross coverage of power states, which refers to the system-level coverage that is represented using the cross of various power states. See “[Cross Coverage of Power States](#)”.

## Ports and Nets

Use the following values with the vopt -pa\_coverage argument:

- portpststate — Enables coverage of user-defined states and transitions of ports and nets
- implicitportnet — Enables coverage of the implicitly created state (by the simulator) that are used in the -supply\_expr option of the add\_power\_state command
- undeftrans — Enables coverage of unnamed and undefined transitions of ports and nets when the add\_state\_transition or describe\_state\_transition command is specified in the UPF file

### vopt Command

```
vopt tb -o tbout -pa_upf test.upf
-pa_coverage=portpststate+implicitportnet+undeftrans
```

## Output in the Coverage Report

```

# -----
# UPF OBJECT                               Metric   Goal   Status
#
#
# -----
# TYPE : PORT /tb/main_port                25.00%   100   Uncovered
# PORT coverage instance \/tb/pa_coverageinfo/main_port/PS_main_port
#                                         25.00%   100   Uncovered
#     Power State undefined               0.00%    100   ZERO
#         bin ACTIVE                   0        1   ZERO
#     Power State OFF_implicit_0       100.00%   100   Covered
#         bin ACTIVE                   1        1   Covered
#     Power State FULL_ON_implicit_1   0.00%    100   ZERO
#         bin ACTIVE                   0        1   ZERO
#     Power State user_port_state_1   0.00%    100   ZERO
#         bin ACTIVE                   0        1   ZERO
#     Power State user_port_state_2   0.00%    100   ZERO
#         bin ACTIVE                   0        1   ZERO
# TYPE : PORT /tb/main_port                0.00%    100   ZERO
# PORT coverage instance \/tb/pa_coverageinfo/main_port/PS_TRANS_main_port
#                                         0.00%    100   ZERO
#     Power State Transitions          0.00%    100   ZERO
#         illegal_bin OFF_implicit_0 -> undefined 0        0   ZERO
#         illegal_bin FULL_ON_implicit_1 -> undefined 0        0   ZERO
#...
# TYPE : NET /tb/main_net                 50.00%   100   Uncovered
# NET coverage instance \/tb/pa_coverageinfo/main_net/PS_main_net
#                                         50.00%   100   Uncovered
#     Power State undefined           100.00%   100   Covered
#         bin ACTIVE                   1        1   Covered
#     Power State user_net_state_1   0.00%    100   ZERO
#         bin ACTIVE                   0        1   ZERO
#     Power State user_net_state_2   0.00%    100   ZERO
#         bin ACTIVE                   0        1   ZERO
# TYPE : NET /tb/main_net                0.00%    100   ZERO
# NET coverage instance \/tb/pa_coverageinfo/main_net/PS_TRANS_main_net
#                                         0.00%    100   ZERO
#     Power State Transitions          0.00%    100   ZERO
#         illegal_bin user_net_state_1 -> undefined 0        0   ZERO
#         illegal_bin user_net_state_2 -> undefined 0        0   ZERO
#         illegal_bin undefined -> user_net_state_1 0        0   ZERO
#         illegal_bin undefined -> user_net_state_2 0        0   ZERO
#         bin user_port_state1 -> user_port_state2 0        1   ZERO
#         bin user_port_state2 -> user_port_state1 0        1   ZERO
#
# TOTAL POWER STATE COVERAGE: 18.75% POWER STATE COVERAGE TYPES: 4
#
# Total Coverage By Instance (filtered view): 18.75%

```

## PSTs

Use the following values with the vopt -pa\_coverage argument:

- portpststate — Enables coverage of user-defined states and transitions of PSTs

- `undeftrans` — Enables coverage of unnamed and undefined transitions of PSTs when the `add_state_transition` or `describe_state_transition` command is specified in the UPF file

#### vopt Command

```
vopt tb -o tbout -pa_upf test.upf -pa_coverage=portpststate+undeftrans
```

#### Output in the Coverage Report

```
# TYPE : PST /tb/PST 33.33% 100 Uncovered
# PST coverage instance \/tb/pa_coverageinfo/PST/PS_PST
#
# 33.33% 100 Uncovered

# Power State undefined 0.00% 100 ZERO
# bin ACTIVE 0 1 ZERO
# Power State user_pst_state_1 0.00% 100 ZERO
# bin ACTIVE 0 1 ZERO
# Power State user_pst_state_2 100.00% 100 Covered
# bin ACTIVE 1 1 Covered
# TYPE : PST /tb/PST 0.00% 100 ZERO
# PST coverage instance \/tb/pa_coverageinfo/PST/PS_TRANS_PST
#
# 0.00% 100 ZERO
# Power State Transitions 0.00% 100 ZERO
# illegal_bin pst_1 -> undefined 0 ZERO
# illegal_bin pst_2 -> undefined 0 ZERO
# illegal_bin undefined -> user_pst_state_1 0 ZERO
# illegal_bin undefined -> user_pst_state_2 0 ZERO
# bin user_pst_state_1 -> user_pst_state_2 0 1 ZERO
# bin user_pst_state_2 -> user_pst_state_1 0 1 ZERO
#
# # TOTAL POWER STATE COVERAGE: 16.66% POWER STATE COVERAGE TYPES: 6
#
# # Total Coverage By Instance (filtered view): 16.66%
```

## Supply Sets

Use the following values with the `vopt -pa_coverage` argument:

- `powerstate` — Enables coverage of user-defined and predefined states and transitions of supply sets
- `undeftrans` — Enables coverage of unnamed and undefined transitions of supply sets when the `add_state_transition` or `describe_state_transition` command is specified in the UPF file

#### vopt Command

```
vopt tb -o tbout -pa_upf test.upf -pa_coverage=powerstate+undeftrans
```

## Output in the Coverage Report

```

# -----
# UPF OBJECT                               Metric      Goal     Status
#
#
# -----
#   TYPE : SUPPLY SET /tb/PD_ss           25.00%    100     Uncovered
#   SUPPLY SET coverage instance \/tb/pa_coverageinfo/PD_ss/PD_ss_PS/PS_PD_ss
#                                         25.00%    100     Uncovered
#       Power State user_state_1          0.00%    100     ZERO
#           bin ACTIVE                  0        1     ZERO
#       Power State user_state_2          0.00%    100     ZERO
#           bin ACTIVE                  0        1     ZERO
#       Power State DEFAULT_NORMAL      0.00%    100     ZERO
#           bin ACTIVE                  0        1     ZERO
#       Power State DEFAULT_CORRUPT    100.00%   100     Covered
#           bin ACTIVE                  1        1     Covered
#   TYPE : SUPPLY SET /tb/PD_ss           0.00%    100     ZERO
#   SUPPLY SET coverage instance \/tb/pa_coverageinfo/PD_ss/PD_ss_PS/PS_TRANS_PD_ss
#                                         0.00%    100     ZERO
#       Power State Transitions         0.00%    100     ZERO
#           bin user_state_2 -> user_state_1    0        1     ZERO
#           bin user_state_2 -> DEFAULT_NORMAL 0        1     ZERO
#           bin user_state_2 -> DEFAULT_CORRUPT 0        1     ZERO
#           bin user_state_1 -> user_state_2    0        1     ZERO
#           bin user_state_1 -> DEFAULT_NORMAL 0        1     ZERO
#           bin user_state_1 -> DEFAULT_CORRUPT 0        1     ZERO
#...
#
#   # TOTAL POWER STATE COVERAGE: 12.50% POWER STATE COVERAGE TYPES: 2
#
#   # Total Coverage By Instance (filtered view): 12.50%

```

## Power Domains

Use the following values with the vopt -pa\_coverage argument:

- powerstate — Enables coverage of user-defined states and transitions of power domains
- simMode — Enables coverage of states and transitions of simstates of the primary supply set
- undeftrans — Enables coverage of unnamed and undefined transitions of power domains when the add\_state\_transition or describe\_state\_transition command is specified in the UPF file

### vopt Command

```
vopt tb -o tbout -pa_upf test.upf -
pa_coverage=powerstate+simMode+undeftrans
```

## Output in the Coverage Report

```
# -----
# UPF OBJECT                               Metric   Goal    Status
#
#
# -----
#  TYPE : POWER DOMAIN /tb/PD              0.00%   100    ZERO
#  POWER DOMAIN coverage instance \/tb/pa_coverageinfo/PD/PD_PS/PS_PD
#                                         0.00%   100    ZERO
#      Power State user_state_1           0.00%   100    ZERO
#          bin ACTIVE                   0       1    ZERO
#      Power State user_state_2           0.00%   100    ZERO
#          bin ACTIVE                   0       1    ZERO
#  TYPE : POWER DOMAIN /tb/PD              0.00%   100    ZERO
#  POWER DOMAIN coverage instance \/tb/pa_coverageinfo/PD/PD_PS/PS_TRANS_PD
#                                         0.00%   100    ZERO
#      Power State Transitions          0.00%   100    ZERO
#          bin user_state_1 -> user_state_2 0       1    ZERO
#          bin user_state_2 -> user_state_1 0       1    ZERO
#  TYPE : POWER DOMAIN SimMode /tb/PD/simMode 50.00%   100    Uncovered
#  POWER DOMAIN SimMode coverage instance \/tb/pa_coverageinfo/PD/simMode/SM_PD
#                                         50.00%   100    Uncovered
#      Power State CORRUPT            100.00%  100    Covered
#          bin ACTIVE                  1       1    Covered
#      Power State NORMAL             0.00%   100    ZERO
#          bin ACTIVE                   0       1    ZERO
#  TYPE : POWER DOMAIN SimMode /tb/PD/simMode 0.00%   100    ZERO
#  POWER DOMAIN SimMode coverage instance \/tb/pa_coverageinfo/PD/simMode/SM_TRANS_PD
#                                         0.00%   100    ZERO
#      Power State Transitions          0.00%   100    ZERO
#          bin CORRUPT -> NORMAL        0       1    ZERO
#          bin NORMAL -> CORRUPT        0       1    ZERO
#
#  TOTAL POWER STATE COVERAGE: 12.50% POWER STATE COVERAGE TYPES: 4
#
#  Total Coverage By Instance (filtered view): 12.50%
```

## Power Switches

Use the `-pa_coverage=switch` value with the `vopt` command to enable coverage of the following:

- Predefined states and transitions of the switch
- Predefined states and transitions of the control port and acknowledge (ack) port of the switch:
  - Predefined states — `ACTIVE_LEVEL`, `INACTIVE`, `ACTIVE_X` and `ACTIVE_Z`
  - Predefined transitions — `HIGH_TO_LOW`, and `LOW_TO_HIGH`

### vopt Command

```
vopt tb -o tbout -pa_upf test.upf -pa_coverage=switch
```

## Output in the Coverage Report

```

# -----
# UPF OBJECT                               Metric    Goal   Status
#
#
# -----
#  TYPE : Power Switch /tb/PD_switch      25.00%    100   Uncovered
#  Power Switch coverage instance \/tb/pa_coverageinfo/PD_switch/PD_switch_PS/PS_PD_switch
#                                              25.00%    100   Uncovered
#      Power State ON                     0.00%    100   ZERO
#          bin ACTIVE                      0        1   ZERO
#      Power State OFF                    100.00%   100   Covered
#          bin ACTIVE                      1        1   Covered
#      Power State PARTIAL_ON            0.00%    100   ZERO
#          bin ACTIVE                      0        1   ZERO
#      Power State ERROR                  0.00%    100   ZERO
#          bin ACTIVE                      0        1   ZERO
#  TYPE : Power Switch /tb/PD_switch      0.00%    100   ZERO
#  Power Switch coverage instance \/tb/pa_coverageinfo/PD_switch/PD_switch_PS/
PS_TRANS_PD_switch
#                                              0.00%    100   ZERO
#      Power State Transitions           0.00%    100   ZERO
#          bin ON -> OFF                 0        1   ZERO
#          bin ON -> PARTIAL_ON         0        1   ZERO
#          bin ON -> ERROR              0        1   ZERO
#...
#  TYPE : Power Switch Control Port /tb/PD_switch/ctrl  100.00%    100   Covered
#  Power Switch Control Port coverage instance \/tb/pa_coverageinfo/PD_switch/ctrl/PS_ctrl
#                                              100.00%   100   Covered
#      Power State ACTIVE_LEVEL        100.00%   100   Covered
#          bin ACTIVE                      1        1   Covered
#      Power State INACTIVE             100.00%   100   Covered
#          bin ACTIVE                      2        1   Covered
#  TYPE : Power Switch Control Port /tb/PD_switch/ctrl  50.00%    100   Uncovered
#  Power Switch Control Port coverage instance \/tb/pa_coverageinfo/PD_switch/ctrl/
PS_TRANS_ctrl
#                                              50.00%    100   Uncovered
#      Power State Transitions           50.00%    100   Uncovered
#          bin HIGH_TO_LOW                0        1   ZERO
#          bin LOW_TO_HIGH               1        1   Covered
#
#  TYPE : Power Switch Ack Port /tb/PD_switch/ack       0.00%    100   ZERO
#  Power Switch Ack Port coverage instance \/tb/pa_coverageinfo/PD_switch/ack/PS_ack
#                                              0.00%    100   ZERO
#      Power State ACTIVE_LEVEL        0.00%    100   ZERO
#          bin ACTIVE                      0        1   ZERO
#      Power State INACTIVE             0.00%    100   ZERO
#          bin ACTIVE                      0        1   ZERO
#  TYPE : Power Switch Ack Port /tb/PD_switch/ack       0.00%    100   ZERO
#  Power Switch Ack Port coverage instance \/tb/pa_coverageinfo/PD_switch/ack/PS_TRANS_ack
#                                              0.00%    100   ZERO
#      Power State Transitions           0.00%    100   ZERO
#          bin HIGH_TO_LOW                0        1   ZERO
#          bin LOW_TO_HIGH               0        1   ZERO
#
#  TOTAL POWER STATE COVERAGE: 29.16% POWER STATE COVERAGE TYPES: 6
#
# Total Coverage By Instance (filtered view): 29.16%

```

## Isolation Strategies

Use the -pa\_coverage=iso value with the vopt command to enable coverage of the following:

- Predefined states and transitions of the simstates of the isolation supply set
- Predefined states and transitions of the isolation enable signal:
  - Predefined states — ACTIVE\_LEVEL, INACTIVE, ACTIVE\_X and ACTIVE\_Z
  - Predefined transitions — HIGH\_TO\_LOW, and LOW\_TO\_HIGH

### vopt Command

```
vopt tb -o tbout -pa_upf test.upf -pa_coverage=iso
```

## Output in the Coverage Report

```

# -----
# UPF OBJECT                               Metric   Goal   Status
#
#
# -----
#  TYPE : ISOLATION CELL SimMode /test/tb/pd_subtop1/iso_btwn_subtops/simMode
#                                         50.00%    100   Uncovered
#  ISOLATION CELL SimMode coverage instance \test/tb/pa_coverageinfo/pd_subtop1/
# iso_btwn_subtops/simMode/SM_iso_btwn_subtops
#                                         50.00%    100   Uncovered
#      Power State CORRUPT                0.00%    100   ZERO
#          bin ACTIVE                     0        1   ZERO
#      Power State NORMAL                 100.00%   100   Covered
#          bin ACTIVE                     1        1   Covered
#  TYPE : ISOLATION CELL SimMode /test/tb/pd_subtop1/iso_btwn_subtops/simMode
#                                         0.00%    100   ZERO
#  ISOLATION CELL SimMode coverage instance \test/tb/pa_coverageinfo/pd_subtop1/
# iso_btwn_subtops/simMode/SM_TRANS_iso_btwn_subtops
#                                         0.00%    100   ZERO
#      Power State Transitions           0.00%    100   ZERO
#          bin CORRUPT -> NORMAL       0        1   ZERO
#          bin NORMAL -> CORRUPT       0        1   ZERO
#  TYPE : Isolation Enable Signal /test/tb/pd_subtop1/iso_btwn_subtops/ISO_SIG
#                                         100.00%   100   Covered
#  Isolation Enable Signal coverage instance \test/tb/pa_coverageinfo/pd_subtop1/
# iso_btwn_subtops/ISO_SIG/PS_ISO_SIG
#                                         100.00%   100   Covered
#      Power State ACTIVE_LEVEL         100.00%   100   Covered
#          bin ACTIVE                   14       1   Covered
#      Power State INACTIVE            100.00%   100   Covered
#          bin ACTIVE                   12       1   Covered
#  TYPE : Isolation Enable Signal /test/tb/pd_subtop1/iso_btwn_subtops/ISO_SIG
#                                         100.00%   100   Covered
#  Isolation Enable Signal coverage instance \test/tb/pa_coverageinfo/pd_subtop1/
# iso_btwn_subtops/ISO_SIG/PS_TRANS_ISO_SIG
#                                         100.00%   100   Covered
#      Power State Transitions         100.00%   100   Covered
#          bin HIGH_TO_LOW              12       1   Covered
#          bin LOW_TO_HIGH              12       1   Covered
#
#  TOTAL POWER STATE COVERAGE: 62.50% POWER STATE COVERAGE TYPES: 4
#
#  Total Coverage By Instance (filtered view): 62.50%

```

## Retention Strategies

Use the `-pa_coverage=ret` value with the `vopt` command to enable coverage of the following:

- Predefined states and transitions of the simstates of the retention supply set
- Predefined states and transitions of the save and restore events:
  - Predefined states — `ACTIVE_LEVEL`, and `INACTIVE`
  - Predefined transitions — `HIGH_TO_LOW`, and `LOW_TO_HIGH`

### vopt Command

```
vopt tb -o tbout -pa_upf test.upf -pa_coverage=ret
```

## Output in the Coverage Report

```

# -----
# UPF OBJECT                               Metric    Goal   Status
#
#
# -----
#  TYPE : RETENTION CELL SimMode /tb/pd_dff/ret_dff/simMode
#          50.00%      100   Uncovered
#  RETENTION CELL SimMode coverage instance \tb\pa_coverageinfo\pd_dff\ret_dff\simMode\
SM_ret_dff
#          50.00%      100   Uncovered
#      Power State CORRUPT            0.00%      100   ZERO
#          bin ACTIVE                0         1   ZERO
#      Power State NORMAL           100.00%     100   Covered
#          bin ACTIVE                1         1   Covered
#  TYPE : RETENTION CELL SimMode /tb/pd_dff/ret_dff/simMode
#          0.00%      100   ZERO
#  RETENTION CELL SimMode coverage instance \tb\pa_coverageinfo\pd_dff\ret_dff\simMode\
SM_TRANS_ret_dff
#          0.00%      100   ZERO
#      Power State Transitions       0.00%      100   ZERO
#          bin CORRUPT -> NORMAL      0         1   ZERO
#          bin NORMAL -> CORRUPT      0         1   ZERO
#  TYPE : Retention Save Event /tb/pd_dff/ret_dff/save_event
#          100.00%     100   Covered
#  Retention Save Event coverage instance \tb\pa_coverageinfo\pd_dff\ret_dff\save_event\
PS_save_event
#          100.00%     100   Covered
#      Power State ACTIVE_LEVEL      100.00%     100   Covered
#          bin ACTIVE                1         1   Covered
#      Power State INACTIVE         100.00%     100   Covered
#          bin ACTIVE                2         1   Covered
#  TYPE : Retention Save Event /tb/pd_dff/ret_dff/save_event
#          100.00%     100   Covered
#  Retention Save Event coverage instance \tb\pa_coverageinfo\pd_dff\ret_dff\save_event\
PS_TRANS_save_event
#          100.00%     100   Covered
#      Power State Transitions       100.00%     100   Covered
#          bin HIGH_TO_LOW           1         1   Covered
#          bin LOW_TO_HIGH           1         1   Covered
#  TYPE : Retention Restore Event /tb/pd_dff/ret_dff/restore_event
#          100.00%     100   Covered
#  Retention Restore Event coverage instance \tb\pa_coverageinfo\pd_dff\ret_dff\
restore_event/PS_restore_event
#          100.00%     100   Covered
#      Power State ACTIVE_LEVEL      100.00%     100   Covered
#          bin ACTIVE                1         1   Covered
#      Power State INACTIVE         100.00%     100   Covered
#          bin ACTIVE                2         1   Covered
#  TYPE : Retention Restore Event /tb/pd_dff/ret_dff/restore_event
#          100.00%     100   Covered
#  Retention Restore Event coverage instance \tb\pa_coverageinfo\pd_dff\ret_dff\
restore_event/PS_TRANS_restore_event
#          100.00%     100   Covered
#  Power State Transitions         100.00%     100   Covered
#          bin HIGH_TO_LOW           1         1   Covered
#          bin LOW_TO_HIGH           1         1   Covered
#

```

```
# TOTAL POWER STATE COVERAGE: 75.00% POWER STATE COVERAGE TYPES: 6
#
# Total Coverage By Instance (filtered view): 75.00%
```

## Cross Coverage of Power States

The simulator supports system-level coverage as part of the Power Aware simulation. The system-level coverage captures the occurrences of various interdependent power states, and is referred as cross coverage of power states. These power states correspond to various power domains of the design.

Cross coverage is implemented according to the following:

- The simulator considers only power states of power domains for cross coverage.
- The simulator enables cross coverage when you enable coverage of power states using the vopt -pa\_coverage (or vopt -pa\_coverage=powerstate) and vsim -coverage commands. See “Default Cross Coverage Behavior”.
- To enable complete cross coverage, add the -pa\_coverage=crossdontcare argument to your vopt command. See “Complete Cross Coverage Behavior”.
- To control the depth of the dependency tree of the power states, specify the -pa\_crosscoveredepth <integer> argument with your vopt command.
- To disable cross coverage, specify the -pa\_coverageoff=crosscov argument with your vopt command.

### Example

Consider the following UPF commands, where power states of a top-level power domain PD are dependent on the power states of PD\_sub1 and PD\_sub2. PD is separated by a single level from PD\_sub1 and PD\_sub2.

```
add_power_state PD_sub1 -state {PD_sub1_ps1 -supply_expr { (main == {FULL_ON}) }}add_power_state PD_sub1 -state {PD_sub1_ps2 -supply_expr { (main == {OFF}) }}add_power_state PD_sub2 -state {PD_sub2_ps1 -supply_expr { (main == {FULL_ON}) }}add_power_state PD_sub2 -state {PD_sub2_ps2 -supply_expr { (main == {OFF}) }}add_power_state PD -state {PD_ps1 -logic_expr {PD_sub1 == PD_sub1_ps1 && PD_sub2 == PD_sub2_ps2}}add_power_state PD -state {PD_ps2 -logic_expr {PD_sub2 == PD_sub2_ps1 }}
```

### Default Cross Coverage Behavior

For default cross coverage calculations, the simulator considers only those power state combinations that you explicitly specify with an add\_power\_state command instead of all possible power state combinations.

For example, the simulator uses the following power state combinations for default cross coverage calculations:

```
PD_ps1, PD_sub1_ps1, PD_sub2_ps2
PD_ps2, PD_sub2_ps1
```

### vopt Command

```
vopt tb -o tb_out -pa_upf test.upf -pa_coverage=powerstate
```

### Output in the Coverage Report

```
# TYPE : POWER STATE CROSS /tb/PD(ID:PD1), /tb/PD_sub2(ID:PD2), /tb/
PD_sub1(ID:PD3)#
100 Uncovered# POWER STATE CROSS coverage instance \tb\pa_coverageinfo\PD/
PD_PS_CROSS/PS_CROSS_PD
# 50.00% 100 Uncovered
# Power State Cross 50.00% 100 Uncovered
# bin \PD1:PD_ps2-PD2:PD_sub2_ps1 1 1 Covered
# bin \PD1:PD_ps1-PD2:PD_sub2_ps2-PD3:PD_sub1_ps1 0 1 ZERO
#
```

### Complete Cross Coverage Behavior

For complete cross coverage, add the `-pa_coverage=crossdontcare` argument to your existing vopt command. This enables coverage of the don't care states. Don't care states are the states whose dependency you do not explicitly specify in the `add_power_state` command.

For example, the simulator uses the following power state combinations for complete cross coverage calculations:

```
PD_ps1, PD_sub1_ps1, PD_sub2_ps2
PD_ps2, PD_sub2_ps1, PD_sub1_ps1 //PD_sub1_ps1 is a don't care state
PD_ps2, PD_sub2_ps1, PD_sub1_ps2 //PD_sub1_ps2 is a don't care state
```

### vopt Command

```
vopt tb -o tb_out -pa_upf test.upf -pa_coverage=powerstate+crossdontcare
```

### Output in the Coverage Report

```
# TYPE : POWER STATE CROSS /tb/PD(ID:PD1), /tb/PD_sub2(ID:PD2), /tb/
PD_sub1(ID:PD3)#
100 Uncovered# POWER STATE CROSS coverage instance \tb\pa_coverageinfo\PD/
PD_PS_CROSS/PS_CROSS_PD#
100 Uncovered# Power State Cross 33.33%
100 Uncovered# bin\PD1:PD_ps2-PD2:PD_sub2_ps1-PD3:PD_sub1_ps2#
0 1 ZERO 33.33%
# bin \PD1:PD_ps2-PD2:PD_sub2_ps1-PD3:PD_sub1_ps1#
1 1 Covered 33.33%
# bin \PD1:PD_ps1-PD2:PD_sub2_ps2-PD3:PD_sub1_ps1 0 1 ZERO
#
```

## Related Topics

[Power Aware Coverage Overview](#)



# Chapter 7

## Graphical Display of Power Aware Operations and Results

---

You can use windows in the Questa SIM graphical user interface (GUI) to obtain a more visual display of your Power Aware design and simulation.

---

### Tip

 For more complete information on using the graphical user interface, refer to the *Questa SIM GUI Reference Manual*.

---

<b>UPF Object Display</b> . . . . .	<b>238</b>
UPF Object Concepts . . . . .	238
Displaying UPF Objects in the GUI . . . . .	238
<b>Power Aware Source Window</b> . . . . .	<b>241</b>
Finding Power Aware Drivers . . . . .	241
UPF Color Scheme in the Source Window . . . . .	243
UPF-specific Context Menu in the Source Window . . . . .	244
Show Drivers Control Bar . . . . .	245
<b>Power Aware Schematic Display</b> . . . . .	<b>246</b>
Top-Down Debugging (From the Test Bench) . . . . .	246
Bottom-Up Debugging (From the Design Under Test) . . . . .	247
Schematic Window Features for Debugging . . . . .	248
<b>Power Aware Waveform Display</b> . . . . .	<b>254</b>
Power Aware Waveform Concepts . . . . .	254
Using Power Aware Highlighting . . . . .	255
<b>Power State and Transition Display</b> . . . . .	<b>256</b>
Power State and Transition Concepts . . . . .	256
Displaying Power Aware State Machines . . . . .	257
Power Aware State Machine List Window . . . . .	258
Power Aware State Machine Viewer Window . . . . .	260
<b>Power Aware Static Check Display</b> . . . . .	<b>265</b>
Viewing Static Checks in the Results Analysis Window . . . . .	265
GUI Elements of the Results Analysis Window for Static Checks . . . . .	266

## UPF Object Display

The Objects, Structure, and Wave windows provide visibility into UPF objects.

<b>UPF Object Concepts .....</b>	<b>238</b>
<b>Displaying UPF Objects in the GUI.....</b>	<b>238</b>

## UPF Object Concepts

The Objects, Structure, and Wave windows includes special power aware-specific information.

- A special icon  identifies UPF objects. It is available in the Structure, Object, and Wave windows.
- Power Domain information in the **PD Name** column of the Structure window (not shown by default).
- Labeling of Isolation and Level-Shifter cells in the in the **Design Unit** column of the Structure window.
- Additional information about UPF objects in the **Kind** and **PAInfo** column of the Objects window.
- Color coding based on different power domains. It is available in the Structure, Object, and Wave windows.
- Special UPF context menu, available with a right-click, for viewing UPF information in the Source and Wave windows.
- Ability to filter out UPF objects from the Structure window: Right-click > Show, uncheck UPF Objects.

## Displaying UPF Objects in the GUI

This task describes how to display UPF objects in the Structure, Objects, and Wave windows of the graphical user interface.

### Prerequisites

- Be able to run an existing Power Aware simulation flow.

### Procedure

1. Open the Questa SIM GUI.
2. Add the **-pa\_enable=debug** argument to your **vopt** command.
3. Run your Power Aware simulation flow as you normally do.

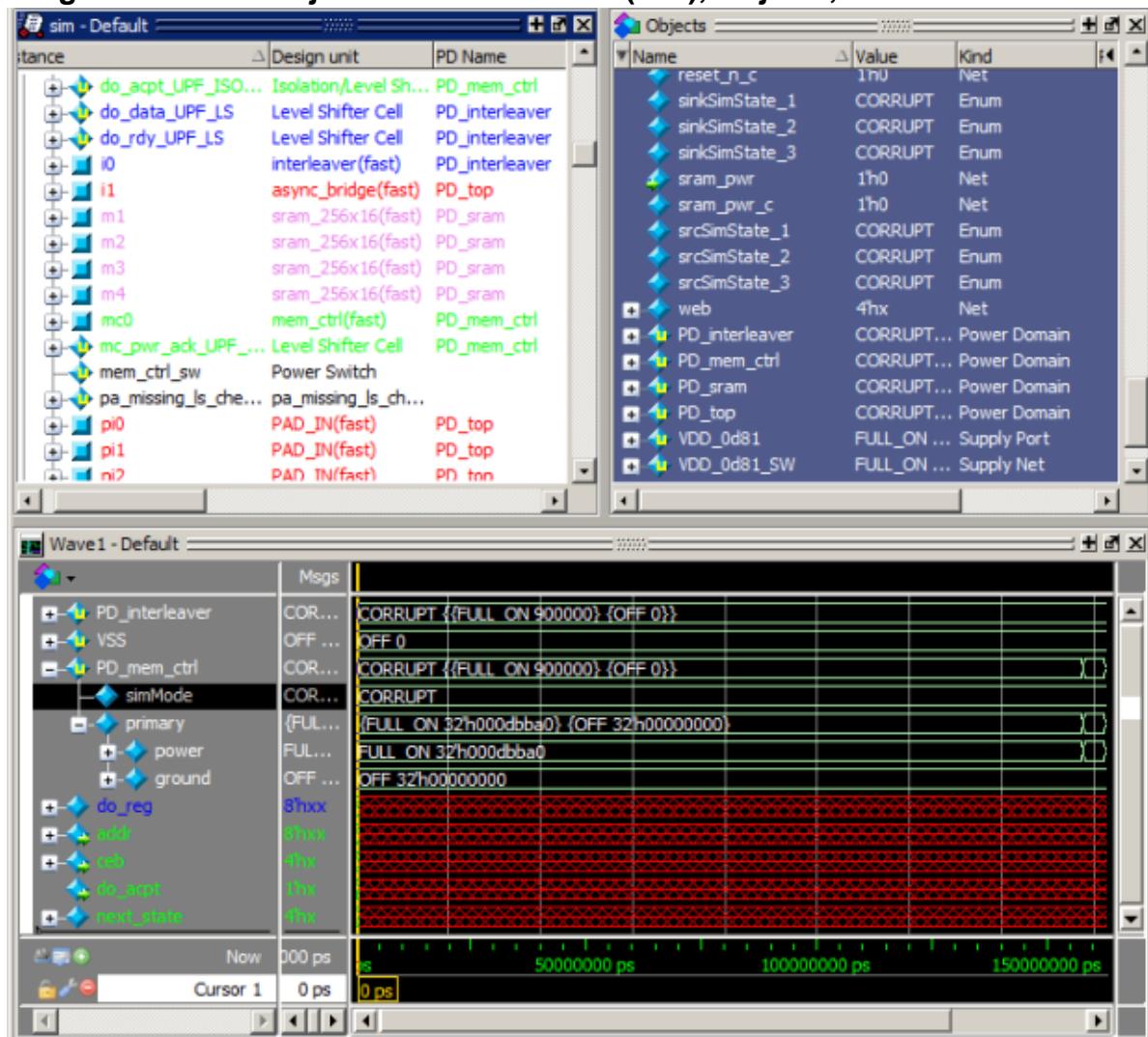
4. Ensure the Objects, Structure, and Wave windows are open:
  - view structure
  - view objects
  - view wave
5. Locate and select a UPF object in the Structure Window (refer to [Figure 7-1](#)).
  - a. Click the UPF object to view additional port and net information in the Objects window.
  - b. Right-click the UPF object and choose Add Wave to add all of the related ports and nets in the Wave window.

The default value of supply ports and nets in these windows are:

- Voltage (Decimal)
- State (Enum, UNDETERMINED, FULL\_ON, OFF, PARTIAL\_ON).

## Results

**Figure 7-1. UPF Objects in the Structure (sim), Objects, and Wave Windows**



# Power Aware Source Window

The Source window allows you to analyze information about your design that is specific to Power Aware.

All of the features in this section are enabled when you add the **-pa\_enable=debug** argument to your **vopt** command line.

<b>Finding Power Aware Drivers .....</b>	<b>241</b>
<b>UPF Color Scheme in the Source Window .....</b>	<b>243</b>
<b>UPF-specific Context Menu in the Source Window .....</b>	<b>244</b>
<b>Show Drivers Control Bar .....</b>	<b>245</b>

## Finding Power Aware Drivers

Use the source window to view drivers in a power aware design. By default, this also includes the display of isolation cells, level shifters, and buffers.

You can disable the functionality of showing isolation cells, level shifters, and buffers by disabling the menu item **Source > UPF > Show PA Cells in Path**.

### Prerequisites

- Specify **-pa\_enable=debug** on your **vopt** command line.
- Run your simulation.

### Procedure

1. In the Structure window, double-click on an instance you want to evaluate.

This opens the source file in which the instance is declared and highlights the specific instance.

2. In the Source Window, double-click on the signal you want to investigate.

This evaluates the design and finds any drivers for the signal selected, and changes the focus of the following windows:

- Source window — selects and highlights the driver (or first in the list of drivers, if more than one is found) of the selected signal.
- Structure window — highlights the associated level of hierarchy of the selected signal.
- Objects window — highlights the selected signal.
- Wave window — highlights the selected signal, if it is in the Wave window.

3. (optional) Use the [Show Drivers Control Bar](#) to select from multiple drivers or to control the preferences of the interface.

## Results

The header bar of the source window contains the [Finding Power Aware Drivers](#) that allows you to find all drivers, RTL and UPF, of the selected signal.

## Related Topics

[Show Drivers Control Bar](#)

## UPF Color Scheme in the Source Window

To access: Source > UPF > Show UPF Coloring

Use this menu choice to locate names of UPF signals in your Source window.

```
Ln#  1/3 Driver Lines
23      #10 iso_sigA = 0;
24      iso_sigB = 0;
25      end
26
27      endmodule
28
29      module A(input in, input clk, output out);
30      assign out = in & clk;
31      endmodule
32
33      module B(input in, input clk, output out);
34      assign out = in & clk;
35      endmodule
36
```

### Objects

- Text Markup Styles.
  - Red underline — (not shown) identifies signals with a related corruption cell.
  - Blue-green outline — identifies signals with a related isolation cell.
  - Purple outline — (not shown) identifies signals with a related level-shifter cell.

### Related Topics

[Finding Power Aware Drivers](#)

## UPF-specific Context Menu in the Source Window

To access: Right-click in the Source window

Use this context menu to perform Power Aware actions for a selected signal.

### Objects

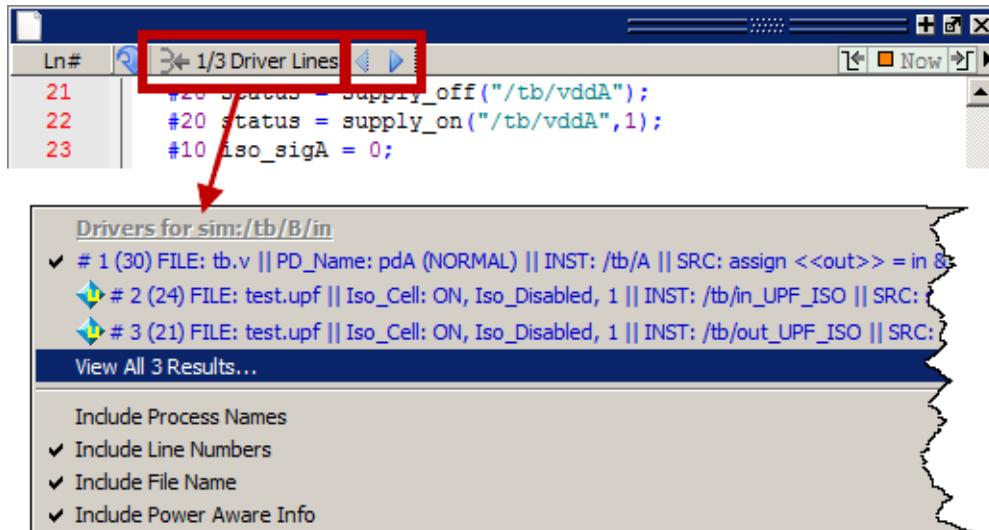
**Table 7-1. Source Window: UPF Context Menu**

Menu Item	Description
UPF > View <ul style="list-style-type: none"><li>• Power Domain</li><li>• Isolation Strategy</li><li>• Level Shifter Strategy</li><li>• UPF Supply Connection</li></ul>	Opens the corresponding UPF file, highlighting the line that defines the behavior. Prints information about the UPF behavior to the Transcript window.
UPF > Add To Wave <ul style="list-style-type: none"><li>• Power Domain</li><li>• Isolation Cell</li><li>• Level Shifter Cell</li><li>• UPF Supply Connection</li></ul>	Adds Power Aware information to the Wave window.

## Show Drivers Control Bar

To access: Available by default.

Use this element to access navigation tools and preference options for signal drivers.



## Objects

- Control bar elements.
  - Driver/Driver Lines — Button to display driver list and preference settings:
    - Driver list — Click to highlight driver in Source, Object, Wave and Structure windows. The values are signals or UPF objects (as denoted by the UPF icon). For HDL objects, the signal information contains power domain information, in the form “PD\_Name: name (state)”. For UPF objects (level shifters or isolation cells) the information contains power aware information in the form “type: current\_state, drive\_read, clamp\_value”.
    - View All Results — Opens the Multiple Drivers dialog box, containing the same information in columnned format.
    - Include <topic> — Alters presentation of driver information to include additional information.
  - Navigation Buttons — Forward and back buttons to change between multiple drivers, when applicable.

## Related Topics

[Finding Power Aware Drivers](#)

## Power Aware Schematic Display

---

You can perform debugging at the same time you run a Power Aware simulation by adding the `-debugdb` argument to both the `vopt` and `vsim` commands. The results of both the Power Aware analysis and the debug operation are provided as Power Aware schematic in the Schematic window. You can also view a correlation between the UPF power intent and the design display in the Schematic window.

Refer to “[Schematic Window](#)” in the *User’s Manual* for more information.

---

### Note

---

 Debugging in Power Aware is supported for RTL usage flow only—it is not available for gate-level simulation (GLS).

---

<b>Top-Down Debugging (From the Test Bench) .....</b>	<b>246</b>
<b>Bottom-Up Debugging (From the Design Under Test).....</b>	<b>247</b>
<b>Schematic Window Features for Debugging.....</b>	<b>248</b>

## Top-Down Debugging (From the Test Bench)

You can run Power Aware analysis and debugging from the top of the design (test bench) with or without specifying an optimized design unit. These methods resemble the conventional two-step and three-step optimization flows.

- No optimized design unit — This flow resembles the conventional two-step flow, where you use the `vopt` command to specify a Power Aware simulation and debugging only; no optimization is performed. When you run the `vsim` command, it performs debugging and runs `vopt` internally to perform optimization (see [Using the Delayed Optimization Flow](#)). For example:

```
vlog design.v
vcom design.vhdl
vopt -pa_upf <config_file> -debugdb tb
vsim tb -pa -debugdb [-vopt]
```

- Optimized design unit — This flow resembles conventional three-step flow, where you use the `vopt` command to specify a Power Aware simulation, the name of the design unit to be optimized, and debugging,. When you run the `vsim` command, it begins simulation on the optimized design unit and runs debugging ([General Steps for Running Power Aware](#)). For example:

```
vlog design.v
vcom design.vhdl
vopt -o optdu -pa_upf <config_file> -debugdb tb
vsim optdu -pa -debugdb
```

## Bottom-Up Debugging (From the Design Under Test)

You can run Power Aware analysis from the DUT hierarchy and debugging on the complete design. Running **vopt -pa\_top** captures the DUT hierarchy for Power Aware analysis

- No optimized design unit — This flow resembles conventional two-step flow, where you use the **vopt** command to specify a Power Aware simulation and debugging without optimization. Use the **-pa\_top** argument to capture the DUT hierarchy for Power Aware analysis. When you run the **vsim** command, it performs debugging and runs **vopt** internally to perform optimization (see [Using the Delayed Optimization Flow](#)). For example:

```
vlog design.v
vcom design.vhd1
vopt -pa_upf <config_file> tb -debugdb -pa_top /tb/dut
vsim tb -pa -debugdb [-vopt]
```

- Optimized design unit — This flow resembles the conventional three-step flow, where you use the **vopt** command to specify a Power Aware simulation, the name of the design unit to be optimized, and debugging. Use the **-pa\_top** argument to capture the DUT hierarchy for Power Aware analysis. For example:

```
vlog design.v
vcom design.vhd1
vopt -o optdu -pa_upf <config_file> tb -debugdb -pa_top /tb/dut
vsim optdu -pa -debugdb
```

This DUT-based flow with an optimized design unit not only does common analysis for Power Aware and debugging, but also provides flexibility to enable Power Aware analysis from specific hierarchy and do code generation in one step.

### Usage Notes

- The **-pa\_top** argument is used to specify hierarchy of UPF root scope. This supports Power Aware analysis of UPF from hierarchy other than the vopt TOP hierarchy. If **vopt** is run from test bench (tb) and UPF scope is starting from DUT (which is instantiated in test bench as **dut\_inst**), then you need to specify **vopt -pa\_top /tb/<dut\_inst>**.
- If **-pa\_top** is specifying the hierarchy other than UPF root scope then an Error message is displayed:

```
** Error: ./src/ss_error_1/test.upf(2): UPF: (vopt-9782) PA Top '/tb/top/dut_inst/top_inst' is specifying incorrect hierarchy for UPF scope 'DUT'.
```

- Do not use the vopt **-pa\_prefix** and **-pa\_replacetop** arguments with **-pa\_top**. If you do, the **-pa\_prefix** and **-pa\_replacetop** arguments are ignored and a Warning message is displayed:

```
** Warning: UPF: (vopt-9780) Option "-pa_prefix/-pa_replacetop" is
not applicable with option "-pa_top". Ignoring option "-pa_prefix/-
pa_replacetop".
```

## Schematic Window Features for Debugging

The Schematic window of the graphical user interface (GUI) provides various features that help you view debugging results from a Power Aware simulation.

In particular, these features appear in the Schematic window as follows:

- Power Domain — All design elements are colored and highlighted according to their respective power domains (see [Figure 7-2](#)).
  - All design elements, such as mux, flip-flops, and gates are colored according to the power domain specification.
  - The granularity of power domain display is at the instance level.
  - Any simulation-only power domains that are specified on a process or signal are not highlighted.
  - Colorization supports up to 16 different power domains.
  - If the power domain is in the OFF state, the schematic is filled with a gray color (see [Figure 7-3](#)).
  - Isolation cells are filled in with a green color.
  - Level-shifter cells are filled in with a purple color.
- Excluded Domains — shown as default schematic color. You can define a power domain to be excluded.
  - Power Aware exclude file support (vopt **-pa\_excludefile**)
  - Currently, PG-type connected instances are excluded. There is no analysis information to decide whether they inherit the parent power domain, create their own, or have multiple power domains (such as memories)

Both are supported with instance-level granularity; signal- and process-level exclusion are not displayed.

- UPF Source Viewing — You can view the source text of the UPF file for power domain specifications. This information for a design element is available by doing either of the following:
  - Right-click and select **View Selection > Power Domain** — Displays UPF source code (see [Figure 7-4](#)).
  - Hover the mouse cursor — Displays a Tool Tip that concatenates the HDL source file with the appropriate line number in the UPF source file (see [Figure 7-5](#)).

### Example 7-1. UPF File to Demonstrate Schematic Display for Debugging

```
set_design_top top

create_power_domain P1 -elements {dut1}
#-----
create_supply_port VDD      -domain P1
create_supply_net VDD_NET  -domain P1
connect_supply_net VDD_NET -ports { VDD }
#-----
create_supply_port GND      -domain P1
create_supply_net GND_NET  -domain P1
connect_supply_net GND_NET -ports { GND }
#-----
create_supply_net VDD_PRI  -domain P1
set_domain_supply_net P1 -primary_power_net VDD_PRI -
primary_ground_net GND_NET
#-----
----- #

create_power_switch P1_SW \
-domain P1 \
-output_supply_port {VDD_SW VDD_PRI} \
-input_supply_port {VDD_SW_In1 VDD_NET} \
-control_port {ctrl1 dut_sleep} \
-on_state {full_s1 VDD_SW_In1 {!ctrl1}} \
-off_state {off_s0 {ctrl1}}


create_power_domain P2 -elements {dut2}
#-----
create_supply_port VDD_P2      -domain P2
create_supply_net VDD_N -domain P2
connect_supply_net VDD_N -ports { VDD_P2 }
#-----
create_supply_port GND_P2      -domain P2
create_supply_net GND_N -domain P2
connect_supply_net GND_N -ports { GND_P2 }
#-----
create_supply_net VDD_P -domain P2
set_domain_supply_net P2 -primary_power_net VDD_P -
primary_ground_net GND_NET


create_power_switch P1_SW2 \
-domain P2 \
-output_supply_port {VDD_SW VDD_PRI} \
-input_supply_port {VDD_SW_In1 VDD_NET} \
-control_port {ctrl1 dut_sleep} \
-on_state {full_s1 VDD_SW_In1 {!ctrl1}} \
-off_state {off_s0 {ctrl1}}


create_power_domain P3 -elements {dut3}
#-----
```

```

create_supply_port VDD_P3      -domain P3
create_supply_net  VDD_NE -domain P3
connect_supply_net VDD_NE -ports { VDD_P3 }
#-----
create_supply_port GND_P3      -domain P3
create_supply_net  GND_NE -domain P3
connect_supply_net GND_NE -ports { GND_P3 }
#-----
create_supply_net  VDD_PR -domain P3
set_domain_supply_net P3 -primary_power_net VDD_PR -
primary_ground_net GND_NE
create_power_switch P1_SW3 \
-domain P3 \
-output_supply_port {VDD_SW VDD_PRI} \
-input_supply_port {VDD_SW_In1 VDD_NET} \
-control_port {ctrl1 dut_sleep } \
-on_state {full_s1 VDD_SW_In1 {!ctrl1}} \
-off_state {off_s0 {ctrl1}}

```

### Example 7-2. Questa SIM Commands to Run Power Aware Debugging

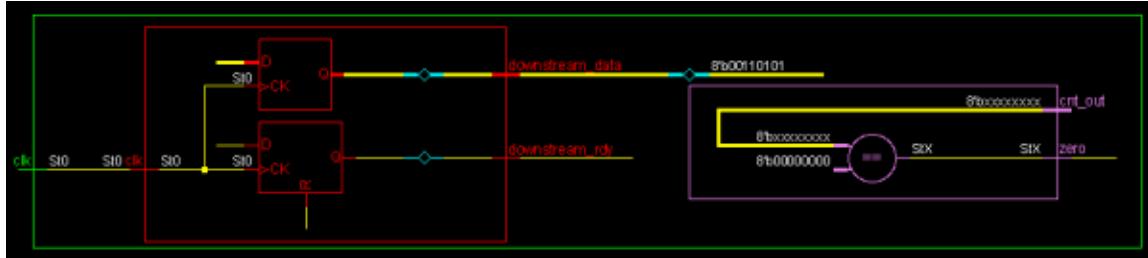
```

vlog -sv mid.v top.v
vopt -pa_upf test.upf top -debugdb
vsim -debugdb -pa -L mtiPA -vopt -do test.do top

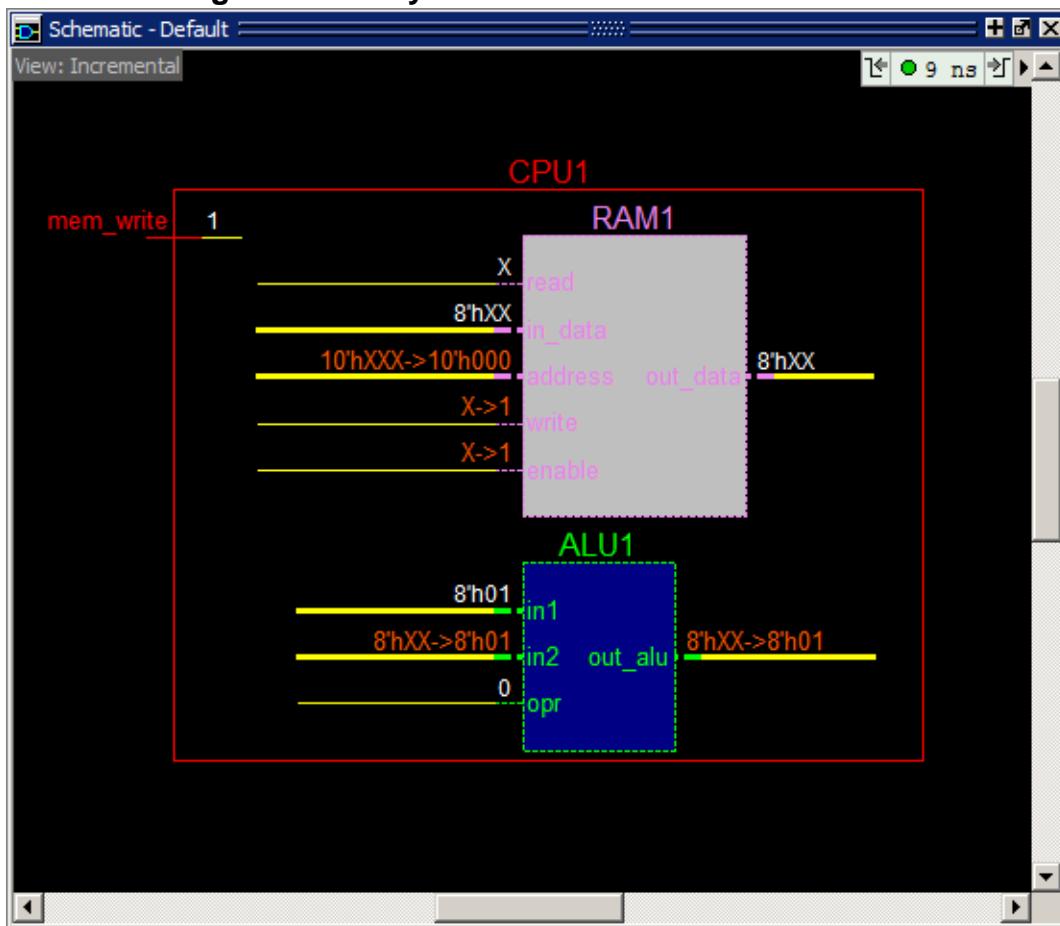
```

### Example 7-3. Schematic Displays for Power Aware Debugging

**Figure 7-2. Color-Coded HDL Design Elements**



**Figure 7-3. Gray Area Indicates a Power Domain That is Off**



**Figure 7-4. UPF Source File: Right-Click and Choose Power Domain**

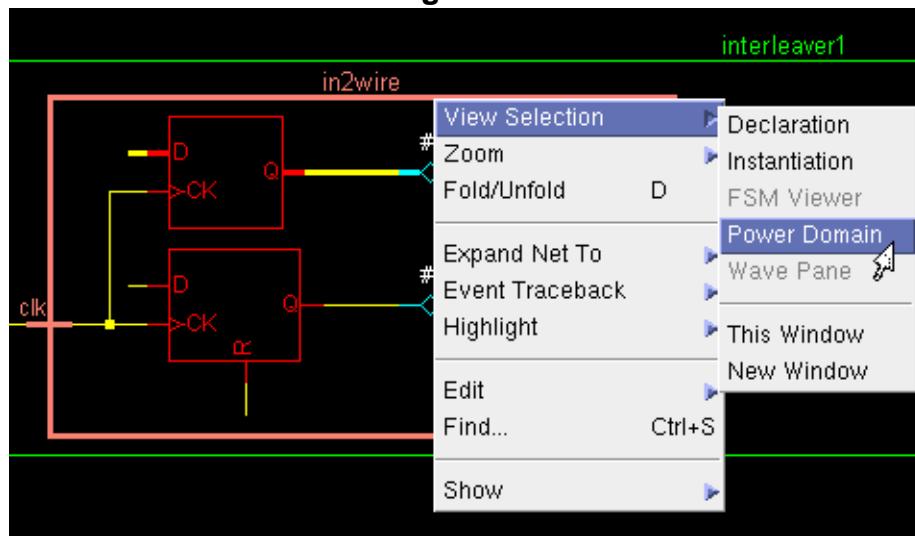
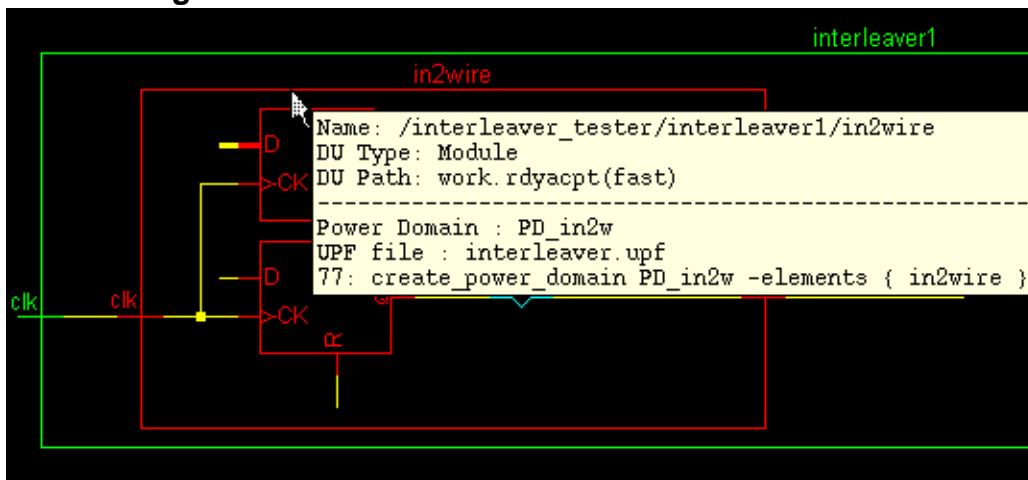


Figure 7-5. UPF Source File: Hover the Mouse and View Tool Tip



## Power Aware Waveform Display

---

You can use the Wave window to view bias mode, corruption, and isolation information for your design.

**Power Aware Waveform Concepts .....** ..... **254**

**Using Power Aware Highlighting.....** ..... **255**

## Power Aware Waveform Concepts

When using the conventional Wave window display, it can be difficult to see the effects of Power Aware simulation. For example, a zero on a signal may represent normal simulation behavior, it may be the result of an isolated port clamped to zero, or it may be corruption on a bit type.

In particular, isolation has been difficult to confirm through simulation that the intent has been met. Typically, you want waveform results to show isolation buffer placement and clamping, identify the corrupted and clamp values associated with that buffer, and confirm that isolation happens at the proper time.

To do this, you can activate Power Aware highlighting in the Wave window, which provides recognizable indicators for the isolation, corruption, and biasing behavior in your simulation results. These waveform indicators provide valuable information by visually distinguishing the values caused by Power Aware activity. This can help you determine if their power intent is correctly applied.

The highlighted indicators show the power state of signals viewed in the Wave window. Highlighting on waveforms appears during the interval when they are corrupted, isolated, or biased.

[Figure 7-6](#) show an example of waveform highlighting, where:

- Bias mode is indicated by blue highlighting
- Corruption is indicated by red cross-hatch highlighting
- Isolation is indicated by green highlighting

---

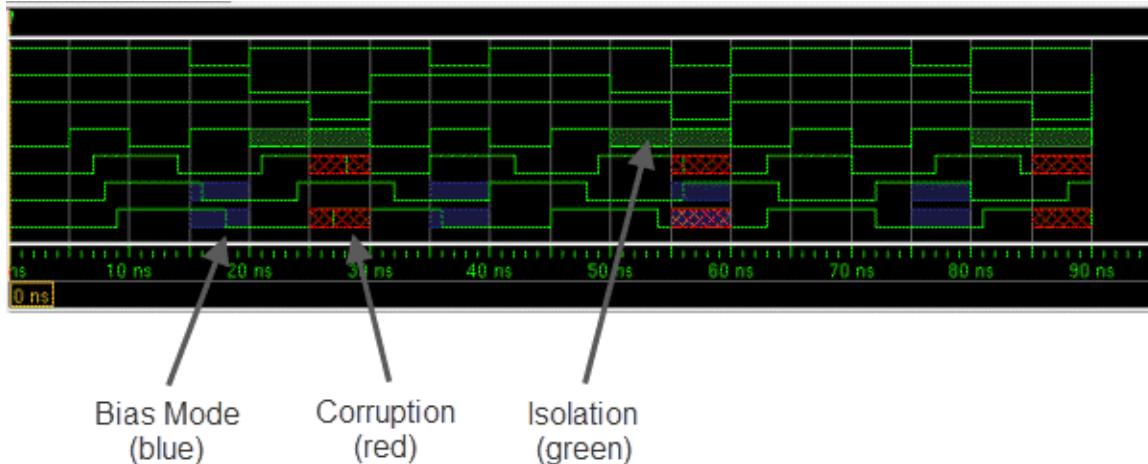
### Tip

 When you hover the mouse cursor over an isolation highlight region, a balloon popup appears. This indicates the strategy name, clamp value, and location, along with the actual signal value.

Also, when you click to expand an isolation highlighted signal (on the + to the left of the signal name), associated signals are displayed that provide more information about the isolation.

---

**Figure 7-6. Power Aware Highlighting in the Wave Window**



## Using Power Aware Highlighting

You enable Power Aware highlighting in the Wave window by altering your **vopt** and **vsim** commands and through the GUI preferences.

### Procedure

1. Enable highlighting during elaboration by adding the **-pa\_enable=highlight** argument to your **vopt** command line.
2. Enable highlighting during simulation by adding the **-pa\_highlight** argument to the **vsim** command.

This argument enables the generation of the WLF data used by the Wave window to display the highlighting.

3. Enable highlighting in the GUI by selecting:

**Wave > Wave Preferences > [Display tab] > PA waveform highlighting**

You only need to perform this action upon your first invocation, or if your GUI settings are reset.

4. View Power Aware activity in post-simulation debug by loading the WLF file containing the information as follows

**vsim -view vsim.wlf**

## Power State and Transition Display

---

Because power domains are not limited to two states (ON or OFF) and multi-voltage capability allows designs to assign different voltage levels to different states, tracking combinations of states in different power domains has become increasingly difficult.

---

### Note

---

 By default, FSM based Power Aware coverage is disabled. Use the -pa\_enable=fsmbasedcov argument with the vopt command to enable FSM based Power Aware coverage.

---

<b>Power State and Transition Concepts .....</b>	<b>256</b>
<b>Displaying Power Aware State Machines .....</b>	<b>257</b>
<b>Power Aware State Machine List Window .....</b>	<b>258</b>
<b>Power Aware State Machine Viewer Window .....</b>	<b>260</b>

## Power State and Transition Concepts

A power state table (PST) defines the allowable combinations of power states of supply ports and nets—those combinations of states that can exist at the same time during simulation of the design. As a result, changing the power state supply port/net changes the state of PST.

In UPF, you can add states on the following objects and model a Power Aware state machine (PASM) corresponding to each:

- PST — resulting from the **add\_pst\_state** UPF command
- Supply port — resulting from the **add\_port\_state** UPF command.
- Supply net — resulting when a PST contains a supply net entry.
- Power domain — resulting from the **add\_power\_state** UPF command
- Supply sets — resulting from the **add\_power\_state** UPF command

The UPF command **add\_power\_state** adds power states on power domains and supply sets. The power states associated with supply sets determine the simulation semantics of connected design elements (power domain).

Power domain elements can be in one of the following simstates based on which power states of the related supply set are true at a given simulation time: NORMAL, CORRUPT, CORRUPT\_ON\_ACTIVITY, CORRUPT\_STATE\_ON\_CHANGE or CORRUPT\_STATE\_ON\_ACTIVITY.

Typically power states are composed hierarchically while creating power intent, for example the power state of a top-level power domain depends upon power states of any contributing

power domains. These contributing power states (leaf level power domains) then depend upon power states of their supply sets.

Power states of a supply set depend upon power states of supply nets associated with functions of the supply set. Therefore, as design complexity increases, such as with IP-level hierarchical power intent, these power states become drivers for the whole power aware verification and it becomes important to provide debugging facilities for power states.

## Differences Between a Conventional RTL FSM and a PASM

Some conceptual differences between conventional RTL Finite State Machines (FSM) and Power Aware state machines are:

- Power Aware State machines are asynchronous in nature—there use no concept of clock.
- Power Aware state machines can reach multiple states at a time (Nondeterministic Situation).
- Power Aware state machines are modeled by Power Aware simulation.
- There are a few interdependent Power Aware state machines. For example, a PST state machine runs in accordance to supply port/net Power Aware state machines.

## Displaying Power Aware State Machines

The dynamic display of power states provides debugging capabilities you can use to verify power intent.

### Prerequisites

- Your UPF file contains power state information (**add\_power\_state**, **create\_pst**, **add\_port\_state**, and/or **add\_pst\_state** UPF commands).
- vopt command includes -pa\_coverage argument
- vsim command includes -coverage argument

### Procedure

1. Run your Power Aware simulation and generate results.
2. Examine the list of state machines in your design in the [Power Aware State Machine List Window](#).
3. View bubble diagrams of individual state machines in the [Power Aware State Machine Viewer Window](#).

## Power Aware State Machine List Window

Use this window to view a list of Power Aware state machines after beginning a Power Aware simulation.

This window lists all of the Power Aware State Machines in the design with the hierarchical path of their creation.

For example, if a PST has been created in **dut/top**, then this PST is visible in scope **dut/top/pa\_coverageinfo**.

This window is very similar to the [FSM List Window](#).

### Accessing

Display the window by using either of the following

- **View > PA State Machine List**
- Command: view powerstatelist

**Figure 7-7. Power Aware State Machine List Example**

The screenshot shows a software interface titled "Power Aware State Machine List". At the top, there's a toolbar with icons for file operations. Below the toolbar is a header row with columns: "Instance", "States", "Transitions", and "UPF TYPE". The main area contains a tree view on the left and a table on the right. The tree view shows a hierarchy starting from "top\_pst\_pst\_state\_info.state\_var", which has three children: "VDD\_0d99\_port\_state\_info.state\_var", "VDD\_0d81\_port\_state\_info.state\_var", and "VSS\_port\_state\_info.state\_var". The table below the tree view lists these nodes along with their state counts and transition counts, categorized by UPF type (PORT or PST). The "VSS\_port\_state\_info.state\_var" node is currently selected in the tree view.

Instance	States	Transitions	UPF TYPE
VDD_0d99_port_state_info.state_var	2	4	PORT
VDD_0d81_port_state_info.state_var	2	4	PORT
vout_p_port_state_info.state_var	3	9	PORT
VSS_port_state_info.state_var	2	4	PORT
top_pst_pst_state_info.state_var	3	9	PST
VDD_0d99_port_state_info.state_var	2	4	PORT
VDD_0d81_port_state_info.state_var	2	4	PORT
VSS_port_state_info.state_var	2	4	PORT
vout_p_port_state_info.state_var	3	9	PORT

### GUI Elements of the Power Aware State Machine List Window

This section describes GUI elements specific to this window.

### Column Descriptions

**Table 7-2. Power Aware State Machine List Window Columns**

Column	Description
Instance	Instance name of the state machine. The hierarchical display shows state dependency, specifically those instances where states of a particular power state machine depend upon states of other power state machines.
States	Number of states.
Transitions	Number of transitions between the states.
UPF Type	One of the following: PORT, NET, PST, POWER DOMAIN, SUPPLY SET

### Popup Menu

**Table 7-3. Power Aware State Machine List Window Popup Menu**

Popup Menu Item	Description
View State Machine	Displays a graphical representation of the power aware state transitions in the <a href="#">Power Aware State Machine Viewer Window</a>
View UPF	Opens the UPF file and displays the line creating the power aware state machine.
Add to	Adds the selected (or all) state machines to the Wave window, List window, or to the log (add log).
Properties	Displays the Power Aware State Machine Properties dialog box, which contains detailed information about the selection.

### PA State Machines Menu

**Table 7-4. PA State Machines Menu**

Menu Item	Description
View State Machine	Displays a graphical representation of the power aware state transitions in the <a href="#">Power Aware State Machine Viewer Window</a> .
View UPF	Opens the UPF file and displays the line creating the power aware state machine.

**Table 7-4. PA State Machines Menu (cont.)**

Menu Item	Description
Add to	Adds the selected (or all) state machines to the Wave window, List window, or to the log (add log).
Options	Opens the State Machine Display Options dialog box that enables you to control: <ul style="list-style-type: none"><li>• Specific instructions when adding state machines to the Wave window</li><li>• Limit the number of path elements in the Instance column.</li></ul>

## Power Aware State Machine Viewer Window

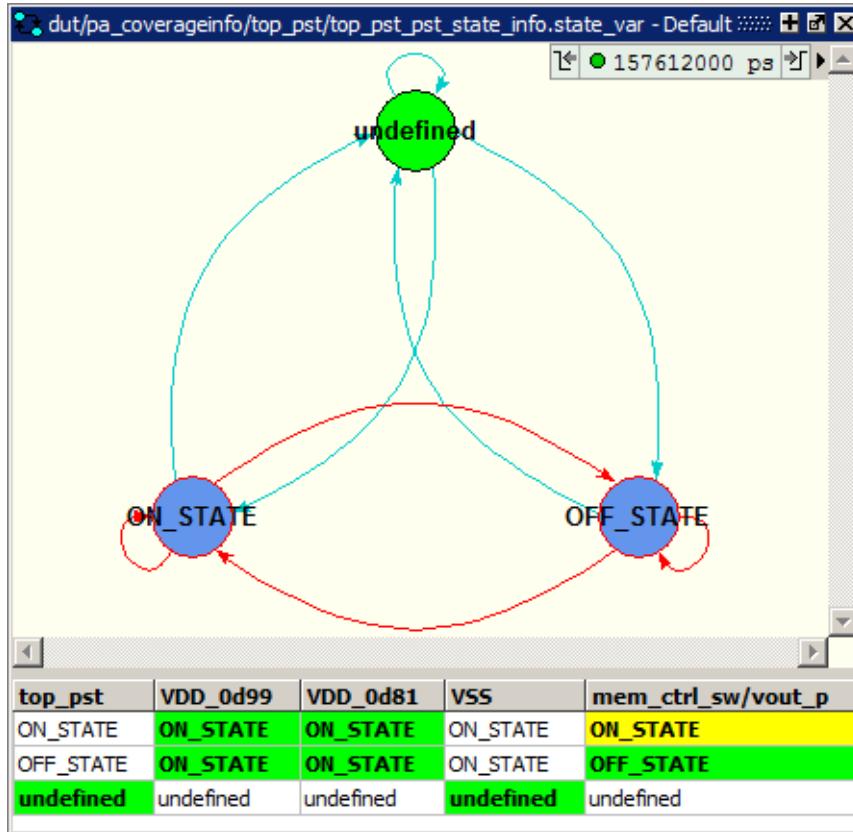
Use this window to view a state diagram of any of your Power Aware state machines.  
This window is similar to the [FSM Viewer Window](#).

### Accessing

Display the window as follows:

- From the Power Aware State Machine List Window, double click the Power Aware state machine you want to analyze.

**Figure 7-8. Power Aware State Machine Viewer Window Example**



#### GUI Elements of the Power Aware State Machine Viewer Window

This section describes GUI elements specific to this window.

#### Window Element Descriptions

- Title bar — The UPF type and hierarchical path.
- Tab label — Leaf name of the state machine.
- State Diagram — All states of a state machine and their transitions.
  - Bubbles — Information associated with logic expression supply expression simstate legality. Information appears in properties box associated with each state bubble.
  - Color — Green represents the current state and yellow represents the previous state. A thick red border identifies an illegal state (as allowed with `add_power_state`).
  - Label — State name
  - Transitions — Directional arrows showing transitions between the states.
  - Time Mode (upper right-hand corner) — The time of the current state transition, which decides yellow and green. It also affects States table.

- Goto Previous and Next State buttons — Enables synchronization between other state machine windows and the wave window and the objects window.
- Synchronization — By default, this window is synchronized with the Wave window, Objects window, and other Power Aware State Machine View windows.

**State Machine to State Machine** — State machines opened remains synchronized with each other, related to the simulation time stamp. This enables you to work with multiple power aware state machines.

**State Machine to Wave** — The Wave window and Power Aware State Machine Viewer windows are synchronized to facilitate your debugging. When you move a wave window time cursor to observe various values of signals in wave window, the Power Aware State Machine Viewer window changes to match the same simulation time stamp, and the reverse is true.

- States table — (Only appears for supply sets, power domains, and PSTs) Shows current and previous values of objects that decide the power state of the supply set or power domain. This table helps you perform root cause analysis of your power aware states. When you encounter an unexpected power state, this feature allows you to understand the reason behind it, specifically when looking for unexpected, illegal, or undefined power states.
  - Column headers — Shows objects (power domains, supply sets, supply nets, logic variables) that contribute to the power states. If a header is associated with a power domain or supply set, you can double click it to view the state transitions of that state machine. For PSTs, the column headers show ports and nets from the creation of the PST with the **create\_pst** UPF command and you can double click them to view the state transitions of that state machine.
  - Rows — Shows values for the time selected in the Time Mode widget, where green indicates the current value and yellow indicates the previous value.

#### Popup Menu

**Table 7-5. Power Aware State Machine Viewer Window Popup Menu**

Popup Menu Item	Description
Transition > View Full Text	Displays the full text of all condition expressions.
View UPF	Opens the UPF file and displays the line creating the power aware state machine.
Show States Table	Toggles the display of the states table for supply sets and power domains.
Zoom Full	Fits the bubble diagram into the visible space.

**Table 7-5. Power Aware State Machine Viewer Window Popup Menu (cont.)**

Popup Menu Item	Description
Set Context	Sets the context (env command) of the session based on your selection.
Add to	Adds the selected (or all) state machines to the Wave window, List window, or to the log (add log).
Properties	Displays the Power Aware State Machine Properties dialog box, which contains the following information about the selection. <ul style="list-style-type: none"> <li>• Encoding</li> <li>• UPF Data</li> <li>• Processes</li> <li>• Clock</li> <li>• Inputs</li> <li>• Outputs</li> </ul>

#### FSM View Menu - For Power Aware State Machines

**Table 7-6. FSM View Menu, Specific to Power Aware State Machines**

FSM View Menu Item	Description
Show States Table	Toggles the display of the states table for supply sets and power domains
Show States Counts	Displays the coverage counts for each state in the state bubble.
Show Transition Counts	Displays the coverage counts for each transition.
Show Transition Conditions	Displays the condition for each transition. The condition format is based on the <a href="#">GUI_expression_format Operators</a> .
Enable Info Mode Popups	Displays popup information when you hover over a state or transition.
Track Wave Cursor	Displays current and previous state information based on the cursor location in the Wave window.

**Table 7-6. FSM View Menu, Specific to Power Aware State Machines (cont.)**

FSM View Menu Item	Description
Transitions to “Reset”	Controls the display of transitions to a reset state: <ul style="list-style-type: none"><li>• Show All</li><li>• Show None — It also adds a “hide all” note to the lower-right hand corner.</li><li>• Hide Asynchronous Only</li><li>• Combine Common Transitions — (default) creates a single transition for any transitions to reset that use the same condition. The transition is shown from a gray diamond that acts as a placeholder.</li></ul>
Options	Displays the FSM Display Options dialog box, which allows you to control: <ul style="list-style-type: none"><li>• how FSM information is added to the Wave Window.</li><li>• how much information is shown in the Instance Column</li></ul>

# Power Aware Static Check Display

---

Use the Results Analysis window of the Questa SIM GUI to view the results of static checks after the vopt stage of your flow.

---

## Restriction

---

 Static checking results are not available for ModelSim SE.

---

<b>Viewing Static Checks in the Results Analysis Window .....</b>	<b>265</b>
<b>GUI Elements of the Results Analysis Window for Static Checks .....</b>	<b>266</b>

## Viewing Static Checks in the Results Analysis Window

When you instruct the optimization step of the Power Aware flow to perform static checks, the output consists of a database file (*report.static.tdb*) containing information about your design. Use this database file to view the static check results in the Results Analysis window.

---

## Restriction

---

 Static check results are not available for ModelSim SE.

---

### Prerequisites

- Enable static checks with the `-pa_checks` argument to vopt:

```
vopt -pa_checks=s
```

The *report.static.tdb* file is generated after the completion of the vopt command.

### Procedure

1. Open the Questa SIM GUI
2. Select **File > Open** to display the Open File dialog box
  - a. Change the filetype dropdown to **All Files**
  - b. In the navigation window select: *report.static.tdb*
  - c. Click **Open**The main Questa SIM GUI opens, containing your static check results.
3. Select a pre-defined configuration
  - a. Right-click in the Results Analysis window and select **Analysis Questions** to display the Analysis Questions dialog box.
  - b. Select one of the pre-defined configurations.

- c. Click **Apply**
  - d. Click **Done**
4. (Optional) View the source-to-sink path of a static check in the Dataflow window:

Right-click the result you want to investigate and select:**View > Add to Dataflow**.  
(Double-clicking on the result also adds the information to the Dataflow window.)

This runs the [add dataflow -connect](#) command to show the path between the Source Port and Sink Port. Refer to “[Dataflow Window](#)” in the *User’s Manual* for more information about this window. This is available only if the design is loaded for simulation.

## Related Topics

[Static RTL Checks](#)

[Static Check Report](#)

## GUI Elements of the Results Analysis Window for Static Checks

The Results Analysis window provides display features that are specific to the loading of a *report.static.tdb* file.

---

### Restriction

---

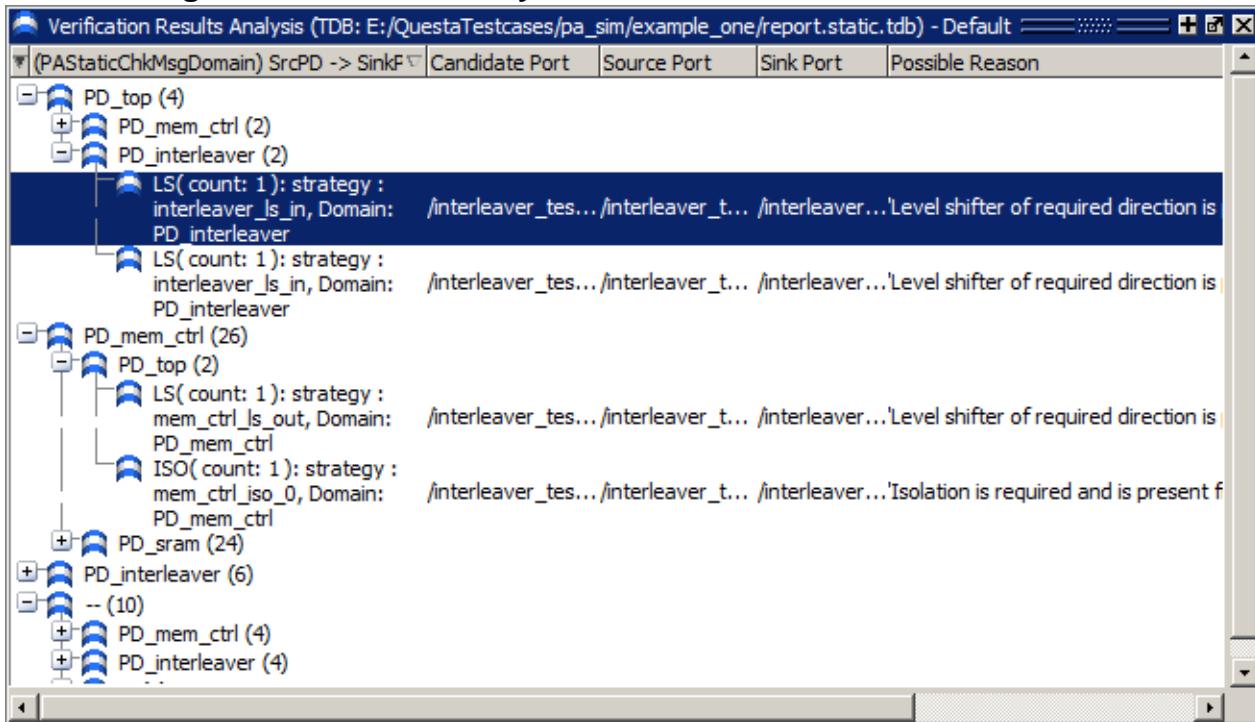


This database file (*report.static.tdb*) is not available for ModelSim SE.

---

Refer to “[Verification Results Analysis Window](#)” in the *User’s Manual* for more general information.

**Figure 7-9. Results Analysis Window With Static Check Results**



## Power Aware Specific Columns

The following columns are specific to the loading of a *report.static.tdb* file.

- Candidate port — Hierarchical port information.
- Possible Reason — Text description of the reason for the static check result.
- Sink PD — Name of the sink power domain.
- Sink Port — Hierarchical port information.
- Source Port — Hierarchical port information.
- Src PD — Name of the source power domain.
- Violation type — Type of violation, such as Valid or Not analyzed.

## Pre-defined Analysis Questions

After loading the TDB file you can choose between two pre-defined configurations of the window, by selecting an Analysis Question. They are:

- Show PA Static Check Messages - Domain Wise

The results are sorted in order of Source Power Domain, Sink Power Domain, then Message.

- Show PA Static Check Messages - Strategy Wise

The results are sorted in order of Cell Type (such as isolation or level shifter), violation type (such as valid or not analyzed), Source Power Domain, and Sink Power Domain

# Chapter 8

## Power Aware Information Model

---

Power Aware information model captures the power-management information, which is the result of application of UPF, Liberty, and HDL commands on the user design. The information is captured in a standard form that can be queried via Tcl commands and HDL package functions.

---

### Note

---

 Power Aware simulation uses the UPF 3.0 version of the UPF packages and supports the same set of UPF package definitions.

---

<b>Creating and Loading Power Aware Information Model Database</b> .....	<b>269</b>
<b>Tcl Commands</b> .....	<b>270</b>
Running Tcl Commands in the Interactive Mode.....	270
Running Tcl Commands From the VSIM Prompt .....	270
Supported Tcl Commands .....	272
<b>HDL Package Functions</b> .....	<b>275</b>
Supported HDL Package Functions .....	275
Using the HDL Package Functions for Power Aware Coverage .....	277

## Creating and Loading Power Aware Information Model Database

Create and load the Power Aware information model database in the simulator to use the Tcl commands and UPF 3.0 package functions.

### Procedure

1. Compile using your existing commands.
2. Optimize using your existing vopt command with the following argument to create the Power Aware information model database, *design.bin.padb*, in the current working directory:

```
vopt <existing_vopt_arguments> -pa_dumpimdb
```

3. Simulate using your existing vsim command with the following argument to load the database in the simulator:

```
vsim <existing_vsim_arguments> -pa_loadimdb
```

## Tcl Commands

The Tcl commands enable you to access the Power Aware information model objects and its properties either in the interactive mode or from the VSIM prompt. You can use the Tcl commands with any previous version of the UPF standard (such as UPF 2.1 or UPF 2.0). Although the output format and properties of the commands are dependent on the current UPF standard (UPF 3.0).

<b>Running Tcl Commands in the Interactive Mode .....</b>	<b>270</b>
<b>Running Tcl Commands From the VSIM Prompt.....</b>	<b>270</b>
<b>Supported Tcl Commands .....</b>	<b>272</b>

## Running Tcl Commands in the Interactive Mode

Run the Tcl commands in the interactive mode to access the Power Aware information model objects and its properties.

### Prerequisites

- Compile your design files. See step 1 in the “[Using the Standard Power Aware Simulation Flow](#)” topic.
- Create Power Aware information model database. See step 2 in the “[Creating and Loading Power Aware Information Model Database](#)” topic.

### Procedure

1. Activate the interactive mode using the following command:

```
vopt -pa_interactive
```

See “[Power Aware in Interactive Mode](#)”.

2. Run the Tcl commands.

For example:

```
PA > upf_query_object_properties /alu/dut/PD_TOP
PA > upf_query_object_type /alu/dut/PD_RAM
```

## Running Tcl Commands From the VSIM Prompt

Run the Tcl commands from the VSIM prompt to access the Power Aware information model objects and its properties.

### Prerequisites

- Create and load the Power Aware information model database. See “[Creating and Loading Power Aware Information Model Database](#)”

## Procedure

Run the Tcl commands from the VSIM prompt.

For example:

```
VSIM > upf_query_object_properties /alu/dut/PD_TOP
VSIM > upf_query_object_type /alu/dut/PD_RAM
```

## Supported Tcl Commands

The simulator supports the syntax and semantics of the basic UPF query commands defined by IEEE Std P1801-2015, Section 11.1.2.

---

### Note

 To query a supply or logic net whose name is same as that of a supply or logic port, suffix @upfSupplyNet or @upfLogicNetT string, respectively:

```
upf_query_object_properties /tb/logic_netport@upfLogicNetT  
upf_query_object_type /tb/logic_net1@upfSupplyNetT
```

---

**Table 8-1. Supported Tcl Commands**

Command	Description
<a href="#">upf_object_in_class</a>	Checks if the object handle belongs to a specified class.
<a href="#">upf_query_object_pathname</a>	Returns the string representing the hierarchical pathname of an object in the Power Aware database, relative to the given scope.
<a href="#">upf_query_object_properties</a>	Returns a string containing the value of the specified property (keyword) on an object in the Power Aware database.
<a href="#">upf_query_object_type</a>	Returns the type of an object in the Power Aware database.

## upf\_object\_in\_class

Checks if the object handle belongs to a specified class.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

### Options

Option	Comments/Restrictions
<object_handle>	
-class <CLASS_ID>	

## upf\_query\_object\_pathname

Returns the string representing the hierarchical pathname of an object in the Power Aware database, relative to the given scope.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

### Options

Option	Comments/Restrictions
object_handle	
-relative_to <object_handle>	

## upf\_query\_object\_properties

Returns a string containing the value of the specified property (keyword) on an object in the Power Aware database.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

### Options

Option	3.0	2.1	2.0	1.0	Description
<object_handle>	Y	Y	Y	Y	
-property property_keyword	Y	Y	Y	Y	

Option	3.0	2.1	2.0	1.0	Description
-verbose	Y	N/A	N/A	N/A	<p>Prints empty attribute names. The name-value pairs of the attributes are separated by newline. Extents have a full hierarchical path. For example: #EXTENT123#/(/tb/top1).</p> <p> <b>Note:</b> The option is not a part of the IEEE Std 1801.</p>

## upf\_query\_object\_type

Returns the type of an object in the Power Aware database.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

### Options

Option	Comments/Restrictions
<object_handle>	

# HDL Package Functions

The HDL package functions enable you to access the Power Aware information model objects and its properties directly in a testbench or simulation model.

**Supported HDL Package Functions.....** ..... [275](#)

**Using the HDL Package Functions for Power Aware Coverage.....** ..... [277](#)

## Supported HDL Package Functions

The simulator supports the syntax and semantics of the HDL package functions defined by IEEE Std P1801-2015, Section 11.2.3.

### Supported UPF 3.0 HDL Package Functions

#### Note

 To use the UPF 3.0 HDL package functions, you need to create the Power Aware information model database. See “[Creating and Loading Power Aware Information Model Database](#)”.

**Table 8-2. List of Supported UPF 3.0 HDL Package Functions for SystemVerilog**

Function Type	Function Name	Comment
Immediate read access	get_supply_on_state	
	get_supply_state	
	get_supply_value	
	get_supply_voltage	
	upf_get_all_power_domain	Returns all the power domains present in the design.   <b>Note:</b> The option is not a part of the IEEE Std 1801.
	upf_get_value_int	
	upf_get_value_str	
Immediate write access	set_supply_state	
	supply_on	
	supply_off	
	supply_partial_on	
Continuous access	upf_create_object_mirror	Write access is not supported.

**Table 8-2. List of Supported UPF 3.0 HDL Package Functions for SystemVerilog (cont.)**

Function Type	Function Name	Comment
HDL access	upf_get_handle_by_name	
	upf_iter_get_next	
	upf_query_object_properties	See “ <a href="#">List of Unsupported Properties</a> ”.
Utility	upf_query_object_pathname	
	upf_query_object_type	

[Table 8-3](#) lists the unsupported properties of the upf\_query\_object\_properties function of the UPF 3.0 package.

**Table 8-3. List of Unsupported Properties**

Unsupported Property	Unsupported Property
UPF_EQUIVALENT_SETS	UPF_PD_STATE_TRANSITIONS
UPF_EXPR_OPERANDS	UPF_REF_KIND
UPF_EXPR_STRING	UPF_REF_OBJECT
UPF_FANIN_CONN	UPF_RETENTION_PARAMETERS
UPF_FANOUT_CONN	UPF_ROOT_DRIVER
UPF_FROM_STATES	UPF_SLICE_BITS
UPF_HICONN	UPF_SMALLEST_ATOMIC_SLICE
UPF_IS_USE_EQUIVALENCE	UPF_SS_TRANSITIONS
UPF_LOCONN	UPF_SUBDOMAINS
UPF_LOGIC_REFS	UPF_SWITCH_EXPR
UPF_NAME_PREFIX	UPF_TO_STATES
UPF_NAME_SUFFIX	UPF_UPPER_BOUNDARY
UPF_NETWORK_ATTRIBUTES	UPF_PST_HEADER
UPF_NEXT_EXTENT	UPF_PST_STATES
UPF_NORMALIZED_BITS	

## Supported UPF 2.0 HDL Package Functions

---

### Note

---

 To use the UPF 2.0 HDL package functions, you do not need to create the Power Aware information model database.

---

Following is the list of supported UPF 2.0 HDL package functions for systemVerilog and VHDL:

- supply\_on
- supply\_off
- supply\_partial\_on
- get\_supply\_value
- get\_supply\_voltage
- get\_supply\_on\_state
- get\_supply\_state

## Using the HDL Package Functions for Power Aware Coverage

Use the UPF 3.0 HDL package functions with the SystemVerilog functional coverage constructs, such as covergroups and coverpoints, for efficient Power Aware coverage. The HDL package functions enable you to write fast and reliable Power Aware coverage infrastructure, and perform random and directed Power Aware simulation. Random because the test scenarios are developed by generic scripts, and directed because the coverage covers very specific scenarios.

### Procedure

---

### Note

---

 The topic shows Power Aware coverage of the states and transitions of all power domains using the HDL package functions.

---

1. Get the handle of the UPF object in your testbench.

For example:

```
pd_iter = upf_get_all_power_domains();
```

2. Get the dynamic property of the UPF object by using the continuous access HDL package function in your testbench. Use assertions for anomalies and other scenarios of interest.

For example:

```
upfPdObjT pd_obj;
pd_hdl = upf_iter_get_next(pd_iter);
$display ("--pd is added:%s", upf_query_object_pathname(pd_hdl));
upf_create_object_mirror (upf_query_object_pathname(pd_hdl),
    pd_obj);
```

3. Create the coverage module with covergroups and coverpoints in your testbench. Pass the handle of the UPF object (extracted in step 1) and its dynamic properties (extracted in step 2) to the coverage module.

The coverage module, which calculates the coverage metrics, is modeled in SystemVerilog and compiled together with the design. Instantiate as many coverage instances as required.

For example:

```
covergroup PD_STATE_COVERAGE (string pd_name, ref upfSimstateE
simstate) @(`simstate);
    CORRUPT: coverpoint simstate
        { bins ACTIVE = {CORRUPT}; }
    NORMAL: coverpoint simstate
        { bins ACTIVE = {NORMAL}; }
endgroup

covergroup PD_TRANS_COVERAGE (string pd_name, ref upfSimstateE
simstate) @(`simstate);
    TRANSITION_COVERAGE:coverpoint simstate
    {
        bins OFF_to_ON = (CORRUPT => NORMAL);
        bins ON_to_OFF = (NORMAL => CORRUPT);
    }
endgroup

pd_state_cov = new (upf_query_object_pathname (pd_obj.handle),
pd_obj.simstate);pd_trans_cov = new (upf_query_object_pathname
(pd_obj.handle), pd_obj.simstate);
```

4. Create and load the Power Aware information model database of your design. See “[Creating and Loading Power Aware Information Model Database](#)”.
5. View the coverage results using the following command in the simulator:

```
coverage report -details
```

## Results

The following section shows the coverage results that is generated using the HDL package functions with the SystemVerilog functional coverage constructs.

```
# COVERGROUP COVERAGE:  
#-----  
# Covergroup Metric Goal Status  
#-----  
# TYPE /tb/PD_STATE_COVERAGE 83.33% 100 Uncovered  
#   covered/total bins:      5       6  
#   missing/total bins:     1       6  
#   % Hit:                 83.33% 100  
# Coverpoint PD_STATE_COVERAGE::CORRUPT 100.00% 100 Covered  
#   covered/total bins:      2       2  
#   missing/total bins:     0       2  
#   % Hit:                 100.00% 100  
# Coverpoint PD_STATE_COVERAGE::NORMAL 100.00% 100 Covered  
#   covered/total bins:      2       2  
#   missing/total bins:     0       2  
#   % Hit:                 100.00% 100  
...  
...
```



# Appendix A

## Power Aware-related Commands

---

This appendix provides information on various Questa SIM and Tcl commands that you use to control your Power Aware simulation.

<b>Questa SIM Commands and Arguments</b> .....	<b>282</b>
<b>Tcl Commands</b> .....	<b>317</b>
<b>Voltage Level-Shifting (Multi-Voltage Analysis)</b> .....	<b>332</b>
<b>Isolation and Level Shifting Restriction on a Port</b> .....	<b>336</b>
<b>Simulation of Designs Containing Macromodels</b> .....	<b>341</b>

# Questa SIM Commands and Arguments

---

Power Aware simulation supports several Questa SIM commands that are specific to Power Aware simulation. You can also use various arguments with the Questa SIM commands for Power Aware simulation.

---

## Note

---

 The Questa SIM commands follow the given syntax:

---

```
<questa_command> -<argument> <value>
```

---

<b>Power Aware-specific Arguments to vopt and vsim.....</b>	<b>282</b>
<b>Usage of vopt -pa_enable and -pa_disable.....</b>	<b>285</b>
<b>Argument to Exclude Design Element.....</b>	<b>297</b>
<b>Command to Control Dynamic Checks.....</b>	<b>300</b>
<b>Arguments to Control Power Aware Message Severity.....</b>	<b>308</b>
<b>Supported UPF Extensions .....</b>	<b>309</b>

## Power Aware-specific Arguments to vopt and vsim

Control your Power Aware simulation with the Power Aware-specific arguments specified to the vopt and vsim commands.

[Table A-1](#) and [Table A-2](#) list the arguments for the vopt and vsim commands that you use to run a Power Aware simulation. Refer to the *Questa SIM Command Reference Manual* for information on these commands and their arguments.

**Table A-1. Power Aware Arguments for vopt**

vopt Argument	Argument Value(s)
-pa_checks=	<check_value> Refer to “ <a href="#">Quick Reference of Static RTL and Dynamic Checks</a> ” for more information.
-pa_checksoption=	<value>
-pa_connectpgpin=	i   a   e   c
-pa_corrupt=   -pa_nocorrupt=	real   integer
-pa_coverage	[=checks] [=crosscov] [=iso] [=portpststate] [=powerstate] [=ret] [=switch]
-pa_coverageoff	None
-pa_crosscoveredagedepth	<integer>

**Table A-1. Power Aware Arguments for vopt (cont.)**

<b>vopt Argument</b>	<b>Argument Value(s)</b>
-pa_dbgfindobj=	<filename>
-pa_defertop	None
-pa_disable=	<value>[+<value>]... Refer to “ <a href="#">Usage of vopt -pa_enable and -pa_disable</a> ” for more information.
-pa_dumpimdb	None
-pa_dumplibertydb=	<database_filename>
-pa_dumpupf	<filename>
-pa_enable=	<value>[+<value>]... Refer to “ <a href="#">Usage of vopt -pa_enable and -pa_disable</a> ” for more information.
-pa_excludefile	<filename> Refer to “ <a href="#">Argument to Exclude Design Element</a> ” for more information.
-pa_genrpt=	[ud] [pa[+b]] [de[+b]] [cell] [conn] [srcsink] Refer to “ <a href="#">Power Aware Reports Reference</a> ” for more information.
-pa_gls	None Required for gate-level designs only.
-pa_hiersep	<alphanum_character>
-pa_intcrptval0	None
-pa_interactive	None
-pa_lib	<library_pathname>
-pa_libertyfiles=	<liberty_filename>,...
-pa_libertyrefresh=	<database_filename>
-pa_libertynestedcmnts	None
-pa_libertyupdate	None
-pa_lsthreshold	<real>
-pa_loadlibertydb=	<database_filename>
-pa_maxregsize	<integer>
-pa_maxpstcol	<integer> Refer to “ <a href="#">PST Analysis Report</a> ” for more information.

**Table A-1. Power Aware Arguments for vopt (cont.)**

vopt Argument	Argument Value(s)
-pa_noinpcorr	None Refer to “ <a href="#">Liberty Models of Hard Macros</a> ” for more information.
-pa_nopartialontrans	None Refer to “ <a href="#">set_partial_on_translation</a> ” for more information.
-pa_pgoutsynchchk	None Refer to “ <a href="#">Synchronization of Internal Power/Ground Pin Sources</a> ” for more information.
-pa_powertop	<work_lib.power_module_name>
-pa_prefix	Deprecated in UPF 2.1. Use -pa_top instead.
-pa_preupffile	<filename>
-pa_pstcompflags=	{p   g   pg   pstids   useallps} Refer to “ <a href="#">Power State Tables</a> ” and “ <a href="#">PST Analysis Report</a> ” for more information.
-pa_replacetop	Deprecated in UPF 2.1. Use -pa_top instead.
-pa_reportdir	<pathname>
-pa_rslvnetcheck	None Refer to “ <a href="#">set_partial_on_translation</a> ” for more information.
-pa_rtlinfo	None
-pa_staticchecksonly	None
-pa_tclfile	<filename>
-pa_top	<pathname>
-pa_upf	<filename>
-pa_upfextensions=	<extension>[+<extension>] Refer to “ <a href="#">Supported UPF Extensions</a> ” for more information.
-pa_upflist	<file_list> This argument is mutually exclusive with -pa_upf.
-pa_upfoverloadsourcemd	None
-pa_upfsanitychecks	None
-pa_upfversion=	1.0   2.0   2.1   3.0 The default argument value is 2.0.

**Table A-1. Power Aware Arguments for vopt (cont.)**

<b>vopt Argument</b>	<b>Argument Value(s)</b>
<b>-pa_zcorrupt</b>	None

**Table A-2. Power Aware Arguments for vsim**

<b>vsim Argument</b>	<b>Argument Value</b>
<b>-pa</b>	None
<b>-pa_allowtimezeroevent=</b>	all
<b>-pa_debugdir</b>	<directory>
<b>-pa_disabletimezeroevent</b>	None
<b>-pa_excludedefaultps</b>	None
<b>-pa_gls</b>	<testbench_top> Earlier used for gate-level designs. Use the -pa argument instead.
<b>-pa_highlight</b>	None
<b>-pa_includeundefined</b>	None
<b>-pa_lib</b>	<pathname>
<b>-pa_loadimdb</b>	None
<b>-pa_logfile</b>	<filename>
<b>-pa_togglelimit=</b>	<integer>

## Usage of vopt -pa\_enable and -pa\_disable

Use the **-pa\_enable** and **-pa\_disable** arguments of the **vopt** command to enable or disable certain actions that are performed during Power Aware simulation.

**Table A-3** lists the possible values for these arguments to the **vopt** command and the syntax is as follows:

```
vopt -pa_enable=<value>[+<value>...]
vopt -pa_disable=<value>[+<value>...]
```

### Usage Notes

- Both **-pa\_enable** and **-pa\_disable** use the same set of values, which means using a value with either argument toggles that action from its default state or from a previously specified state.

- Table A-3 lists each value, a brief description of the Power Aware action performed, and the default argument.
- Specify one or more values for either argument—there is no order dependency when specifying multiple values. To specify more than one value for either argument, use the + operator between the values. For example:

```
vopt -pa_enable=anonupfobjects+insertiso
vopt -pa_disable=ackportbehavior+detectiso
```

**Table A-3. Power Aware Actions for vopt -pa\_enable and -pa\_disable**

Value	Action	Default
ackportbehavior	For UPF 2.0, power switches with the -ack_port switch require a supply set. <ul style="list-style-type: none"> <li>• -pa_enable — Issues an error if this condition is not met.</li> <li>• -pa_disable — Allows violation of this condition and uses an always-on supply set, essentially reverting to UPF 1.0 functionality.</li> </ul>	-pa_enable
alwaysonbuf	Turns off corruption of buf primitives.	-pa_disable
anonupfobjects	Creates anonymous supply sets or nets for future replacement by associated objects.	-pa_enable
aoncellcrpt	Disables corruption of always-on Liberty cells.  Corruption semantics of always-on cells are disabled when the Liberty model for these cells is not present.	-pa_enable
appliestowithsrcsink	Enables you to use the -applies_to option with -source or -sink on the <a href="#">set_isolation</a> and <a href="#">set_level_shifter</a> UPF commands, which is an error according to UPF 2.0.	-pa_disable

**Table A-3. Power Aware Actions for `vopt -pa_enable` and `-pa_disable` (cont.)**

Value	Action	Default
assertduringpoweroff	<p>Enables assertions present in the testbench or design instance during power down.</p> <p>By default, assertions are disabled during power down. However, if the testbench or design instance has any assertion control tasks (such as asserton, assertoff, assertkill, and so on), then the assertions are controlled by the assertion control tasks, and the simulator displays the following warning message:</p> <pre>** Warning: (vopt-9959) Assertions during power off will not be disabled due to presence of assertion control tasks in the design.</pre> <p> <b>Note:</b> If you specify both <code>assertduringpoweroff</code> and <code>forceasrtoffpwrdown</code> values to the <code>-pa_enable</code> argument, then the <code>assertduringpoweroff</code> value takes priority.</p>	<code>-pa_disable</code>
autoconnls	Connects the unconnected pg pins of the ELS and level shifter cell to the always-on power supply created by the simulator.	<code>-pa_disable</code>
autoconnpacell	Connects all unconnected supply ports of a Power Aware cell to the primary supply of the power domain of the cell.	<code>-pa_disable</code>
autotestplan   <b>Restriction:</b> Not available for ModelSim SE (Questa SIM only).	Enables or disables the generation of a test plan for Power Aware verification management. See “ <a href="#">Analyzing Coverage Using the Power Aware Test Plan</a> ”.	<code>-pa_disable</code>
behaveseqcrpt	<p>Enables corruption of behavioral sequential blocks. Behavioral sequential blocks are those always blocks that are edge sensitive and cannot be synthesized.</p> <p>By default, the corruption of behavioral sequential blocks is disabled, and the simulator displays the following warning message:</p> <pre>** Warning: ./src/counter.v(5): (vopt-9031) Ignoring Corruption of Behavioral Sequential block in module 'counter'.</pre>	<code>-pa_disable</code>

**Table A-3. Power Aware Actions for `vopt -pa_enable` and `-pa_disable` (cont.)**

<b>Value</b>	<b>Action</b>	<b>Default</b>
biascorruption	Enables recognition of bias supplies (such as pwell, nwell, deeppwell, deepnwell) for Power Aware corruption.	-pa_disable
bndrypacell	Enables placement of isolation, level shifter and buffer cells on the boundary ports of switch and retention cells.	-pa_disable
bufdebug	<p>Enables you to view the buffer in the Schematic window of the simulator, which the following commands insert:</p> <ul style="list-style-type: none"> <li>• <code>set_related_supply_net</code></li> <li>• <code>set_port_attributes -driver_supply/-receiver_supply/-repeater_supply</code></li> </ul> <p>If you do not enable bufdebug, you cannot view the buffer in the Schematic window but the buffer information is still available in <i>report.pa.txt</i> file:</p> <p>Signals with Buffer cells:  1. Signal: /tb/top/B1_LINK_TOP/dout, buffer cell: /tb/top/B1_LINK_TOP/dout_UPF_RS</p>	-pa_disable
cellonsimdisable	Enables placement of cells (such as isolation, level-shifter, repeaters) on the simstate disabled instances.	-pa_disable
coadebug	Enables display of debug messages for CORRUPT_ON_ACTIVITY based simstate corruption.	-pa_disable
coainputcrpt	Enables corruption of flip-flops when the value of D input changes with a change in the CORRUPT_ON_ACTIVITY or the CORRUPT_STATE_ON_ACTIVITY simstate.	-pa_disable
conninsidemacro	Connects the unconnected pg pins of the cells present inside a macro cell to the always-on power supply created by the simulator.	-pa_disable
constasfeedthru	Prevents corruption of signals driven by constants.	-pa_enable
controlrs	Enables receiver supply of control signals according to strategy supply.	-pa_disable

**Table A-3. Power Aware Actions for `vopt -pa_enable` and `-pa_disable` (cont.)**

<b>Value</b>	<b>Action</b>	<b>Default</b>
crosscoverage	Disables cross coverage for Power Aware simulation. This is an alternate method of specifying <code>vopt -pa_coverageoff=crosscov</code> .	<code>-pa_enable</code>
csnopt	Enables optimization of <a href="#">connect_supply_net</a> UPF command.	<code>-pa_disable</code>
customblret	Enables custom retention models in balloon-latch retention configuration to handle zero pin retention requirements.	<code>-pa_disable</code>
custommsret	Enables custom retention models in master-slave retention configuration to handle zero pin retention requirements.	<code>-pa_disable</code>
debug	Enables the appearance of UPF objects created in the Structure, Object, and Wave windows.	<code>-pa_disable</code>
defaultlssupplies	Enables corruption of level-shifter with default supplies.	<code>-pa_disable</code>
defaultoff	Changes the default state of supply nets and ports to either OFF or FULL_ON <ul style="list-style-type: none"> <li>• <code>-pa_enable=defaultoff</code> sets default to OFF</li> <li>• <code>-pa_disable=defaultoff</code> sets default to FULL_ON</li> </ul>	<code>-pa_enable</code>
detectaon	Detects the always-on cells present in the design.	<code>-pa_enable</code>
detectiso	Detects isolation cells present in the design.	<code>-pa_enable</code>
detectret	Detects retention cells present in the design.	<code>-pa_enable</code>
detectls	Detects level-shifter cells present in the design.	<code>-pa_enable</code>
detectsw	Detects switches instantiated in the design and their association with a UPF <a href="#">create_power_switch</a> command.	<code>-pa_enable</code>
donottouchassertinbind	Prevents assertduringpoweroff and forceasrtoffpwrdown value to affect the assertions present in the bind instances and modules during power down.	<code>-pa_disable</code>

**Table A-3. Power Aware Actions for vopt -pa\_enable and -pa\_disable (cont.)**

<b>Value</b>	<b>Action</b>	<b>Default</b>
forceasrtoffpwrdown	Disables assertions present in the testbench and design instance during power down, even if the testbench or design instance has assertion control directives, such as asserton, assertoff, assertkill, and so on. See “assertduringpoweroff”.	-pa_disable
fsmbasedcov	Enables FSM based Power Aware coverage. See “ <a href="#">Power State and Transition Display</a> ”.	-pa_disable
highlight	Enables highlighting in the Wave window.	-pa_disable
ignoreconflictsupply	Ignores the driver supply, set by the <a href="#">set_port_attributes -driver_supply</a> or <a href="#">set_related_supply_net</a> command, to avoid flagging of an error when this driver supply is different from the actual driver supply of the port.	-pa_disable
ignorespecialdrivers ignoregroundsupplyconn ignorepowersupplyconn ignoretielow ignoretiehigh	Prevents the application of level shifter and isolation cell strategies based on the following scenarios: <ul style="list-style-type: none"> <li>• Ports connected to ground supplies</li> <li>• Ports connected to power supplies</li> <li>• Ports tied to 0</li> <li>• Ports tied to 1</li> </ul> It also does not perform any static level shifter, isolation checking or dynamic checks (missing level-shifter or missing isolation).	-pa_disable

**Table A-3. Power Aware Actions for vopt -pa\_enable and -pa\_disable (cont.)**

<b>Value</b>	<b>Action</b>	<b>Default</b>
ignorehangingoutput	<p>Ignores static/dynamic analysis and the application of isolation/level-shifter strategies for paths that have hanging output.</p> <p>An output is hanging if either of the following is true:</p> <ul style="list-style-type: none"> <li>• The actual (highconn) of port is not driving any logic.</li> <li>• The lowconn side of port is not being driven by any logic.</li> </ul> <p>Note that if the driver or receiver supply of a port is known then such port is not treated as a hanging output port. Therefore, this value does not ignore the static/dynamic analysis and application of isolation/level-shifter strategies for such ports.</p>	-pa_disable
ignoresupplyconn	<p>Ignores static/dynamic analysis and the application of isolation/level-shifter strategies when ports are connected to power/ground supplies.</p>	-pa_enable
ignoreundriveninput	<p>Ignores static/dynamic analysis and the application of isolation/level-shifter strategies for paths that have undriven input.</p> <p>An input is undriven if either of the following is true.</p> <ul style="list-style-type: none"> <li>• The actual (highconn) of the port is not being driven by any logic</li> <li>• The lowconn side of port is not driving any logic</li> </ul> <p>Note that if the driver or receiver supply of a port is known then such a port is not treated as an undriven input port. Therefore, this value does not ignore the static/dynamic analysis and application of isolation/level-shifter strategies for such ports.</p>	-pa_disable
immedcrpt	Corrupts value in continuous delay assign statements instantly at power down.	-pa_disable

**Table A-3. Power Aware Actions for **vopt -pa\_enable** and **-pa\_disable** (cont.)**

<b>Value</b>	<b>Action</b>	<b>Default</b>
implicitportnet	Enables coverage of implicit port and net states.  The simulator creates implicit port and net states for the supply ports, supply nets, and/or supply set handle functions that are used in the –supply_expr option of the add_power_state command.	-pa_enable
inoutsupplyconn	Disables the inout functionality between connections involving a UPF supply net and an HDL inout port.  Specifically, the connection is made in such a way that all existing HDL drivers of the inout port are removed, leaving only the UPF supply net to drive the HDL inout port.	-pa_enable
insertiso	Inserts isolation cells.	-pa_enable
insertret	Controls retention cell insertion. <ul style="list-style-type: none"> <li>• -pa_enable=insertret enables retention cell insertion in the gate-level flow (-pa_gls).</li> <li>• -pa_disable=insertret disables retention cell insertion in the RTL or mixed RTL/gate-level flows.</li> </ul>	-pa_enable
insertls	Inserts level-shifter cells.	-pa_enable
libertycellcrpt	Enables treating any HDL cell as a hard macro, if the Liberty model is present. Honors Liberty-based corruption (based on power_down_function and related_supplies).	-pa_disable
libertyopt	Enables optimization of handling of Liberty PG ports.	-pa_disable

**Table A-3. Power Aware Actions for `vopt -pa_enable` and `-pa_disable` (cont.)**

<b>Value</b>	<b>Action</b>	<b>Default</b>
libertypamodel	<p>Enables use of Liberty model pg_pin/ pg_type definitions and attributes that define Power Aware behavior.</p> <p>Applies several modeling consistency and application checks, including:</p> <ul style="list-style-type: none"> <li>• checking that automatic connections are not applied to internal pg pins.</li> <li>• checking that any supply net connected to one or more internal pg pins is resolved appropriately.</li> </ul> <p>See “<a href="#">Synchronization of Internal Power/Ground Pin Sources</a>” and “<a href="#">Liberty Models of Hard Macros</a>”.</p>	-pa_enable
libertypamodelopt	<p>Enables corruption of the following, based on Liberty corruption semantics:</p> <ul style="list-style-type: none"> <li>• outputs of combinational cells</li> <li>• outputs and inputs of sequential cells</li> </ul> <p>This provides a more optimized Liberty-based corruption.</p>	-pa_disable
libretcell	<p>Enables support for the corruption of boundary ports of a retention cell using Liberty information.</p> <p>When you specify:  <code>-pa_enable=libretcell+libertypamodel</code>  then input ports are corrupted according to the related_power_pin and related_ground_pin attributes of your Liberty library and the output ports are corrupted according to the power_down_function Liberty attribute.</p>	-pa_enable
limitassertports	Truncates the dynamic check messages to a threshold value of 16. The complete list of messages are reported in <i>report.pa-asserts.-tags.txt</i> file.	-pa_enable
logicconnectivity	Enables support of UPF-created logic ports in the <a href="#">set_isolation</a> , <a href="#">set_level_shifter</a> , <a href="#">set_repeater</a> and, <a href="#">set_port_attributes</a> UPF commands. It also enables static/dynamic analysis.	-pa_disable

**Table A-3. Power Aware Actions for `vopt -pa_enable` and `-pa_disable` (cont.)**

<b>Value</b>	<b>Action</b>	<b>Default</b>
logicportiso	Enables support of UPF-created logic ports in the <a href="#">set_isolation</a> UPF command.	<code>-pa_disable</code>
lowerboundary	Performs isolation on the ports present in lower boundary of power domain.	<code>-pa_enable</code>
lscellcrpt	<p>Disables Liberty-based corruption of level-shifter cells.</p> <p>If the Liberty model for a level-shifter cell is not present, then driver-based corruption semantics is applied using <code>input_supply_set</code>, <code>output_supply_set</code>, and <code>internal_supply_set</code> of the specified UPF strategy.</p> <p>Corruption semantics of a level-shifter cell are disabled if Liberty model is not present and all three supplies mentioned above are missing from UPF strategy.</p> <p>The simulator does not do any default automatic connections for level-shifter cells. User has to make explicit connections for these cells.</p>	<code>-pa_enable</code>
modsrn	Disables support of the <a href="#">set_related_supply_net</a> command. The command is not a part of the IEEE Std 1801.	<code>-pa_enable</code>
msglimit	<p>Limits logging of messages to a default limit of 50 in the <code>vopt</code> log file.</p> <p>However, the complete set of messages is available for debugging in the new report file <code>&lt;pa_reportdir&gt;/msg.vopt.log</code>.</p>	<code>-pa_enable</code>
orderbysupply	Order isolation and level-shifter cells based on their supplies.	<code>-pa_enable</code>
pdhieritestplan	<p>Enables the hierarchical view of the power domains in the Power Aware test plan.</p> <p>See “<a href="#">Analyzing Coverage Using the Power Aware Test Plan</a>”.</p> <p> <b>Note:</b> To use the <code>pdhieritestplan</code> argument, enable generation of the test plan using the <code>autotestplan</code> argument.</p>	<code>-pa_disable</code>

**Table A-3. Power Aware Actions for vopt -pa\_enable and -pa\_disable (cont.)**

<b>Value</b>	<b>Action</b>	<b>Default</b>
powerunconnets	Enables treating the unconnected pins for a Liberty macromodel (such as backup_power, bias power or ground pins) as power-on to prevent these unconnected pins from causing undesired corruption semantics.	-pa_disable
powerstatesonports	Enables you to specify states on ports/net via <a href="#">add_power_state</a> command in PST.	-pa_disable
pstcomp	Disables cross-PST analysis. See “ <a href="#">Cross-PST Analysis</a> ”.	-pa_enable
reevalinitial	Applies initial re-evaluation globally in the complete design.	-pa_disable
restoreatcoa	Restores the level sensitive retention flip-flop when there is a transition from CORRUPT to CORRUPT_ON_ACTIVITY simstate. This is applicable only for single control balloon-latch retention configuration.	-pa_disable
squarebracketsindex	Enables use of square brackets ([ ]) in a UPF file without an escape character.	-pa_disable
statetransition	Enables use of <a href="#">describe_state_transition</a> on PST/Port.  For transitions marked illegal, the Questa SIM simulator displays an assertion message. These transitions are excluded from PST/Port transition coverage, which affects coverage results.	-pa_disable
supplylogicexpr	Corrupts power states if supply and logic expressions are different.	-pa_disable
swcellcrpt	Disables Liberty-based corruption of switch cells.  If the Liberty model for a switch cell is not present, then driver-based corruption semantics are applied using supply_set of the switch as specified in the UPF strategy.  Because the simulator does not insert default automatic connections for switch cells, you need to make explicit connections for these cells.	-pa_enable

**Table A-3. Power Aware Actions for `vopt -pa_enable` and `-pa_disable` (cont.)**

<b>Value</b>	<b>Action</b>	<b>Default</b>
syntaxchecks	Checks the syntax of the UPF file that you specify with the <code>vopt -pa_upf</code> argument, before loading the design. If you specify the top-level design unit name with the <code>vopt</code> command, then the simulator checks the syntax of the UPF file, and if no errors are found, it elaborates the design.	<code>-pa_enable</code>
trailingedgerestore	Enables the restore event at the trailing edge of a level sensitive restore event of a balloon-style level sensitive retention register when the <code>-restore_condition</code> evaluates to true.  By default, the restore event occurs at the leading edge.	<code>-pa_disable</code>
udpnoret	Enables the retention of sequential UDPs in a Gate-Level or mixed RTL/Gate-Level Power Aware simulation. This does not impact the default behavior of UDP corruption.  See “ <a href="#">Automatic Corruption and Retention of UDPs</a> ”.	<code>-pa_disable</code>
undeterminedstate	Controls behavior related to power aware switches or supply nets/ports going into an UNDETERMINED state as defined by UPF 2.0. <ul style="list-style-type: none"> <li>• <code>-pa_enable</code> — Enables UPF 2.0 behavior</li> <li>• <code>-pa_disable</code> — Reverts to UPF 1.0 behavior</li> </ul>	<code>-pa_disable</code>
upf2.1hierarchynavigation	Enables updated functionality for the <code>set_scope</code> UPF command in UPF 2.1.	<code>-pa_disable</code>
verbosereporting	Toggles the addition of a full hierarchical path of the power domain for the report, <code>report.pa.txt</code> .	<code>-pa_disable</code>
vsim_msgs	Enables the display of additional vsim messages that are related to various Power Aware objects, such as isolation, retention, power switch, and PST.	<code>-pa_disable</code>

## Argument to Exclude Design Element

Use the -pa\_excludefile argument with the vopt command to exclude any module, instance or signal within a particular hierarchical path from Power Aware processing.

The -pa\_excludefile argument excludes the design elements listed in the exclude file, <filename>, from Power Aware processing:

```
vopt -pa_excludefile <filename>
```

Each entry in the exclude file must be of the following form. Use the pound sign (#) to begin a comment line in the exclude file.

```
<module_name> [-a] [<hier_path>] [-s | -sr <signal_name>]
```

- <module\_name>— (required) Excludes the module, <module\_name>, from Power Aware processing. To specify any regular expression in <module\_name>, use quotation marks.
- -a — (optional) Enables recursive exclusion of all instances within <module\_name>. If you specify this argument, the simulator excludes Power Aware processing of all instances present within <module\_name>.
- <hier\_path> — (optional) Excludes the instance of <module\_name> present at <hier\_path> from Power Aware processing. To specify any regular expression in <hier\_path>, use quotation marks.

When Power Aware processing excludes an instance of <module\_name>, the Questa SIM simulator displays the following message:

```
** Note: (vopt-9691) Excluding power aware module '<module_name>' in path '<hier_path>'.
```

When you use <hier\_path> with the -a argument, <hier\_path> limits the recursive exclusion to a particular scope.

- -s <signal\_name>— (optional) Excludes <signal\_name> present in <module\_name> from Power Aware processing. To specify any regular expression in <signal\_name>, use quotation marks.
- -sr <signal\_name> — (optional) Excludes <signal\_name> present in <module\_name> and in the generate block of <module\_name> from Power Aware processing. To specify any regular expression in <signal\_name>, use quotation marks.

## Examples

### Exclude Modules

- Excludes all instantiation of the module, bot\_mod, found within the top hierarchy, when you specify the following in the exclude file:

```
bot_mod
```

The Questa SIM simulator displays the following messages:

```
** Note: (vopt-9691) Excluding power aware module 'bot_mod' in path
'/top/t1'.

** Note: (vopt-9691) Excluding power aware module 'bot_mod' in path
'/top/t1/t2'.
```

- Excludes all instantiation of the module, bot\_mod2 and bot\_mod3, found within the top hierarchy, when you specify the following in the exclude file.

```
"bot_mod[2-3]" /top
```

The Questa SIM simulator displays the following messages:

```
** Note: (vopt-9691) Excluding power aware module 'bot_mod2' in path
'/top/inst1'.

** Note: (vopt-9691) Excluding power aware module 'bot_mod3' in path
'/top/inst2'.
```

### Exclude Signals

- Excludes signal, sig, present in all instantiations of the module, bot\_mod, found within the instance /top/t1, when you specify the following in the exclude file:

```
bot_mod /top/t1 -s sig
```

The Questa SIM simulator displays the following messages:

```
** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig' in power
aware module 'bot_mod' in path '/top/t1/sig'.
```

A warning message is reported if there was no match found in exclude file, for signal, sig, in module, bot\_mod, in instance, /top/t1:

```
** Warning: exclude.txt(1): (vopt-9720) No match found in exclude
file, for signal 'sig' in module 'bot_mod' in instance path '/top/
t1'.
```

- Excludes signals, sig2, sig3, sig4, present in all instantiation of the module, bot\_mod, found within the instance, /top/t1, when you specify the following in the exclude file:

```
bot_mod /top/t1 -s "sig[2-4]"
```

The Questa SIM simulator displays the following messages:

```
** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig3' in
power aware module 'bot_mod' in path '/top/t1/sig3'.

** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig4' in
power aware module 'bot_mod' in path '/top/t1/sig4'.

** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig2' in
power aware module 'bot_mod' in path '/top/t1/sig2'.
```

- Excludes signal, sig, present in the scope of all instantiations of the module, bot\_mod, found within the instance /top/t1, when you specify the following in the exclude file:

```
bot_mod /top/t1 -sr sig
```

The Questa SIM simulator displays the following messages:

```
** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig' in power
aware module 'bot_mod' in path '/top/t1/blk/fg_5/sig'.

** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig' in power
aware module 'bot_mod' in path '/top/t1/blk/fg_4/sig'.

** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig' in power
aware module 'bot_mod' in path '/top/t1/blk/fg_3/sig'.

** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig' in power
aware module 'bot_mod' in path '/top/t1/blk/fg_2/sig'.

** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig' in power
aware module 'bot_mod' in path '/top/t1/blk/fg_1/sig'.

** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig' in power
aware module 'bot_mod' in path '/top/t1/blk/sig'.

** Note: exclude.txt(1): (vopt-9013) Excluding signal 'sig' in power
aware module 'bot_mod' in path '/top/t1/sig'.
```

## Command to Control Dynamic Checks

Use the pa msg command for granular control of dynamic checks, and control the display of warning and error messages that occur during these checks. You can either run the pa msg command at the VSIM prompt, or include the command in a Tcl configuration file and invoke the file at the VSIM prompt.

### **Note**

 When you enable granular control of dynamic checks, you also enable the underlying assertions. This means that the warning and error messages, and any resulting coverage collection that occur during dynamic checks are enabled.

## Syntax

```
pa msg [-enable | -disable] [-checkIds <list-of-dyn-chksId>] [<message numbers>]
      [-pa_checks=<spec>] [-all] [-domains <list-of-domain-name>] [-domain <pd_name>]
      [-strategies <list-of-strategy-name>] [-strategy <strategy_name>]
      [-powerswitches <list-of-switch-name>] [-elements <list-of-elements>]
      [-ports <list_of_ports>] [-pst <list-of-pst-names>] [-transitive [<TRUE | FALSE>]]
      [-models <list-of-modules>] [-scope <scope>] [-severity {note | warning | error | fatal}]
      [-outfile <pa-assertion-outfile-name>] [-glitch_window <duration><time_spec>]
      [-stopafter <number>] [-suppressafter <number>]
```

## Arguments

### Dynamic Check Selection

### **Note**

 If you do not specify any dynamic check with the pa msg command, the command selects all dynamic checks, except the ones with the severity NOTE.

- **-checkIds <list-of-dyn-chksId>**

(optional) Specifies a list of one or more dynamic checks that you want to enable, disable, set the severity for, and so on.

The values you specify in <list-of-dyn-chksId> can be either of the following types:

- The argument values of the vopt -pa\_checks command (such as ira, irc, ifc). The argument values are specified in the “Usage Syntax” column of dynamic checks.

For example:

```
pa msg -enable -checkIds {ira irc ifc}
```

- The mnemonics of checks flagged in the error message (such as QPA\_UPF\_INCORRECT\_LS\_CHK, QPA\_UPF\_MISSING\_LS\_CHK). The mnemonics are specified in the “Mnemonics” column of dynamic checks.

For example:

```
pa msg -disable -checkIds {QPA_ISO_DIS_PG QPA_ISO_EN_PSO}
```

- The numerical value of the error messages (such as 8915, 8930). The numerical values are specified in the “Error Message” column of dynamic checks.

For example:

```
pa msg -disable -checkIds {8934 8910}
```

If the value specified in <list-of-dyn-chksId> does not exist, it causes a semantic error.

Refer to the columns: Usage Syntax, Mnemonics, and Error Message in [Table 5-15](#), [Table 5-16](#), [Table 5-17](#), and [Table 5-18](#) for more information.

- <message numbers>

(optional) Specifies a space-separated list of one or more dynamic checks that you want to enable, disable, set the severity for, and so on. Specify the numerical values of the dynamic check messages in <message numbers>. For details on the numerical values, refer to the “Error Message” column in [Table 5-15](#), [Table 5-16](#), [Table 5-17](#), and [Table 5-18](#). If you do not specify any message number, then the arguments of the pa msg command affect all dynamic checks. If the value specified in <message numbers> does not exist, it causes a semantic error.

For example:

```
pa msg <arguments> 8918 8919  
pa msg -enable 8930 8919
```

---

**Note**

 The <message numbers> argument is deprecated. Use the -checkIds argument instead.

---

- -pa\_checks=[<spec>]

(optional) Specifies a list of one or more dynamic checks that you want to enable, disable, set the severity for, and so on. Specify the argument values of the vopt -pa\_checks command (such as ipc, ifc, ira) in <spec>. For details on the values you specify in <spec>, refer to the “Usage Syntax” column in [Table 5-15](#), [Table 5-16](#), [Table 5-17](#), and [Table 5-18](#). To specify more than one dynamic check, use a plus sign (+) to separate multiple values. If you do not specify any dynamic check, then the arguments of the pa msg command affect all dynamic checks.

For example:

```
pa msg <arguments> -pa_checks=irc  
pa msg -enable -pa_checks=irc+upc
```

**Note**

 The -pa\_checks=<spec> argument is deprecated. Use the -checkIds argument instead.

- -all  
(optional) Selects all dynamic checks.

For example:

```
pa msg <arguments> -all
```

**Granular Control of Selected Dynamic Checks****Tip**

-  If you specify more than one value in any argument list, separate the values with a space, and enclose the list in curly braces ({}).

- -domains <list-of-domain-name>

(optional) Restricts the selected dynamic check to the power domains listed in <list-of-domain-name> and its UPF objects (such as isolation, level shifter, retention strategies). This argument does not have any impact on messages related to checks that are not associated with any power domain, such as uml and umi. If you do not use the -domains argument with the pa msg command, then the selected dynamic check is applied to all power domains. If a specified power domain does not exist, it causes a warning message, and the argument is ignored.

For example:

```
pa msg -enable -checkIds {iep iepcoa} -domains
{TOP/PD_mid2 TOP/PD_mid3}
```

- -domain <pd\_name>

(optional) Restricts the selected dynamic check to the power domain listed in <pd\_name> and its UPF objects (such as isolation, level shifter, retention strategies). This argument does not have any impact on messages related to checks that are not associated with any power domain, such as uml and umi. If you do not use the -domain argument with the pa msg command, then the selected dynamic check is applied to all power domains. If a specified power domain does not exist, it causes a warning message, and the argument is ignored.

For example:

```
pa msg -enable -checkIds {iep iepcoa} -domain
{TOP/PD_mid2}
```

**Note**

 The -domain argument is deprecated. Use the -domains argument instead.

- -strategies <list-of-strategy-name>

(optional) Restricts the selected dynamic check to the strategies listed in <list-of-strategy-name>. This argument does not have any impact on checks that are not associated with any strategy, such as p, t, uml, umi, and npu. If you do not use the -strategies argument with the pa msg command, then the selected dynamic check is applied to all strategies. If a specified strategy does not exist, it causes a warning message, and the argument is ignored.

For example:

```
pa msg -checkIds {irc it} -disable -strategies
{TOP/PD_mid1.iso_PD_mid1}
```

If you use the -strategies argument with the -domain argument, then dynamic checks in domain.strategy are selected.

For example, the following pa msg command enables all dynamic isolation checks that belong to the strategy, iso\_mid1, of the power domain, <scope>/PD\_mid1:

```
pa msg -enable -checkIds {d} -domain PD_mid1 -strategy iso_mid1
```

- -strategy <strategy\_name>

(optional) Restricts the selected dynamic check to the strategy listed in <strategy\_name>. This argument does not have any impact on checks that are not associated with any strategy, such as p, t, uml, umi, and npu. If you do not use the -strategy argument with the pa msg command, then the selected dynamic check is applied to all strategies. If a specified strategy does not exist, it causes a warning message, and the argument is ignored.

For example:

```
pa msg -checkIds {irc} -enable -strategy {TOP/PD_mid1.iso_PD_mid1}
```

#### **Note**



The -strategy argument is deprecated. Use the -strategies argument instead.

---

- -powerswitches <list-of-switch-name>

(optional) Restricts the selected dynamic check to the power switches listed in <list-of-switch-name>. If you do not use the -powerswitches argument with the pa msg command, then the selected dynamic check is applied to all power switches. If a specified power switch does not exist, it causes a warning message, and the argument is ignored.

For example:

```
pa msg -enable -checkIds {ugc} -powerswitches {pd_sw_mid2}
-domains PD_top
```

- -elements <list-of-elements>

(optional) Restricts the selected dynamic checks to the design elements (such as instance, port, net or signal names) listed in <list-of-elements>. If you do not use the -elements argument with the pa msg command, then the selected dynamic check is applied to all

elements. If a specified element does not exist, it causes a warning message, and the argument is ignored.

The -elements argument is not applicable to all dynamic checks. You receive a NOTE message when the -elements argument is not applicable to a dynamic check, and the argument is ignored.

For example, the command:

```
pa msg -enable -checkIds {icp} -domain PD1 -strategies iso_mid1  
-scope /tb/TOP/mid1 -elements {P1 P2}
```

issues the message:

```
** Note: (vsim-4087) Option 'pa msg' is ignored for message number  
8918, as option '-ports/-elements' is not applicable for this  
message number.
```

- **-ports <list\_of\_ports>**

(optional) Restricts the selected dynamic check to the ports listed in <list\_of\_ports>. If you do not use the -ports argument with the pa msg command, then the selected dynamic check is applied to all ports. If a specified port does not exist, it causes a warning message, and the argument is ignored.

For example, the following pa msg command enables isolation clamp value check for ports /tb/TOP/mid1/P1 and /tb/TOP/mid1/P2, which belong to strategy, iso\_mid1, of power domain PD1:

```
pa msg -enable -checkIds {icp} -domain PD1 -strategies iso_mid1  
-scope /tb/TOP/mid1 -ports {P1 P2}
```

The -ports argument is not applicable to all dynamic checks. You receive a NOTE message when the -ports argument is not applicable to a dynamic check, and the argument is ignored.

For example, the command:

```
pa msg -checkIds {iep} -domain PD1 -strategy iso_mid1 -scope /tb/  
TOP/mid1 -ports {P1 P2}
```

issues the message:

```
** Note: (vsim-4087) Option 'pa msg' is ignored for message number  
8918, as option '-ports/-elements' is not applicable for this  
message number.
```

---

### **Note**

---

 The -ports argument is deprecated. Use the -elements argument instead.

---

- **-pst <list-of-pst-names>**

(optional) Restricts the selected dynamic check to the power state tables (PSTs) listed in <list-of-pst-names>. If you do not use the -pst argument with the pa msg command, then the selected dynamic check is applied to all PSTs. If a specified PST does not exist, it causes a warning message, and the argument is ignored.

For example:

```
pa msg -disable -checkIDs {pis} -pst {PowerStateTable_3}
```

- **-transitive [<TRUE | FALSE>]**

(optional) Determines the top-level hierarchy at which the dynamic check is performed. The argument is used in a context where instance hierarchies are specified with the **-elements** argument. The **-transitive** argument has no meaning if you specify it without the **-elements** argument.

- **TRUE** (default) — Causes the **pa msg** command to apply to all hierarchies on and below the specified instance path.
- **FALSE** — Causes the **pa msg** command to apply only to the specified design hierarchies that have instance names specified by the **-elements** argument.

For example:

```
pa msg -enable -checkIDs {d} -elements {mid1/out2_bot}
-transitive TRUE
```

- **-models <list-of-modules>**

(optional) Restricts the selected dynamic check to a list of Verilog modules or VHDL entities specified in **<list-of-modules>**. If you do not use the **-models** argument with the **pa msg** command, then the selected dynamic check is applied to all models. If a specified model does not exist, it causes a warning message, and the argument is ignored.

For example:

```
pa msg -enable -checkIDs {QPA_ISO_PORT_TOGGLE}
-models {mid.arch1 mid_1}
```

- **-scope <scope>**

#### **Note**

 The **-scope** argument must be accompanied by either the **-domains**, **-elements**, **-models**, **-ports**, or **-strategies** argument.

---

(optional) Restricts the selected dynamic check to the scope specified by **<scope>**. The **pa msg** command applies the selected dynamic check to the specified domains/strategies/ports/elements/models, and their UPF objects (such as isolation, level shifter, retention strategies).

By default, the scope of the **pa msg** command is the design from which **vsim** is invoked. For example, if **tb** is the design from which **vsim** is invoked, then the following command enables all dynamic checks in the **/tb** scope:

```
pa msg -enable -all
```

Use of the -scope argument changes the scope of the pa msg command to <scope>. For example, the following command shows you how to enable all dynamic checks in the power domain /tb/dut/mid1/PD1 and /tb/dut/mid1/PD2:

```
pa msg -enable -all -scope /tb/dut/mid1 -domains {PD1 PD2}
```

### **Actions on Selected Dynamic checks**

#### **Tip**

 Select the dynamic checks using the arguments specified in “Dynamic Check Selection”, and “Granular Control of Selected Dynamic Checks”.

- -enable | -disable

(optional) Enables or disables the selected dynamic checks. If you do not use either the -enable or the -disable argument with a pa msg command, then -enable is applied by default.

#### **Caution**



Do not use the -enable and -disable arguments on the same check in the pa msg command. This causes a syntax error.

- -enable

Enables the selected dynamic checks.

For example:

```
pa msg -enable -checkIds {iepcoa idpcoa}
```

- -disable

Disables the selected dynamic checks.

For example:

```
pa msg -disable -checkIds {ira}
```

- -severity {note | warning | error | fatal}

(optional) Sets the severity of the messages of the selected dynamic check. If you do not select any dynamic check, then the -severity argument applies to all dynamic check messages.

For example:

```
pa msg -checkIds {8930 8918} -severity note
```

- -outfile <pa-assertion-outfile-name>

(optional) Redirects the messages resulting from the selected dynamic check to a specified text file. If you do not select any dynamic check, then this argument applies to all dynamic check messages. By default, these messages are displayed in the transcript window.

For example:

```
pa msg -checkIds {8930} -outfile checks_transcript_8930
```

- **-glitch\_window <duration><time\_spec>**

(optional) Specifies the maximum allowed time window of a glitch on the control lines. If the duration of any spike is greater than the maximum allowed time window of a glitch, then the spike is treated as a glitch, or else the spike is ignored. The **-glitch\_window** argument is applicable to glitch detection checks (**-pa\_checks=ugc**) only. Refer to “Glitch detection” check in [Table 5-18](#). If you specify the **-glitch\_window** argument with any other checks, then this argument is ignored.

<duration> — Specifies the maximum allowed time window of a glitch. You can specify any positive real number or zero (0). The default value is zero.

<time\_spec> — Specifies the time unit of <duration>. It can be fs, ps, ns, us, or ms. The default value is the time unit of your simulation.

For example:

```
pa msg -checkIds {ugc} -glitch_window 3ns  
pa msg -checkIds {ugc} -glitch_window 2
```

---

**Note**

 There is no space between <duration> and <time\_spec>.

---

- **-stopafter <number>**

(optional) Stops the simulation after a selected dynamic check message(s) is displayed for <number> times. If you do not select any dynamic check, then the simulation stops when any dynamic check message is displayed for <number> times.

For example:

```
pa msg -checkIds {8930 8918} -stopafter 10
```

- **-suppressafter <number>**

(optional) Limits the number of message(s), of a selected dynamic check, to <number>, where the default value is 15. Once the limit is reached, the selected dynamic check messages are suppressed. If you do not select any dynamic check, then the Questa SIM simulator limits the number of messages of all dynamic checks to <number>.

For example:

```
pa msg -checkIds {8930 8918} -suppressafter 20
```

---

**Note**

 All arguments may not be applicable to all dynamic checks. If an argument is not applicable to a dynamic check, then the Questa SIM simulator ignores that argument.

---

## Description

By default, all dynamic checks are disabled at time zero, resulting in no messages at this time. You can change this behavior with the pa msg command before running your simulation. For example, if you want to enable power domain status check (-pa\_checks=p) at time zero, do the following:

```
vsim > pa msg -enable -checkIDs {p}
vsim > run 1ns
```

You can use the pa msg command any time during simulation. It affects the simulation after the time stamp of its usage until any new pa msg command overrides the previous one. In the following example, the isolation functionality check (-pa\_checks=ifc) is disabled after 100ns and is re-enabled at 150ns. Therefore, in the time window from 100ns to 150ns, this check is disabled.

```
vsim > run 100ns
vsim > pa msg -checkIDs {ifc} -disable
vsim > run 50ns
vsim > pa msg -checkIDs {ifc} -enable
```

## Related Topics

[Precedence Order of Arguments](#)

[Pathname Convention in Arguments](#)

# Arguments to Control Power Aware Message Severity

Use the following vopt arguments to suppress or control the severity of messages that occur while running vopt.

- **-suppress <msg\_num>** — Suppresses a particular message by its ID number (msg\_num).  
Messages are not displayed and vopt processing continues.
- **-warning <msg\_num>** — Changes the severity of a particular message to Warning.  
Messages are displayed and vopt processing continues.
- **-error <msg\_num>** — Changes the severity of a particular message to Error.  
Messages are displayed and vopt processing stops.
- **-note <msg\_num>** — Changes the severity of a particular message to Note.  
Messages are displayed and vopt processing continues.
- **-fatal <msg\_num>** — Changes the severity of a particular message to Fatal.

Messages are displayed and vopt processing stops.

Applying different severity levels enables you to prevent stoppage of vopt operation or inconsistent behavior in processing and resolve phases when the message is encountered. An example of this is when vopt stops for errors in the processing phase, while continuing when the same errors occur in the resolve phase.

---

**Tip**

 When you suppress or lower the severity of an error message, you may not see the desired result, since the corresponding UPF command behavior gets bypassed or ignored.

---

## Examples

- Use vopt to suppress or change the severity of a single message:

```
vopt -pa_upf top.upf -o t tb -suppress <msg_num>
```

- Use vopt to suppress or change the severity of multiple messages:

```
vopt -pa_upf top.upf -o t tb -A <msg_num1>, <msg_num2>, <msg_num3>
```

## Supported UPF Extensions

Use the -pa\_upfextensions argument to the vopt command to define and apply various UPF behaviors that are not supported by the current standard (UPF 3.0).

Table A-4 lists the values for -pa\_upfextensions argument that you can specify to override the supported UPF behavior.

## Usage

```
vopt -pa_upfextensions=[<extension_name>]
```

## Description

- To specify more than one value for this argument, use the + operator between values (there is no order dependency when specifying multiple values). For example:

```
vopt -pa_upfextensions=relatedsnet+genblk+v
```

- To enable all values, specify the following:

```
vopt -pa_upfextensions=all
```

- To enable a limited number of values (see “default” in Table A-4), specify either of the following:

```
vopt -pa_upfextensions=default  
vopt -pa_upfextensions
```

## Arguments

**Table A-4. Argument Values for vopt -pa\_upfextensions**

Value	Description
all	Enables (specifies) all values of the -pa_upfextensions argument.
ackport	<p>Enables the following behavior:</p> <p>If you specify a boolean_function for -ack_port, the result of the boolean_function is driven on -ack_port's port_name delay time units after a control port transition.</p> <p>Without this value, the default behavior is as defined in UPF 2.1 semantics. An acknowledge value is driven onto the specified port_name delay time units after the switch output transitions to a FULL_ON state and the inverse acknowledge value is driven onto the specified port_name delay time units after the switch output transitions to an OFF state.</p>
altgenname	<p>Supports the synthesis style hierarchical paths for generate blocks. Power Aware simulation recognizes a hierarchical path with an escaped generate scope of the form that a synthesis tool generates, and maps such a name to a hierarchical name of conventional form.</p> <p>For example, each of the following styles:</p> <pre>{&lt; prefix &gt;/}gen_label[index].name2 {/&lt; suffix &gt;} /* 'gen_label[index].name2' is the new name of the instance 'name2' within scope gen_label[index].*/ {&lt; prefix &gt;/}\gen_label[index].name2 {/&lt; suffix &gt;} /* The new name could also be double-escaped. */ {&lt; prefix &gt;.}gen_label[index].name2 {/&lt; suffix &gt;} /* Use '..' as a path separator for generate scopes */</pre> <p>would map to:</p> <pre>{&lt; prefix &gt;/}gen_label[index]/name2 {/&lt; suffix &gt;}</pre>
autovctgnd	<p>Enables automatic recognition and application of the correct VCT, based on the set_domain_supply_net and supply_set functions.</p> <p>The Questa SIM simulator applies the ground-specific VCT on the supply net connection if that net is used as a primary ground net, an isolation ground net, retention ground net, a related ground supply used in set_related_supply_net or is tied to a macro PG pin of type primary ground.</p> <p>This is the default behavior of the simulator.</p>
bindchecker	Enables use of -parameters in the bind_checker UPF command.
case	Enables you to specify predefined values (such as TRUE, FALSE, high, low, posedge, and latch) in case-insensitive form.

**Table A-4. Argument Values for vopt -pa\_upfextensions (cont.)**

Value	Description
connectpgports	Enables the creation of a port when <a href="#">connect_supply_net</a> identifies a port which is missing in HDL but is specified as a pg_pin in a Liberty model.  This is equivalent to using vopt -pa_connectpgpin=c
default	Enables (specifies) only the following values of the -pa_upfextensions argument: <ul style="list-style-type: none"><li>• case</li><li>• genblk</li><li>• ignorepgports</li><li>• locationinstance</li><li>• mapcell</li><li>• nonameclash</li><li>• nonlrmstatename</li><li>• relatedsnet</li><li>• s</li><li>• squarebracketasindex</li><li>• v</li></ul> NOTE: Specifying vopt -pa_upfextensions with no values has the same effect.
dirbasedrelsupp	Enables you to specify -driver_supply and -reciever_supply on the <a href="#">set_port_attributes</a> command. The Questa SIM simulator extracts the input and the output ports from the port list specified in the command and applies driver and receiver supply, respectively.

**Table A-4. Argument Values for vopt -pa\_upfextensions (cont.)**

Value	Description
flathiername	<p>Enables you to specify synthesis-style hierarchical names for instances in your UPF file. Specifically, this value allows for two types of references in UPF:</p> <ul style="list-style-type: none"> <li>• Enables you to refer to flattened instance names of the netlist using a hierarchical name in your UPF file.</li> </ul> <pre>set_isolation pmu_0 -elements {top_i/shell_i/ tsmu_clk_sel_o } //UPF snippet  flat_mod \top_i/shell_i/ tsmu_clk_sel_o(.o(out1),.in(in1)); //Flattened netlist hierarchy</pre> <p>For this example, the hierarchical path “top_i/shell_i/tsmu_clk_sel_o” in the UPF file does refer to an escaped HDL instance “\top_i/shell_i/tsmu_clk_sel_o”.</p> <ul style="list-style-type: none"> <li>• Enables you to refer to the escaped instance name of the netlist by its un-escaped name in your UPF file.</li> </ul> <pre>set_retention R_RULE_2 -domain PD_TOP -elements { top_i/ bus_A2_reg[1] } ... set_retention R_RULE_1 -domain PD -elements { bus_A2_reg[0] } //UPF snippet  core_mem_mod \bus_A2_reg[0] ( .ret(ret_en), .d(n7),.sin(n4), .shift(n4), .clk(clk), .q(bus_A2[1]) ); core_mem_mod \bus_A2_reg[1] ( .ret(ret_en), .d(n7),.sin(n4), .shift(n4), .clk(clk), .q(bus_A2[1]) ); //netlist snippet</pre> <p>For this example, the unescaped instance name “bus_A2_reg[0]” in UPF is a valid reference to an escaped HDL name “bus_A2_reg[0]”.</p>
genblk	Supports the use of a generate block with the -elements option of the <a href="#">set_scope</a> , <a href="#">find_objects</a> , and <a href="#">create_power_domain</a> commands.
genericretcond	<p>Enables UPF_GENERIC_CLOCK and UPF_GENERIC_ASYNC_LOAD in save, restore, or retention conditions.</p> <p>This is the default behavior of the simulator.</p>

**Table A-4. Argument Values for vopt -pa\_upfextensions (cont.)**

Value	Description
genericclk	<p>Enables the query_retention and bind_checker commands to use UPF_GENERIC_CLOCK for custom retention checks. For example, the following clock specification:</p> <pre data-bbox="540 418 1144 566">always @(posedge clk or posedge reset)   if(reset)     q &lt;= 1;   else     q &lt;=d;</pre> <p>used with:</p> <pre data-bbox="540 656 1168 713">set_retention -restore_condition{ret &amp;&amp; !UPF_GENERIC_CLOCK}</pre> <p>results in clk replacing UPF_GENERIC_CLOCK in the always block.</p>
ignorepgports	<p>Bypasses connection of a supply net to a port using the <a href="#">connect_supply_net</a> command when the port is missing in the verification model but is a power or ground (PG) pin in the Liberty model.</p> <p>In this case, the connect_supply_net command to these ports is ignored.</p> <p>This is equivalent to using vopt -pa_connectpgpin=i</p>
ignorepgportsaon	<p>Bypasses connection of a supply net to a port using the <a href="#">connect_supply_net</a> command when the port is missing in the verification model but is a power or ground (PG) pin in the Liberty model.</p> <p>In this case, connect_supply_net command to these ports is ignored and the Power Aware simulation semantics of the parent instance of the port are disabled.</p> <p>This is equivalent to using vopt -pa_connectpgpin=a</p>
ignoresupply_expr	<p>Ignores supply_expr when both supply and logic expressions are present in an <a href="#">add_power_state</a> command.</p>
internalconn	<p>Enables connections to internal wires/registers representing supply ports, where the default is that internal connections are not supported. Refer to the section “<a href="#">Extended Power Aware HDL Models</a>” for more information.</p>

**Table A-4. Argument Values for vopt -pa\_upfextensions (cont.)**

<b>Value</b>	<b>Description</b>
liberty	Relaxes syntax and semantic checks on Liberty file contents in order to accept files that do not follow the 2013.3 Liberty specification but are accepted by other products, specifically: <ul style="list-style-type: none"> <li>• Allows five arguments to the edges and shifts groups in the generated_clock group.</li> <li>• Allows the vector_id in the sensitization group to be floating point instead of integer.</li> </ul>
locationinstance	Enables you to specify an instance name as a value to the -location option of the <a href="#">set_isolation</a> and <a href="#">set_level_shifter</a> UPF commands. For example: <pre>set_isolation -location &lt;instance_name&gt; set_level_shifter -location &lt;instance_name&gt;</pre>
mapcell	Supports the use of -lib_cells specified in <a href="#">map_isolation_cell</a> and <a href="#">map_level_shifter_cell</a> UPF commands in <libraryname/cellname> format. For example: <pre>map_power_switch PD_ONE -domain PD_SUP -lib_cells{LIB/NODE}</pre>
matchonlypwrgrnd	Matches only the power and ground supply set functions for the supply set equivalence analysis. By default, the Questa SIM simulator matches all specified supply set functions.
noextsupplycheck	Instructs the Questa SIM simulator to ignore 1) the receiver supply attribute on a hierarchical output port and 2) the driver supply attribute on a hierarchical input port.
nonameclash	Ignores the name clash error that occurs for ports of the <a href="#">set_power_switch</a> command that are specified for the -input_supply_port or -output_supply_port options and that already exist in RTL.
nonlrmstatenames	Enables non-standard UPF names in state name of <a href="#">add_port_state</a> command. For example: <pre>add_port_state VN1 -state {1p1 1.0}</pre>
querycmds	Dumps out the full hierarchical paths of UPF objects in the return value. The action also returns the primary_power_net and primary_ground_net for the query_power_domain command.
relatedsnet	Supports the behavior for the Tcl command <a href="#">set_related_supply_net</a> . This is the default behavior of the simulator.
repeater	Enables you to insert repeater buffers at the self location for output ports. By default, the Questa SIM simulator inserts repeater buffers at the parent location for output ports.

**Table A-4. Argument Values for vopt -pa\_upfextensions (cont.)**

Value	Description
s	Enables relative paths in <code>set_scope</code> command. For example: <code>set_scope .../.../</code>
squarebracketasindex	Enables use of un-escaped square brackets ([]) as indices, in your UPF file.
statetransition	Enables use of <code>describe_state_transition</code> on PST/Port. For transitions marked illegal, the Questa SIM simulator displays an assertion message. These transitions are excluded from PST/Port transition coverage, which affects coverage results.
swctrl	For a UPF power switch, if the boolean expression of an on state or partial-on state specification evaluates to X or Z, and the corresponding input supply port's value has a state of FULL_ON or PARTIAL_ON then switch output is set to UNDETERMINED state. If the control input to a power switch is in an unknown state, and the supply input to the switch is not OFF, then the output of the switch is potentially unknown. This value enables Power Aware simulation to more accurately represent the output of the power switch in this situation.
transferstates	Enables transferring state or voltage, added to a supply set, to its associated supply set handle.
v	Enables automatic insertion of VCT for pins detected as power and ground pins.

**Table A-4. Argument Values for vopt -pa\_upfextensions (cont.)**

<b>Value</b>	<b>Description</b>
wildcard	Supports the use of the wildcard character (*) to the following UPF commands: <ul style="list-style-type: none"><li>• add_domain_elements -elements</li><li>• bind_checker -elements</li><li>• connect_logic_net -ports</li><li>• connect_supply_net -ports</li><li>• connect_supply_set -elements</li><li>• create_power_domain -elements -exclude_elements</li><li>• map_isolation_cell -elements</li><li>• map_level_shifter_cell -elements</li><li>• map_retention_cell -elements</li><li>• set_design_attribute -elements</li><li>• set_isolation -elements</li><li>• set_level_shifter -elements</li><li>• set_retention -elements</li><li>• set_pin_related_supply -pins</li><li>• set_related_supply_net -objects</li><li>• set_port_attribute -elements -ports</li></ul>

## Tcl Commands

---

Power Aware simulation supports Tcl commands that you specify in either a UPF or Tcl file. Use the vopt -pa\_upf or vopt -pa\_tclfile argument to include the UPF or Tcl file in the vopt run.

<b>describe_state_cross_coverage .....</b>	<b>318</b>
<b>getenv .....</b>	<b>319</b>
<b>pa_checks .....</b>	<b>320</b>
<b>save_checks_config .....</b>	<b>325</b>
<b>set_corruption_extent .....</b>	<b>326</b>
<b>set_feedthrough_object .....</b>	<b>327</b>
<b>set_pgpin_vct .....</b>	<b>328</b>
<b>set_related_supply_net .....</b>	<b>329</b>

## describe\_state\_cross\_coverage

Provides results of cross coverage for a specified list of power domains in a design.

### Syntax

```
describe_state_cross_coverage -domains <domain_list> [-depth=<integer>]
```

### Arguments

- **-domains <domain\_list>**  
Creates a list of dependent power domains for which the Questa SIM simulator has to provide cross coverage. The number of domains you list must match the number you specify for **-depth**.
- **-depth=<integer>**  
(optional) Specifies the depth of subordinate (child) power domains whose cross coverage needs to be considered. This number must match the number of power domains you list for **-domains**. The default value is 1.

### Description

For a depth of 1 (default), the Questa SIM simulator determines a list of dependent power domains of the listed domain. These dependent power domains, together with the specified domain, form a domain group. The Questa SIM simulator provides cross coverage results for this particular group of power domains.

Specify the **describe\_state\_cross\_coverage** command in either a UPF or Tcl file, and include the file in the vopt run using the **-pa\_upffile** or **-pa\_tclfile** argument.

### Examples

```
describe_state_cross_coverage -domains PDSYS1 PDSYS2 -depth=2
```

# getenv

Returns the value of any environment variable.

## Syntax

`getenv {<variable>}`

## Arguments

- **variable**

The name of an environment variable. Enclose the variable in curly braces ({}).

## Description

Specify the getenv command in either a UPF or a Tcl file, and include the file in the vopt run using the -pa\_upffile or -pa\_tclfile argument

## Examples

Return the value of the environment variable \$myvar.

```
set abc [getenv {myvar}]
echo $abc
```

## pa\_checks

Provides granular control of static RTL and dynamic checks.

### Syntax

```
pa_checks [-enable | -disable] -checkIds <list-of-chksId> [-domains <list-of-domain-name>]
           [-strategies <list-of-strategy-name>] [-powerswitches <list-of-switch-name>]
           [-elements <list-of-elements>] [-transitive [<TRUE | FALSE>]]
           [-sources <list-of-source-pd-ss>] [-sinks <list-of-sink-pd-ss>]
           [-models <list-of-modules>] [-pst <list-of-pst-names>]
           [-chkctrl_expr {<ctrl-signal-expression>}]
```

### Arguments

- **-enable | -disable**

(optional) Enables or disables the checks selected by the -checkIds argument in a pa\_checks command. If you do not use either the -enable or the -disable argument with a pa\_checks command, then -enable is applied by default.

#### **Caution**

 Do not use the -enable and -disable arguments on the same check in the pa\_checks command. This causes a syntax error.

---

- **-enable**

Enables the selected dynamic checks.

For example:

```
pa_checks -enable -checkIds {smi sri}
```

- **-disable**

Disables the selected dynamic checks.

For example:

```
pa_checks -disable -checkIds {sml}
```

- **-checkIds <list-of-chksId>**

(required) Specifies a list of one or more static RTL and dynamic checks that you want to enable or disable.

The values you specify in <list-of-chksId> can be either of the following types:

- The argument values of the vopt -pa\_checks command (such as smi, sml, icp, t). The argument values are specified in the “Usage Syntax” column of static RTL and dynamic checks.

For example:

```
pa_checks -enable -checkIds {smi sml ira}
```

- The mnemonics of checks flagged in the error message (such as ISO\_NOT\_REQUIRED, LS\_MISSING). The mnemonics are specified in the “Mnemonics” column of dynamic checks.

For example:

```
pa_checks -disable -checkIds {ISO_NOT_REQUIRED LS_MISSING}
```

- The numerical value of the error messages (such as 9750, 9693). The numerical values are specified in the “Error Message” column of dynamic checks.

For example:

```
pa_checks -disable -checkIds {9750 9693}
```

- *all* to enable or disable all checks.

For example:

```
pa_checks -enable -checkIds {all}
```

For details on the values you specify to <list-of-chkIds>, refer to the columns: Usage Syntax, Mnemonics, and Error Message in [Table 5-3](#), [Table 5-4](#), [Table 5-5](#), [Table 5-6](#), [Table 5-15](#), [Table 5-16](#), [Table 5-17](#), and [Table 5-18](#).

- **-domains <list-of-domain-name>**

(optional) Restricts checks to the power domains listed in <list-of-domain-name> and their UPF objects (such as isolation, level shifter, retention strategies). If you do not use the -domains argument with the pa\_checks command, then checks are applied to all power domains. If a specified power domain does not exist, it causes a semantic error and a warning message; the argument is ignored.

For example:

```
pa_checks -enable -checkIds {smi iepcoa} -domains
{TOP/PD_mid2 TOP/PD_mid3}
```

- **-strategies <list-of-strategy-name>**

(optional) Restricts checks to the strategies listed in <list-of-strategy-name>. If you do not use the -strategies argument with the pa\_checks command, then checks are applied to all strategies. If a specified strategy does not exist, it causes a semantic error and a warning message; the argument is ignored.

For example:

```
pa_checks -checkIds {irc it} -disable -strategies
{TOP/PD_mid1.iso_PD_mid1}
```

If you use the -strategies argument with the -domain argument, then checks in domain.strategy are selected.

For example, the following pa\_checks command enables all static missing isolation checks that belong to the strategy, iso\_mid1, of the power domain, <scope>/PD\_mid1:

```
pa_checks -enable -checkIDs {smi} -domain PD_mid1 -strategy iso_mid1
```

If you use the -strategies argument with a check that does not belong to any UPF strategy, it causes a semantic error and a warning message; the argument is ignored.

For example, the following checks do not have any UPF strategy:

- Power domain status check, -pa\_checks=p
- Power domain toggle check, -pa\_checks=t

- **-powerswitches <list-of-switch-name>**

(optional) Restricts checks to the power switches listed in <list-of-switch-name>. If you do not use the -powerswitches argument with the pa\_checks command, then checks are applied to all power switches. If a specified power switch does not exist, it causes a semantic error and a warning message; the argument is ignored.

For example:

```
pa_checks -enable -checkIDs {ugc} -powerswitches {pd_sw_mid2}  
-domains PD_top
```

- **-elements <list-of-elements>**

(optional) Restricts checks to the design elements (such as instance, port, net or signal names) listed in <list-of-elements>. If you do not use the -elements argument with the pa\_checks command, then checks are applied to all elements. If a specified element does not exist, it causes a semantic error and a warning message; the argument is ignored.

For example:

```
pa_checks -enable -checkIDs {QPA_ISO_PORT_TOGGLE} -elements  
{mid1/out2_bot}
```

- **-transitive [<TRUE | FALSE>]**

(optional) Determines the top-level hierarchy at which the dynamic checks are performed. The argument is used in a context where instance hierarchies are specified with the -elements argument. The -transitive argument has no meaning if you specify it without the -elements argument.

- TRUE (default) — Causes the pa\_checks command to apply to all hierarchies on and below the specified instance path.
- FALSE — Causes the pa\_checks command to apply only to the specified design hierarchies that have instance names specified by the -elements argument.

For example:

```
pa_checks -enable -checkIDs {d} -elements {mid1/out2_bot}
          -transitive TRUE
```

- **-sources <list-of-source-pd-ss>**

(optional) Specifies a list of source power domains or supply sets in <list-of-source-pd-ss>. If a specified source name does not exist, it causes a semantic error and a warning message; the argument is ignored.

For example:

```
pa_checks -disable -checkIDs {smi} -sources {PD_mid1}
```

- **-sinks <list-of-sink-pd-ss>**

(optional) Specifies a list of sink power domains or supply sets in <list-of-sink-pd-ss>. If a specified sink name does not exist, it causes a semantic error and a warning message; the argument is ignored.

For example:

```
pa_checks -disable -checkIDs {smi} -sinks {PD_mid2}
```

- **-models <list-of-modules>**

(optional) Restricts checks to a list of Verilog modules or VHDL entities specified in <list-of-modules>. If you do not use the -models argument with the pa\_checks command, then checks are applied to all models. If a specified model does not exist, it causes a semantic error and a warning message; the argument is ignored.

For example:

```
pa_checks -enable -checkIDs {QPA_ISO_PORT_TOGGLE}
          -models {mid.arch1 mid_1}
```

- **-pst <list-of-pst-names>**

(optional) Restricts checks to the power state tables (PSTs) listed in <list-of-pst-names>. If you do not use the -pst argument with the pa\_checks command, then checks are applied to all PSTs. If a specified PST does not exist, it causes a semantic error and a warning message; the argument is ignored.

For example:

```
pa_checks -disable -checkIDs {pis} -pst {PowerStateTable_3}
```

- **-chkctrl\_expr {<ctrl-signal-expression>}**

(optional) Specifies control signal expressions whose value determines whether to enable or disable the selected dynamic checks.

For example:

```
pa_checks -checkIDs {rpo rop} -chksctrl_expr {/tb/retpwr && /tb/rtc}
```

**Note**

-  All arguments may not be applicable to all Power Aware checks. If an argument is not applicable to a Power Aware check, then the simulator flags an appropriate warning message and discards that argument.
- 

## Description

Specify the pa\_checks command in either a UPF or Tcl file, and include the file in the vopt run using the -pa\_upffile or -pa\_tclfile argument.

## Related Topics

[Precedence Order of Arguments](#)

[Pathname Convention in Arguments](#)

## **save\_checks\_config**

Saves the configuration of static RTL and dynamic checks performed by the Questa SIM simulator, to a specified file.

### **Syntax**

**save\_checks\_config <file\_name>**

### **Arguments**

- **<file\_name>**

Specifies the file name in which the configuration of static RTL and dynamic checks performed by the Questa SIM simulator, is saved.

### **Description**

Specify the save\_checks\_config command in either a UPF or Tcl file, and include the file in the vopt run using the -pa\_upf\_file or -pa\_tcl\_file argument.

### **Examples**

`save_checks_config config_file`

### **Related Topics**

[Controlling Checks With the pa\\_checks Command](#)

## **set\_corruption\_extent**

Changes the corruption extent of the power domains created by the UPF file.

### Syntax

```
set_corruption_extent -domains <domain_name> ... -ce {o | os | osw | sc | scb}
```

### Arguments

- **domain\_name**

The name of any power domain in the current scope. Specify multiple domain names in a space-separated list, enclosed in braces ({}).

- **-ce**

Sets the corruption extent, which takes one of the following values:

o — outputs only.

os — outputs and sequential elements.

osw — outputs and sequential and non-sequential wires.

sc — sequential and combination logic (based on UPF corruption semantics honoring all sequential and combination logic for corruption—excluding any buffers in the path for corruption).

scb — sequential, combination, and buffer logic.

### Description

Specify the set\_corruption\_extent command in either a UPF or Tcl file, and include the file in the vopt run using the -pa\_upffile or -pa\_tclfile argument.

### Examples

Change the corruption semantics of domains P1 and P2 created in the scope tb to outputs only.

```
set_scope tb
set_corruption_extent -domains {P1 P2} -ce o
```

## **set\_feedthrough\_object**

Enables conversion functions to be treated as feedthroughs for UPF-based corruption. The objective is to detect conversion functions in the PA-RTL and treat them as feedthrough paths.

### Syntax

```
set_feedthrough_object -function <function_list> [-package <package_name> ]
```

### Arguments

- **-function <function\_list>**  
A required list of one or more function names (you must specify at least one function name).
- **-package <package\_name>**  
Detects only functions from the specified package, `package_name`. Optional.

### Description

Specify the `set_feedthrough_object` command in either a UPF or Tcl file, and include the file in the vopt run using the `-pa_upffile` (UPF file) or `-pa_tclfile` (Tcl) argument.

In an RTL logic, a function call creates a driver in the design, to which Power Aware attempts to apply the specified power intent. However, functions that are intended only to convert or assign data types should not be considered as part of the power intent—they do not require isolation, retention, or corruption.

You can create a Tcl file to identify such functions in your design so that they are excluded from your power intent. These are referred to as a “feedthrough” functions. In the Tcl file, you use the `set_feedthrough_object` command for each function you want to exclude.

## **set\_pgpin\_vct**

Specifies the VCT to be used for pins of any particular pg\_type attribute. This command is specific to Questa SIM.

### Syntax

```
set_pgpin_vct <pg_type> -vct <vct_name>
```

### Arguments

- <pg\_type>

Identifies the value of a pg\_type attribute, such as primary\_power, primary\_ground, backup\_power, backup\_ground, nwell, pwell, deepnwell, deeppwell, internal\_power, or internal\_ground.

- -vct <vct\_name>

Identifies the VCT.

### Description

Specify the set\_pgpin\_vct command in either a UPF or Tcl file, and include the file in the vopt run using the -pa\_upffile or -pa\_tclfile argument.

### Examples

```
set_pgpin_vct primary_power -vct UPF2VHDL_SL
```

## [set\\_related\\_supply\\_net](#)

Enables associating an instance signal pin or a hierarchical port with specific supply nets. Thus, you can create a supply net based on supply pins to specify your related supplies of cells.

### Syntax

```
set_related_supply_net -object_list <objects> -reset -power <power_net_name>  
-ground <ground_net_name>
```

### Arguments

- **-object\_list <objects>**  
List of ports or pins that is to have a related power or ground supply defined. Pins or ports are referenced relative to the active scope.
- **-power <power\_net\_name>**  
The related supply power net, referenced relative to the active scope. You can specify this switch by itself or in combination with -ground.
- **-ground <ground\_net\_name>**  
The related supply ground net, referenced relative to the active scope. You can specify this switch by itself or in combination with -power.
- **-reset**  
Not currently supported.

### Description

Specify the set\_related\_supply\_net command in either a UPF or Tcl file, and include the file in the vopt run using the -pa\_upf\_file or -pa\_tclfile argument. Also, specify the vopt -pa\_upf\_extensions argument in the vopt command (see “[Supported UPF Extensions](#)”).

Power Aware simulation internally maps this command to the [set\\_port\\_attributes](#) command that you specify in the UPF file, specifically the following options:

```
-related_power_port <power_net_name>  
-related_ground_port <ground_net_name>
```

Note that UPF 2.0 interprets the **related\_power\_net** and **related\_ground\_net** attributes as defining the driver supply set of an output port or the receiver supply set of an input port. The standard also declares that it is an error in the following cases:

- If the actual driving logic is present and its supply is not the same as the driver supply.
- If the actual receiving logic is present and its supply is not the same as the receiver supply.

As a result, use of **set\_related\_supply\_net** for any purpose other than specifying the driver supply set of a macro model output or the receiver supply set of a macro model input may generate errors.

Specifically, Power Aware simulation gives a vopt error message (vopt-9814). Use the -warning argument of vopt to change the severity of this message to a warning so that simulation may continue:

```
vopt -warning 9814
```

---

**Tip**

 Refer to “[Arguments to Control Power Aware Message Severity](#)” for more information on changing the level of message severity.

---

### Default Behavior

The functionality in the section is enabled by default.

For top-level ports (in the absence of boundary information), it is left up to the simulator to decide how to handle the primary input and output ports.

You can define the supply related to the primary input and output ports, and instruct the simulator as to what voltage the input ports can be driven and what voltage the output ports can drive.

If you use the -source, -sink, or -diff\_supply\_only options for setting isolation strategies, the related supply tells the simulator what supply is powering the cell on the other side of the domain boundary, which can then be used to determine whether or not isolation is necessary based upon your constraints. This related supply information is useful for static analysis.

This command relates the specified power and/or ground net to the port or pin specified.

When you use **set\_related\_supply\_net** on a primary port:

- Input port assumes the supply net specified is the driver of the input port.
- Output port assumes the supply net specified is the receiver of the output port.

That is, the command specifies the external supplies. In other words a buffer insertion takes place at the location parent.

- For input port — Buffer --> Input port
- For output port — Output Port --> Buffer

Static analysis honors this information of related supplies.

As buffers are now inserted in design just like isolation, when supply of buffer goes off, then buffer output gets corrupted.

For this case, when isolation is also applied on the port at which srns is defined:

- For input ports — buffer is applied at location parent (followed by isolation cells).
   
Buffer -- ISO -- Input port
- For output ports — buffer is applied at location parent (preceded by isolation cells)
   
Output Port -- ISO -- Buffer

When buffer (set\_related\_supply\_net supply) goes off:

- Input Port — Corruption is seen on the input port.
- Output Port — Corruption is seen on the logic (or port) driven by output port.

#### **Simulator Behavior When Using set\_related\_supply\_net for Multiple Ports**

When you specify the set\_related\_supply\_net Tcl command for multiple ports of the same feed-through path, the simulator behaves in the following ways:

- It inserts repeater buffers in the path that breaks the feed-through path into multiple segments, which changes your results as isolation and level-shifter cells are inserted for these segments based on the source and sink supplies of the segment.
- It performs power aware checks on the isolation and level-shifter requirements for each segment, resulting in the power aware static report containing a Supplies field corresponding to the supply attributes along with domain information, for example:

```
Source power domain : { PD_MID1(Supplies(SS1_PWR,SS1_GNET)) } ->
Sink power domain: { PD_TOP(Supplies(SS2_PWR,SS2_GNET)) } [ Total
count: 1 ]
  1. LS( count: 1 ): Candidate Port: /tb/top_inst/mid1/out1_bot,
  count:1, level shifting strategy : ls1, Domain: PD_MID1,
    Source port{Supplies(SS1_PWR,SS1_GNET)}: /tb/top_inst/mid1/
    bot1/out1_bot, count:1 [ LowConn ] -> Sink port
    {Supplies(SS2_PWR,SS2_GNET)}: /tb/top_inst/mid1/out1_bot, count:1 [ 
    HighConn ]
    Possible reason:'Level shifter of required direction is present
    from (Supplies(SS1_PWR,SS1_GNET)) => (Supplies(SS2_PWR,SS2_GNET))
```

This behavior also applies when using the set\_port\_attributes UPF command with the -repeater\_supply option.

# Voltage Level-Shifting (Multi-Voltage Analysis)

---

You implement voltage level-shifting capability for Power Aware simulation primarily through Unified Power Format (UPF) commands and Power Aware arguments to the Questa SIM `vopt` command.

The supply network state provides information about the possible power states of the network. Power Aware simulation uses that information to detect level shifters wherever a signal crosses from a power domain operating at a voltage level that may be different than the voltage level of another power domain to which it connects (also known as multi-voltage analysis).

<b>Power State Tables.....</b>	<b>332</b>
<b>Level Shifter Specification .....</b>	<b>334</b>

## Power State Tables

Power Aware simulation uses information from a Power State Table (PST) in Power Aware analysis. PSTs are also parsed and dumped to the PST Analysis Report (*report.pst.txt*). It is assumed that the PST is complete; any domains that are not mentioned in PST is not used for analysis.

The traversal does not skip any power switches encountered in the supply network path. The traversal goes only behind direct connectivity of supply ports and supply nets that are created in UPF. It does not go behind a supply net present in the design or the UPF supply net/port that is directly connected to an HDL supply net or port.

### Example A-1. Simple PST Example

```
Pst top_pst, File:../UPF/rtl_top.upf(127).
Header ==>          : VDD_0d99 VDD_0d81 VSS
ON  ../UPF/rtl_top.upf(133): ON      ON      ON
OFF ../UPF/rtl_top.upf(134): ON      ON      ON

List of possible states on:
VDD_0d99 [ source supply port: VDD_0d99, File:../UPF/rtl_top.upf(21) ]
  1. ON: 0.99,1.10,1.21

VDD_0d81 [ source supply port: VDD_0d81, File:../UPF/rtl_top.upf(22) ]
  1. ON: 0.81,0.90,0.99

VSS [ source supply port: VSS, File:../UPF/rtl_top.upf(23) ]
  1. ON: 0.00,0.00,0.00
```

## Power State Conversion

The Questa SIM simulator converts the power states specified by the `add_power_state` command in the UPF file, to the corresponding PST states.

**Table A-5** lists the warning messages displayed in the transcript window during the power state conversion.

**Table A-5. Power State Conversion Warning Messages**

Warning	Description Warning Message
vopt-9882	A mapped state in the generated PST is undetermined.  ** Warning: (vopt-9882) Undetermined state: OFF[A11_S0] on UPF Object: SS:/tb/foo.primary[A11] found. Ignoring it for PST analysis.
vopt-9883	A mapped state in the generated PST is unreachable.  ** Warning: (vopt-9883) Unreachable state: OFF[A7_S0] on UPF Object: SS:/tb/ foo.default_retention[A7] found. Ignoring it for PST analysis.
vopt-9934	A power state cannot be converted to a PST.  ** Warning: ./src/aps_err1/test.upf(19): (vopt-9934) Unable to convert power state: ON[A0_S1] present on SS:/tb/ss_vh[A0] to any PST state. Ignoring it.

The [PST Analysis Report](#) (*report.pst.txt*) file indicates a reason for any failure in the power state to PST state conversion:

```
Mapped PST states: Conversion FAILED
Failed to convert sub-expr:((vdd_vh == vss) ^ vss == `{FULL_ON}))
```

## Cross-PST Analysis

PSTs provide static information about the relationships between different supplies used in power management. This information is used for the determination of operating voltage for level-shifter placements and static checks for isolation and level-shifter cells.

In most development scenarios, designers prefer to split the information into sub-PSTs, typically defined for each IP, relying on verification tools to interpret and analyze the PSTs. This method controls the number of states required, based on the number of supply nets and voltages, which makes a global PST unreasonable to design and manage.

By default, the cross-PST analysis is enabled. The results are reported in the [PST Analysis Report](#) (*report.pst.txt*) and [Static Check Report](#) (*report.static.txt*) report files. You disable the cross-PST analysis with the `-pa_disable=pstcomp` argument to the `vopt` command.

Cross-PST analysis is based on actual domain crossings involved in the design. Power Aware simulation first identifies the power domain crossings or supply set pairs that require analysis and secondly performs the analysis. If there are more than one PSTs involved in a particular domain crossing then the simulator first composes those PSTs to create a Composed PST and then identifies the list of port state on the primary supplies of the domains in question.

For scenarios where any PSTs contain unrelated information and cannot be composed together using common supplies, the simulator picks all possible combinations using the port states of the supplies of source and sink. The simulator uses only those port states that are referred to in one of the PSTs and are also not marked as UNREACHABLE.

For each domain-crossing pair, the simulator performs analysis between matching functions of a source supply set and a sink supply set. You control the analysis with the -pa\_pstcompflags argument to the vopt command, which produces a message (vopt-9881) in the vopt transcript to indicate the kind of supply functions used in PST analysis. The arguments are:

- -pa\_pstcompflags=pg — (default) Analyzes, simultaneously, one or both of the domain-crossing pairs: SourceSS.power and SinkSS.power and/or SourceSS.ground and SinkSS.ground.
- -pa\_pstcompflags=p — Analyzes the domain-crossing pair: SourceSS.power and SinkSS.power.
- -pa\_pstcompflags=g — Analyzes the domain-crossing pair: SourceSS.ground and SinkSS.ground.

Cross-PST analysis is based on two scenarios in your environment:

- Vertical composition — This is when a PST (created by the IP integrator) in a parent scope is composed of an PST (created by the IP provider) in a child scope.
  - Each power state of the PST in the parent scope must enable at least one power state of every PST in a child scope, otherwise the Power Aware simulation marks the states as UNDETERMINED and ignores them for PST analysis. All potential Errors are flagged as Warnings.
  - If the PST in the child scope contains power states that are not enabled by any power state of the PST in the parent scope, then those states are marked as UNREACHABLE states and ignored for PST analysis. A Warning message is flagged for the state.
- Horizontal composition — This is when a PST present in a scope is composed with other PSTs present in the same scope.
  - All PSTs present in same scope and referring to same or equivalent supply should use the same set of port states for that supply or equivalent supplies. If there is a PST state that refers to a port state which is absent in another PST in the same scope and having the same or equivalent supply, it is marked as UNDETERMINED. A warning is flagged and the state is ignored for PST Analysis.

## Level Shifter Specification

Power Aware simulation parses the **set\_level\_shifter** command in the UPF file and selects a list of candidate ports for level shifter insertion.

These ports are also dumped into the UPF report file (*report.pa.txt*) as in the following example:

```

Level Shifter Strategy: my_ls, File: ./src/simple_mv7/test.upf(63).
    Rule (high_to_low), Threshold (0), Applies_to (outputs).
    Level Shifted Candidate Ports:
        1. Signal : /tb/TOP/bot2/out1_bot

Level Shifter Strategy: my_ls_bot3, File: ./src/simple_mv7/test.upf(69).
    Rule (low_to_high), Threshold (0), Applies_to (outputs).
    Level Shifted Candidate Ports:
        1. Signal : /tb/TOP/bot3/out1_bot

```

## Threshold Control for Level Shifters

You can set a global threshold level for a Power Aware analysis containing multiple voltage levels using the following command:

```
vopt -pa_lsthreshold <real>
```

where <real> is any numerical value that specifies a voltage threshold.

Use this argument with the vopt command when you know that level shifting is not required for particular range of voltage differences. You can then specify a global threshold— otherwise Power Aware simulation flags missing level shifter errors even if the potential difference between two domains is within an acceptable range.

### Level Shifter Instances

If you have used the [set\\_level\\_shifter](#) -instance command in a UPF file to instantiate level shifters, Power Aware simulation detects those instances and perform level shifting checks on them (see [Static RTL Checks](#)).

An instance is recognized as a level shifter instance in any of the following cases:

- The level shifter instance is specified with **-instance** option of [set\\_level\\_shifter](#) command.
- For a gate-level design, any of the following:
  - The **is\_level\_shifter** attribute is specified for the module. Example:

```

(* is_level_shifter = 1 *)
module ls_buf(
    (*pg_type = "primary_power"*) input logic
    pwr_rail,
    (*pg_type = "primary_ground"*) input logic
    gnd_rail,
    (* level_shifter_data_pin = 1 *)input data,
    output logic out);
    assign out = (data);
endmodule

```

- Instantiation has prefix or suffix string for level shifter specified with **name\_format** UPF command. Example:

```
LVLHLD1BWP lsinst2_UPF_LS(.I(w2), .Z(w4));
```

### Limitations on Level Shifting

- Support for VHDL and Verilog synthesizable data types. Restricted support for SystemVerilog (array, struct).
- Level shifters not specified at the power-domain boundary are not considered for multi-voltage checks.

## Isolation and Level Shifting Restriction on a Port

The `set_isolation` and `set_level_shifter` UPF commands each have -source and -sink options, which you can use to apply isolation or level shifting only to certain paths of a specific port in your design. When you specify either or both of these options, Power Aware simulation identifies all the paths through the given port and applies isolation or level shifting to only those paths whose driver and receiver supplies match the specified source and sink supplies.

### Isolation and Level Shifting Behavior

Using the -source and -sink options affects Isolation ([set\\_isolation](#)) and Level Shifter ([set\\_level\\_shifter](#)) insertion behavior in the following ways:

- Determine all the paths passing through a given port.  
To determine the path, all the buffers, isolation cells, and level shifter cells are treated as feedthroughs and actual drivers and receivers are determined.
- Insert isolation or level shifter cells after matching source or sink supplies, if specified.  
To match equivalent supplies, the driver nets of primary power and ground nets are matched.
- Determine, via the -location option (for both commands), the placement of an isolation or level shifter cell.

You can specify any of the following values for **set\_isolation -location** or **set\_level\_shifter -location**:

- Fanout — isolation or level shifter cell is placed at all fanout locations (sinks) of the port.
- Fanin — isolation or level shifter cell is placed at all fanin locations (sources) of the port.

- Faninout — isolation or level shifter cell is placed at all fanout locations (sinks) for each output port, or at all fanin locations (sources) for each input port.
- Parent — isolation or level shifter cell is placed in the parent of the domain whose interface port is being isolated or shifted.
- Automatic — same as Parent.
- Self — isolation or level shifter cell is placed inside the domain whose interface port is being isolated or shifted.
- Sibling — same as Self.

---

**Note**

 Level shifter cells are not currently inserted in RTL — their effect is not present in simulation. Only Power Aware checking (vopt -pa\_checks) validate these cells.

---

#### Usage of the **-source** and **-sink** Options

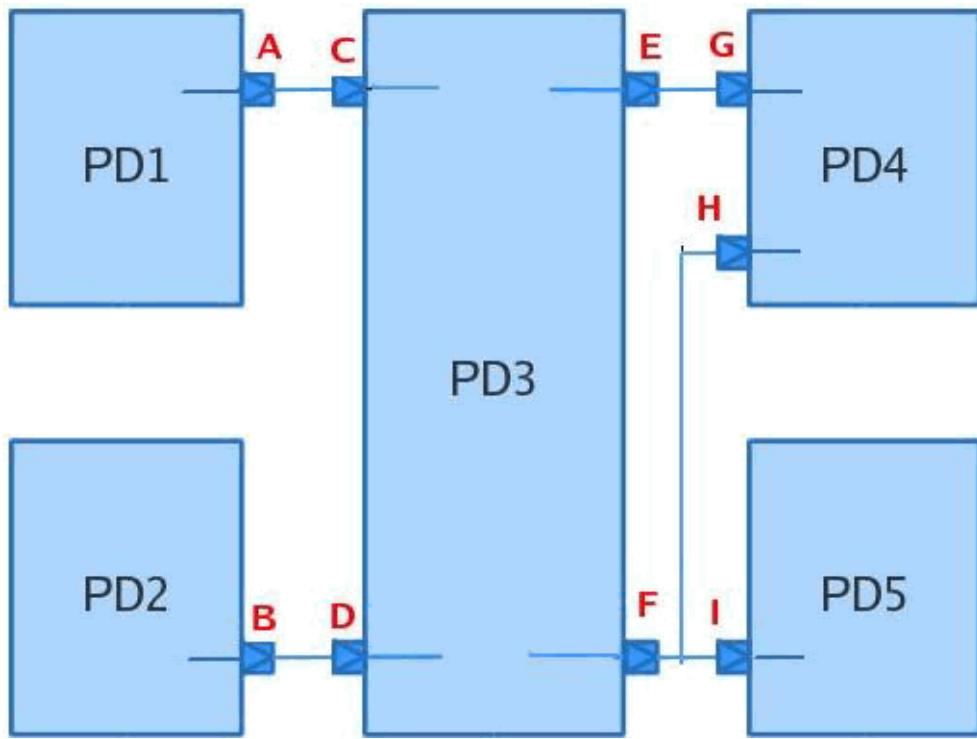
[Figure A-1](#) shows a block diagram of power domains, ports, and paths for use in the examples that follow.

#### Example A-2. UPF Commands That Define Power Domains

The following list shows fragments of UPF commands used to define the diagram of [Figure A-1](#):

```
create_supply_set PD1_SS ...
create_power_domain PD1 ...
associate_supply_set PD1_SS -handle PD1.primary
create_supply_set PD2_SS ...
create_power_domain PD2 ...
associate_supply_set PD2_SS -handle PD2.primary
create_power_domain PD3 ...
```

**Figure A-1. Supply Paths to Power Domains**



#### Source Examples

- Place isolation cell at Port C and isolate Path A-C.

```
set_isolation isol1 -domain PD3 -source PD1_SS -location parent ...
```

- Place isolation cell at Port B and isolate Path B-D.

```
set_isolation iso2 -domain PD3 -source PD2_SS -location fanin ...
```

#### Sink Examples

- Place isolation cells at Port G and Port H and isolate Path E-G and F-H.

```
set_isolation isol1 -domain PD3 -sink PD4_SS -location fanout ...
```

- Place isolation cell at Port H and isolate Path F-H.

```
set_isolation iso2 -domain PD3 -elements {F} -sink PD4_ss  
-location fanout ...
```

- Place isolation cell at Port F and isolate Path F-H.

```
set_isolation iso3 -domain PD3 -elements {F} -sink PD4_ss  
-location parent ...
```

#### Differential Supply Examples

You can prevent the application of isolation into a path from driver to receiver for isolation strategy by using the **-diff\_supply\_only** option to the **set\_isolation** command.

For these examples, assume that **PD2**, **PD3** and **PD4** all have same supply sets:

```
create_supply_set PD2_SS ...
create_power_domain PD2 ...
associate_supply_set PD2_SS -handle PD2.primary
associate_supply_set PD2_SS -handle PD3.primary
associate_supply_set PD2_SS -handle PD4.primary
```

- Place isolation cells at Ports C and F and isolate Paths A-C and F-I. Do not isolate path F-H.

```
set_isolation iso1 -domain PD3 -applies_to both
-diff_supply_only TRUE -location parent ...
```

- Place isolation cells at Ports A and I and isolate Paths A-C and F-I.

```
set_isolation iso2 -domain PD3 -applies_to both
-diff_supply_only TRUE -location faninout ...
```

- Place isolation cell at Port A and isolate Path A-C.

```
set_isolation iso3 -domain PD3 -applies_to both -source PD1_SS
-diff_supply_only TRUE -location faninout ...
```

### Multiple Strategies Example

- Isolate the same port F with different Sinks. Here, Port H is isolated with iso1 and port I with iso2.

```
set_isolation iso1 -domain PD3 -elements {F} -sink PD4_SS
-location fanout -clamp 1 ...
```

```
set_isolation iso2 -domain PD3 -elements {F} -sink PD5_SS
-location fanout -clamp 0 ...
```

### Multiple Isolation Cells Examples

Refer to [Figure A-2](#) for the following examples on using multiple isolation cells.

- Using -source and -location fanout — Isolates all output port paths and places isolation cells at Ports G, H and I. Places two isolation cells at Port H and I for the same port F.

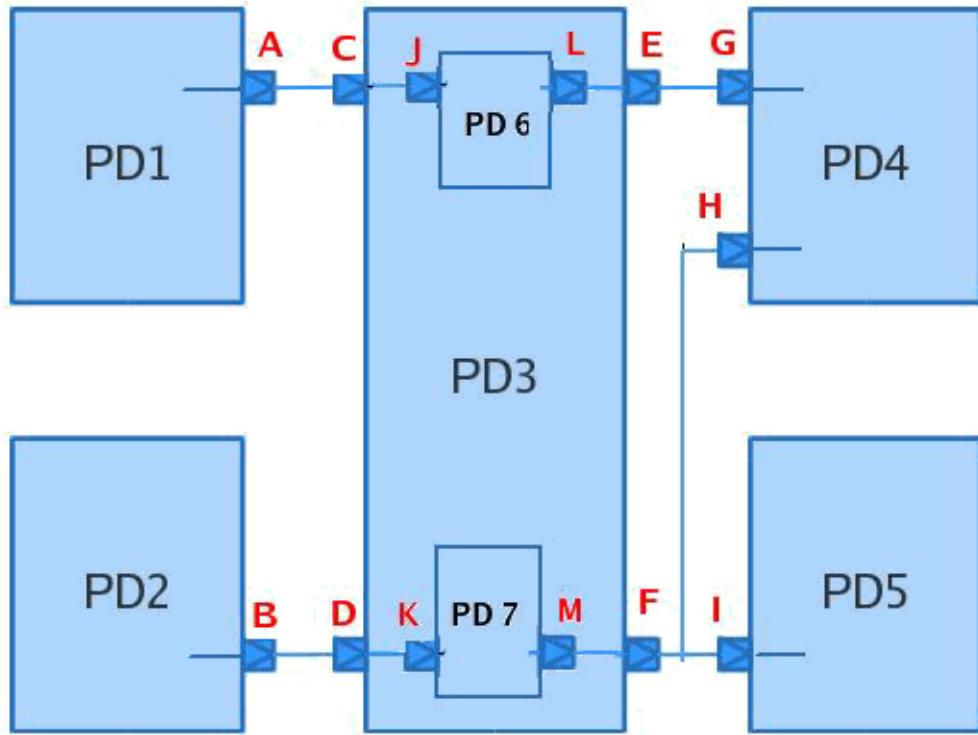
```
set_isolation iso1 -domain PD3 -source PD3_SS -location fanout ...
```

- Using -source and -location parent — Places isolation cells at Ports E and F. Places only one Isolation Cell at Port F, which isolates both Paths F-H and F-I.

```
set_isolation iso2 -domain PD3 -source PD3_SS -location parent ...
```

- Places isolation at Port G. Relative ordering is maintained with iso3 cell in front of iso4 cell, means Iso3 -> iso4 -> port G.

```
set_isolation iso3 -domain PD3 -elements {E} -location fanout  
-clamp 1 ...  
  
set_isolation iso4 -domain PD4 -elements {G} -location parent  
-clamp 0 ...
```

**Figure A-2. Multiple Isolation Cells****Multiple Strategies in a Path Examples**

Refer to [Figure A-2](#) for the following examples on using multiple strategies in a path.

- Places isolation cell at Port H and isolates path M-F-H.

```
set_isolation iso1 -domain PD7 -sink PD4 -location fanout ...
```

- Places isolation cell at Port M and isolates only path M-F-H. This means the clamp value is seen at Port H. All other ports M, F and I do not have clamp value during power down.

```
set_isolation iso2 -domain PD7 -sink PD4 -location parent ...
```

- Places three isolation cells at Port H with relative ordering: iso3 > iso4 > iso5 > port H.

```
set_isolation iso3 -domain PD7 -sink PD4 -location fanout ...
```

```
set_isolation iso4 -domain PD3 -elements {F} -sink PD4  
-location fanout ...
```

```
set_isolation iso5 -domain PD4 -elements {H} -location parent ...
```

- Places two isolation cells at Port F and isolates only port H with relative ordering: Port F > iso6 > iso7. This means the clamp value (o/p of iso7) is seen at Port H. All other ports (M, F and I) do not have clamp value during power down.

```
set_isolation iso6 -domain PD7 -sink PD4 -location fanout ...
set_isolation iso7 -domain PD3 -elements {F} -sink PD4
    -location parent ...
```

## Simulation of Designs Containing Macromodels

A macromodel is a block-level model in a design that has been optimized for power, area, or timing and has been silicon-tested. Defining the power intent for a macromodel depends on whether you have access to its internal structure (logic and topology). If internal access is not available, you can specify power intent only on its external pins—this is referred to as a “hard macro.”

To specify power intent for a hard macro, you define related-supplies attributes on these accessible pins as boundary ports. Similarly, you can isolate a hard macro from the rest of the design by applying the isolation on its boundary ports.

Specifying power intent for a hard macro is available by any of the following methods:

- In a UPF file with the commands:
  - [set\\_port\\_attributes](#)
  - [set\\_pin\\_related\\_supply](#)
- Related-supplies attributes in RTL
- Using a Liberty file in GLS

### UPF Commands for Defining Hard Macro Power Intent

To specify hard macro power intent, use the UPF commands and their options described below.

- [set\\_port\\_attributes](#)
  - receiver\_supply — This option specifies the supply of the logic reading the port. If the receiving logic is not within the logic design starting at the design root, it is presumed the receiver supply is the supply for the receiving logic.
  - driver\_supply — This option specifies the supply of the logic driving the port. If the driving logic is not within the logic design starting at the design root, it is presumed the driver supply is the supply for the driver logic and the port is corrupted when the driver supply is in a simstate other than NORMAL.

- -related\_power\_port | -related\_ground\_port — Either of these options create an implicit supply set containing the supply nets connected to the ports. The behavior differs, depending on the mode of the port being attributed:
  - in mode — the implicitly created supply set is treated as the -receiver\_supply set.
  - out mode — the implicitly created supply set is treated as the -driver\_supply set.
  - inout mode — the implicitly created supply set is treated as both the -receiver\_supply and -driver\_supply set.
- -receiver\_supply — If you use this option and if the receiving logic is within the logic design starting at the design root, it shall be an error if its supply is not the receiver supply.
- -driver\_supply — If you use this option and if the driver logic is within the logic design starting at the design root, it shall be an error if its supply is not the driver supply.
- [set\\_pin\\_related\\_supply](#)

This command defines the related power and ground pins for signal pins on a library cell. It conveys information similar to **related\_power\_pin** and **related\_ground\_pin** in Liberty, but it *may* override them. This command is restricted to only leaf-library cells and non-synthesizable hierarchical modules.

#### Attributes in RTL

You can define the following attributes in RTL to specify the power intent of hard macros:

- UPF\_related\_power\_pin
- UPF\_related\_ground\_pin

System Verilog Example:

```
(* UPF_related_power_pin = "my_Vdd" *) output my_Logic_Port;
```

VHDL Example:

```
attribute UPF_related_power_pin of my_Logic_Port : signal is "my_Vdd";
(* UPF_related_power_pin = "my_Vdd" *) output my_Logic_Port;
```

#### Liberty File

You can define the following attributes at the pins in a Liberty file:

- related\_power\_pin
- related\_ground\_pin

The related\_power\_pin and related\_ground\_pin attributes are defined at the pin level for output, input, and inout pins. These attributes associate a predefined power and ground pin with the

corresponding signal pins under which they are defined. A default related\_power\_pin and related\_ground\_pin always exist in any cell.

#### Example of Power Intent on a Hard Macro

[Figure A-3](#) shows an example of a block diagram of hard macro power domains, using the following top-level UPF definition:

```

set_scope top

create_power_domain PD_top -include_scope

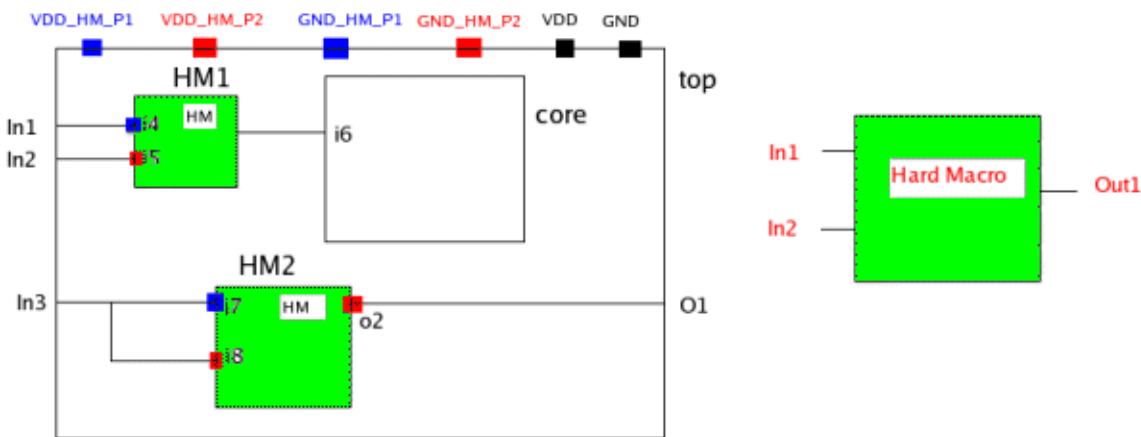
create_power_domain PD_HM1 -elements { HM1 }

set_domain_supply_net PD_top -primary_power_net VDD
    -primary_ground_net GND

set_domain_supply_net PD_HM1 -primary_power_net VDD_HM_P1
    -primary_ground_net GND_HM_P1

```

**Figure A-3. Design Consisting of Hard Macros**



The following sections show how to define the hard macro power intent for each method.

#### UPF Commands

Constraints are specified on pins.

```

set_pin_related_supply HM1 -pins { i4 } -related_power_pin VDD_HM_P1
    -related_ground_pin GND_HM_P1

set_port_attributes -ports { HM2/i7 } -related_power_port VDD_HM_P1
    -related_ground_port GND_HM_P1

set_port_attributes -ports { HM2/i8 HM2/o2 HM1/i5 } -related_power_port
    VDD_HM_P2 -related_ground_port GND_HM_P2

```

#### RTL Attributes

For HM1 —

```
(* UPF_related_power_pin = "VDD_HM_p1", UPF_related_ground_pin =
"GND_HM_p1" *) input in4;
(* UPF_related_power_pin = "VDD_HM_p2", UPF_related_ground_pin =
"GND_HM_p2" *) input in5;
```

For HM2 —

```
(* UPF_related_power_pin = "VDD_HM_p1", UPF_related_ground_pin =
"GND_HM_p1" *) input in7;
(* UPF_related_power_pin = "VDD_HM_p2", UPF_related_ground_pin =
"GND_HM_p2" *) input in8;
(* UPF_related_power_pin = "VDD_HM_p2", UPF_related_ground_pin =
"GND_HM_p2" *) output o2;
```

## Liberty File Attributes

```
library (PALIB) {
cell (HM) {
pg_pin (VDD_HM_P1) {
pg_type : primary_power;
user_pg_type : "abc";
voltage_name : COREVDD1;
}

pg_pin (GND_HM_P1) {
pg_type : primary_ground;
voltage_name : COREGND1;
}

pg_pin (VDD_HM_P2) {
pg_type : backup_power;
user_pg_type : "abc";
voltage_name : COREVDD1;
}

pg_pin (GND_HM_P2) {
pg_type : backup_ground;
voltage_name : COREGND1;
}

pin(in1) {
direction : input;
related_ground_pin : GND_HM_P1;
related_power_pin : VDD_HM_P1;
}

pin(in2) {
direction : input;
related_ground_pin : GND_HM_P2;
related_power_pin : VDD_HM_P2;
}

pin(out1) {
direction : input;
related_ground_pin : GND_HM_P2;
related_power_pin : VDD_HM_P2;
}
}
}
```



# Appendix B

## Model Construction for Power Aware Simulation

---

Power Aware verification uses Verilog behavioral models of Power Aware cells. These models encapsulate the Power Aware behaviors of various types of design state, such as clock-low retention flip-flops and active-high retention latches.

Verilog HDL constructs and attributes for Power Aware models provided by a silicon (or library) vendor. These models trigger relevant events for the simulator to modify the runtime behavior of the design. Typically, these modifications consist of corrupting states and output values and storing or restoring state values based on power control network activity.

Typically, your UPF power specification file relates inferred registers or latches to these models. However, it is possible to capture Power Aware functionality in combination with register and latch functionality in a single model. Because of this, you can create Power Aware models in Verilog to specify Power Aware behavior for inferred registers and latches, as well as provide the functionality directly through direct instantiation of the Power Aware models. Combining both of these functional descriptions in a single model facilitates the testing of the model for use in Power Aware verification.

<b>Basic Model Structure .....</b>	<b>347</b>
<b>Named Events in Power Aware .....</b>	<b>349</b>
<b>Attributes .....</b>	<b>350</b>
<b>Model Interface Ports .....</b>	<b>351</b>
<b>Model Construction Examples .....</b>	<b>353</b>

## Basic Model Structure

Model vendors can implement the Verilog model in any style. The cells communicate important events to the simulator using named events. Power Aware verification defines a standard set of named event identifiers that are used in the model. Each named event corresponds to a particular action to be taken by the simulator.

The simulator communicates with the model by connecting the model to the clock, reset, power (on/off) and power retention signals. Through events on these signals, the model determines when the Power Aware events are triggered, notifying the simulator that the normal RTL behavior must be modified to reflect power control network activity. Because the only inputs are single-bit inputs and the only “output” to the simulator is the triggering of named events, the

models (at the RTL or higher abstraction level) are general-purpose and can work with inferred registers and latches of any data type (for VHDL and SystemVerilog support).

The model communicates with the simulator only by triggering the defined named events. The simulator then maps the event trigger into the appropriate Power Aware behavior for the inferred register or latch that the Power Aware model is associated (using the Power Specification File).

The port interface to the Power Aware model can contain additional port declarations. For example, a Power Aware latch model might define **enable**, **data in** and **data out** ports. For the purpose of Power Aware RTL verification, these additional ports are ignored and is not connected to the functional network of the design.

As additional ports are permitted (but ignored), it is feasible to define a Power Aware model that can be verified as functionally correct as it can be instantiated into a test circuit and exercised to ensure it triggers the Power Aware events at the appropriate time and that saved and restored values match what is expected.

A benefit of this approach is that a single Power Aware model can be created for both gate-level verification and Power Aware RTL verification. The gate-level functionality can be used in gate level simulations and would include the cell's Power Aware functionality but not the triggering of the Power Aware events that are designed for use in RTL (or higher) abstraction level simulations. The Power Aware events can be used without the overhead of the gate-level functionality, for RTL and higher abstraction simulations. Together, both sets of functionality can be used to verify the correctness of the model. The example below shows the use of conditional compilation to control inclusion of functional code, the Power Aware code or both in a simulation.

Modeling using conditional compilation implies that the model would be compiled twice into different libraries for use in gate-level and RTL simulations. An alternative would be the use of parameters and conditional generates to control the inclusion of functionality. In any case, the ability to specify a single model for use at multiple abstraction levels simplifies the support and maintenance associated with the development and deployment of Power Aware IP models.

## Assumptions and Advantages

- The silicon foundry is responsible for the specification of these models to match the behavior of their Power Aware cell technology.
- Capturing the Power Aware behavior in standard Verilog gives the vendor the flexibility to add new cell types and behaviors without creating additional simulation requirements for those cells.
- Foundries providing Power Aware models have control over the protection of their intellectual property (IP).

## Named Events in Power Aware

The following declarations of named events in a Power Aware Verilog model control simulator activity as indicated.

```
event pa_store_value
```

The simulator stores the current value of the inferred registers or latches that the Power Aware model is associated within the power specification file.

```
event pa_store_x
```

The simulator stores a corruption value for the inferred registers or latches. Corruption values depend on the type of data inferring the register or latch. A table mapping corruption values to data types is specified separately.

```
event pa_restore_value
```

The simulator restores the value previously saved for the inferred registers or latches to the corresponding signal(s) in the design. Restoration of a value results in an event on the corresponding signal to facilitate propagation of known, good states throughout a block that has had power restored.

```
event pa_restore_x
```

The simulator restores (re-initializes) the inferred registers or latches to an unknown state specified by the corruption value for the signal's data type. The simulator propagates an event on the restored corruption value.

```
event pa_corrupt_register;           // corrupt the register
```

The simulator corrupts the current value of the signal corresponding to the inferred registers or latches. The corruption value used is determined by the data type/corruption value table specified separately. No event is propagated due to the corruption. NOTE: See [Usage Note for Sequence Requirements](#) (below).

```
event pa_set_register;              // set the register
```

The simulator sets the current value of the signal corresponding to the inferred register or latch to a set value, which is inferred from the RTL code. The simulator propagates an event on the set signal value.

```
event pa_reset_register;           // reset the register
```

The simulator sets the current value of the signal corresponding to the inferred register or latch to a reset value, which is inferred from the RTL code. The simulator propagates an event on the reset signal value.

```
event pa_restore_hold_register
```

The simulator restores the value previously saved and holds that value until a pa\_release\_register event is raised. NOTE: See [Usage Note for Sequence Requirements](#) (below).

```
event pa_release_register
```

The simulator releases any forced values on a register. If the register is combinational, the simulator re-evaluates the register. NOTE: See [Usage Note for Sequence Requirements](#) (below).

```
event pa_release_reeval_register
```

The simulator re-evaluates a latch at powerup. Forces the register to be re-evaluated if the latch enable is active when power is restored.

```
event pa_iso_on
```

The simulator is notified that an isolation period has begun. Use pa\_release\_register event to identify the end of an isolation period.

The model is responsible for raising the named event when the model of the Power Aware cell is in the appropriate state.

For instance, when the retention signal goes high and the clock is in the proper state in a CLRFF (clock-low, retention flip-flop), the model should raise the pa\_store\_value event. When the power goes low, the pa\_corrupt\_register event should be raised.

## Usage Note for Sequence Requirements

When you use a pa\_restore\_hold\_register or pa\_corrupt\_register event, you must include a corresponding pa\_release\_register or pa\_release\_reeval\_register event in the model in the next sequence. The release event can be in a different always block (for example), but it must be next in the sequence.

# Attributes

To assist in the identification of Power Aware cells and facilitate their mapping inferred sequential elements, attributes are placed within the module to provide easily located information. The attributes names and allowed values, as well as contexts in which they are used, are as specified.

## Retention Cells and Memories

The attribute name is **is\_retention** and the allowed attribute values are the strings corresponding to the **pacell\_type**.

```
(* is_retention = <pacell_type_string> *)
```

Where **pacell\_type\_string** is one of:

```
"FF_CKHI"
"FF_CKLO"
"FF_CKFR"
"LA_ENHI"
"LA_ENLO"
"LA_ENFR"
"RETMEM_CKHI"
"RETMEM_CKLO"
"RETMEM_CKFR"
```

Note that within this context, the pacell types of ANY\_CKHI, ANY\_CKLO and ANY\_CKFR have no meaning as these types are used to map any inferred register or latch. Within the context of attributing a retention cell, that cell is either a register or latch and that information is known at the time of attribution. For example:

```
(* is_retention = "FF_CKFR" *)           // Clock free register
(* is_retention = "RETMEM_CKHI" *)         // Retention memory sensitive
                                            //      on posedge of clock
```

### Isolation Cells

The behavior of isolation cells is automatically introduced at the RTL or higher levels through specification of output corruption. However, it is necessary to attribute the gate level library isolation cell models to ensure that the gate level design matches the verified and specified RTL (or higher) functionality. The **is\_isolation\_cell** attribute is Boolean. For example:

```
(* is_isolation_cell *)      // According to IEEE 1364, this
                             // is equivalent to:
(* is_isolation_cell = 1 *)
```

### Level Shifters

Level shifters imply no functional behavior at RTL or higher. However, they are required to ensure proper operation and scaling of signal values from one voltage domain to another. Attributing level shifter cells in the gate level library ensures the gate level design matches the verified and specified RTL (or higher) design specification. The **is\_level\_shifter** attribute is Boolean. For example:

```
(* is_level_shifter *)      // According to IEEE 1364, this
                             // is equivalent to:
(* is_level_shifter = 1 *)
```

## Model Interface Ports

The model must define the necessary ports using the names as specified in this section. All ports are required even if the optional functionality does not apply to a specific inferred register or latch. This port interface specification allows generic models that can be applied to a variety of inferred registers and latches.

All ports specified below are input ports. The simulator connects the ports to the corresponding functional and power control network signals by name. Verilog is case sensitive.

- PWR — Power control network signal that indicates whether power is on or off for the power island that this model is associated with. This port is always connected.
- Retention port(s) — One of the following must be defined by the module and the retention port(s) is always be connected:
  - RET — Power control network signal that indicates whether or not the state of the inferred register or latch must be saved (or restored).
  - SAVE, RESTORE — Separate ports to signal save and restore separately.
- CLK — Functional network data in enable signal. For registers, this is a clock signal. For latches, this is the enable signal. This port is always connected.
- SET — Functional network control signal indicating that the inferred sequential model's value should be set or preset. The functionality of this port is optional and the port is not connected if there are no inferred set or preset conditions. This port needs to be modeled active high or active posedge. The simulator infers the control signal and its polarity and automatically negates the signal's value when it is connected to the model if it is active low or active negedge.
- RESET — Functional network control signal indicating that the inferred sequential model's value should be reset or clear. The functionality of this port is optional and the port is not connected if there are no inferred reset or clear conditions. The simulator infers the control signal and its polarity and automatically negates the signal's value when it is connected to the model if it is active low or active negedge.

Power and retention ports may be connected to an expression involving two or more signals (each).

## Customizing Activity at Time Zero

The default models present in the mtiPA library are equipped to handle any power aware activity at time zero (0). However, there may be instances where you want to use your own custom simulation models for power aware verification. In such cases, if you require power aware activity at time 0 to be honored then you need to ensure that your models are equipped to do so.

To utilize time zero corruption through your own models instead of the default ones (mtiPA) you must add some extra logic in this model as follows:

```
parameter int pa_time0param = 0;
initial begin
    #0;
    if(pa_time0param == 1 && (PWR === 1'bx || PWR === 1'b0))
        ->pa_corrupt_register;
end
```

The changed models like corrupt.v and upf\_retention\_ret.v, and so on, have the same extra logic, as follows:

```
module CORRUPT(PWR);
    parameter int pa_time0param = 0;
    input PWR;
    event pa_corrupt_register;
    event pa_release_register;

    initial begin
        #0;
        if(pa_time0param == 1 && (PWR === 1'bx || PWR === 1'b0))
            ->pa_corrupt_register;
    end

    always @ (negedge PWR)
        -> pa_corrupt_register;
    always @ (posedge PWR)
        -> pa_release_register;
endmodule
```

## Model Construction Examples

You can use these examples as a base for creating your own models.

### Register Model

The following Verilog code represents a behavioral model of a Clock Free Retention Flip Flop (CFRFF) modified to use many of the listed named events. The RTL verification event generation functionality is separated from the “gate-level” cell functionality to demonstrate how a single model can be defined for use at both RTL and gate levels for Power Aware verification as well as facilitate the verification of both functional aspects of the model.

```
module CFRFF (
    PWR, RET, CLK, SET, RESET
`ifdef PA_GLS_FUNC // Extra ports would be left unconnected
    , D, Q           // It is not necessary to conditionally
`endif                  //      compile them out for RTL PA
) ;

    input  PWR;
    input  RET;
    input  CLK;
    input  SET;      // Not used in this model
    input  RESET;
`ifdef PA_GLS_FUNC
    input  D;
    output Q;
    reg   Q;

    reg  reg_q;
    reg  reg_q_ret;
    reg  ret_value;
    reg  restore_value;
    reg  posedge_power_w_reset;
    reg  negedge_ret_w_reset;
    reg  reset_active;
`endif // PA_GLS_FUNC

    // MG event declarations
`ifdef PA_RTL_FUNC          // Would not be needed in GLS
    event pa_store_value;
    event pa_store_x;
    event pa_restore_value;
    event pa_restore_x;
    event pa_corrupt_register;
    event pa_reset_register;
`endif // PA_RTL_FUNC

`ifdef PA_GLS_FUNC
// Functionality in this section is used only in Gate Level
// simulations or in the verification of the PA RTL functionality
// (triggering of the appropriate events at the appropriate time).

    initial
        begin
            Q                = 0;
            ret_value        = 0;
            restore_value    = 0;
            posedge_power_w_reset = 0;
            negedge_ret_w_reset = 0;
            reset_active     = 0;
        end

    always @ (PWR, RESET,
              RET, reg_q, ret_value,
              posedge_power_w_reset, negedge_ret_w_reset)
        begin : output_mux
            if(~PWR)
                begin

```

```

        Q <= 1'bx;
    end
else if (posedge_power_w_reset)
    Q <= reg_q;
else if (negedge_ret_w_reset)
    Q <= reg_q;
else if (RET)
begin
    Q <= reg_q_ret;
end
else
    Q <= reg_q;
end

always @( RESET)
begin
    reset_active = RESET;
end

always @(posedge CLK or negedge RESET)
begin : ff_process
if (RESET)
    reg_q <= 1'b0;
else
    reg_q <= D;
end

always @(posedge CLK)
begin : ret_ff_process
if (~RET)
    reg_q_ret <= D;
end

always @(posedge RET)
begin
    ret_value <= reg_q;
end

always @(negedge RET)
    restore_value <= ret_value;

always @(negedge PWR)
begin
if (!RET)
begin
    wait (PWR);
    if (~RESET) posedge_power_w_reset <= 1;
    wait (!CLK);
    wait (CLK);
    posedge_power_w_reset <= 0;
end
end
end

`endif // PA_GLS_FUNC

`ifdef PA_RTL_FUNC
// Functionality in this section is used for Power Aware RTL (or higher)

```

```
// abstraction verification. It can also be combined with the gate
// level functionality for the purpose of verifying both.

    always @(posedge RET)
        -> pa_store_value;

    always @(negedge RET)
        begin
            if ( (RESET) && PWR)
                -> pa_reset_register;
            end

    always @(posedge PWR)
        begin
            if (RET)
                begin
                    -> pa_restore_value;
                end
            end

    always @(negedge PWR)
        -> pa_corrupt_register;

`endif // PA_RTL_FUNC

endmodule
```

### Corrupt Model

The following Verilog code shows how to create a simple corruption model that initiates and releases corruption on a register:

```
module CORRUPT(PWR);
    input PWR;
    event pa_corrupt_register;
    event pa_release_register;

    always @(negedge PWR)
        -> pa_corrupt_register;

    always @(posedge PWR)
        -> pa_release_register;

endmodule // corrupt
```

where

- The `pa_corrupt_register` statement causes the simulator to corrupt the current value of the signal corresponding to the inferred registers or latches.
- The `pa_release_register` statement causes the simulator to release any forced values on a register. If the register is combinational, the simulator re-evaluates the register.

# Appendix C

## UPF Commands and Reference

---

This appendix provides information on the supported UPF versions, commands and options, and attributes, which is a part of the IEEE Std 1801, to express the power management architecture of a design.

<b>Supported UPF Versions.....</b>	<b>357</b>
<b>Supported UPF Commands .....</b>	<b>359</b>
<b>Supported Query Commands.....</b>	<b>417</b>
<b>Supported UPF Attributes .....</b>	<b>427</b>

## Supported UPF Versions

This section provides information on the UPF versions that you can use with Power Aware simulation.

Power Aware simulation supports the following UPF versions:

- UPF 3.0 — IEEE Std 1801-2015:  
<http://standards.ieee.org/findstds/standard/1801-2015.html>
- UPF 2.1 — IEEE Std 1801-2013:  
<http://standards.ieee.org/findstds/standard/1801-2013.html>
- UPF 2.0 — IEEE Std 1801-2009:  
<http://standards.ieee.org/findstds/standard/1801-2009.html>
- UPF 1.0 — A subset of IEEE Std 1801-2009 (UPF 2.0) that correspond to the original Accellera UPF definition.

### Power Aware Simulation Modes

Select the syntax and semantics of your UPF commands by selecting the Power Aware simulation mode. By default, the simulator enables UPF 2.0 syntax and semantics.

Select the hybrid modes when you use the legacy UPF 2.1, 2.0, and 1.0 code in a context requiring UPF 3.0 features. **Table C-1** lists the syntax and semantics of UPF commands in each simulation mode.

**Table C-1. Power Aware Simulation Modes**

Mode	Selected When	Syntax	Semantics
UPF 3.0	<code>vopt -pa_upfversion=3.0</code>	UPF 3.0	UPF 3.0
UPF 2.1	<code>vopt -pa_upfversion=2.1</code>	UPF 2.1	UPF 2.1
UPF 2.0 (default)	<code>vopt -pa_upfversion=2.0</code>	UPF 2.0	UPF 2.0
UPF 1.0	<code>vopt -pa_upfversion=1.0</code>	UPF 1.0	UPF 1.0
Hybrid 3.0	<ul style="list-style-type: none"><li>• <code>vopt -pa_upfversion=3.0</code></li><li>• <code>upf_version</code> (in the UPF file) is 2.1, 2.0, or 1.0</li></ul>	According to the <code>upf_version</code> selected in the UPF file	UPF 3.0
Hybrid 2.1	<ul style="list-style-type: none"><li>• <code>vopt -pa_upfversion=2.1</code></li><li>• <code>upf_version</code> (in the UPF file) is 2.0, or 1.0</li></ul>	According to the <code>upf_version</code> selected in the UPF file	UPF 2.1

## Related Topics

[Using the Standard Power Aware Simulation Flow](#)

## Supported UPF Commands

This section provides information on the UPF commands that you can use with Power Aware simulation.

The description of the UPF and query commands and options use the conventions listed in the following table.

Convention	Description
deprecated	The command is deprecated in the IEEE Std 1801.
no   N	The simulator does not support the command or option.
not applicable   N/A	The command or option is not a part of the IEEE Std 1801, and is introduced in any later UPF version.
yes   Y	The simulator completely supports the command or option.
yes, legacy	The simulator supports the legacy command.
yes, partial   Y (partial)	The simulator partially supports the command or option.

---

### Tip

 The UPF commands follow the given syntax:

---

```
<upf_command> -<option> <argument>
```

---

**Table C-2. List of Supported UPF Commands**

Command	Description
<code>add_domain_elements</code>	Adds design elements to a power domain.
<code>add_port_state</code>	Adds states to a port.
<code>add_power_state</code>	Defines power states of an object.
<code>add_pst_state</code>	Defines the states of each of the supply nets for one possible state of the design.
<code>add_state_transition</code>	Defines named transitions among power states of an object.
<code>add_supply_state</code>	Adds states to a supply port, a supply net, or a supply set function.
<code>apply_power_model</code>	Binds system-level IP power models to instances in the design and connects the interface supply set handles of a previously loaded power model defined with <code>begin_power_model</code> and <code>end_power_model</code> .
<code>associate_supply_set</code>	Associates two or more supply sets.
<code>begin_power_model</code>	Begins the definition of a power model. This command is used in conjunction with the <code>end_power_model</code> command.

**Table C-2. List of Supported UPF Commands (cont.)**

Command	Description
<code>bind_checker</code>	Inserts checker modules and bind them to instances.
<code>connect_logic_net</code>	Connects a logic net to a logic port.
<code>connect_supply_net</code>	Connects a supply net to the supply ports.
<code>connect_supply_set</code>	Connects a supply set to particular elements.
<code>create_composite_domain</code>	Defines a composite domain comprised of one or more subdomains.
<code>create_hdl2upf_vct</code>	Defines a VCT that can be used in converting HDL logic values into state type values.
<code>create_logic_net</code>	Defines a logic net.
<code>create_logic_port</code>	Defines a logic port.
<code>create_power_domain</code>	Defines a power domain and its characteristics.
<code>create_power_state_group</code>	Creates a name for a group of related power states.
<code>create_power_switch</code>	Defines a power switch.
<code>create_pst</code>	Creates a power state table (PST).
<code>create_supply_net</code>	Creates a supply net.
<code>create_supply_port</code>	Creates a supply port on an instance.
<code>create_supply_set</code>	Creates or updates a supply set, or updates a supply set handle.
<code>create_upf2hdl_vct</code>	Defines VCT that can be used in converting UPF supply_net_type values into HDL logic values.
<code>describe_state_transition</code>	Describes a state transition's legality.
<code>end_power_model</code>	Terminates the definition of a power model. It is used in conjunction with the begin_power_model command.
<code>load_simstate_behavior</code>	Loads the simstate behavior defaults for a library.
<code>load_upf</code>	Executes commands from the specified UPF file in the current scope or in the scope of each specified instance.
<code>load_upf_protected</code>	Loads a UPF file in a protected environment that prevents corruption of existing variables.
<code>map_isolation_cell</code>	Maps a particular isolation strategy to a library cell or range of library cells.
<code>map_level_shifter_cell</code>	Maps a level-shifter strategy to a simulation or implementation model.
<code>map_retention_cell</code>	Constrains implementation alternatives, or specifies a functional model, for retention strategies.

**Table C-2. List of Supported UPF Commands (cont.)**

<b>Command</b>	<b>Description</b>
<code>name_format</code>	Defines the format for constructing names of implicitly created objects.
<code>save_upf</code>	Creates a UPF file of the structures relative to the active or specified scope.
<code>set_design_attributes</code>	Applies attributes to models or instances.
<code>set_design_top</code>	Specifies the design top module.
<code>set_domain_supply_net</code>	Sets the default power and ground supply nets for a power domain.
<code>set_equivalent</code>	Declares that supply nets or supply sets are electrically or functionally equivalent.
<code>set_isolation</code>	Specifies an isolation strategy.
<code>set_isolation_control</code>	Specifies the control signals for a previously defined isolation strategy.
<code>set_level_shifter</code>	Specifies a level-shifter strategy.
<code>set_partial_on_translation</code>	Defines the translation of PARTIAL_ON.
<code>set_pin_related_supply</code>	Defines the related power and ground pair for a library cell.
<code>set_port_attributes</code>	Defines information on ports.
<code>set_power_switch</code>	Extends an HDL model containing no more than acknowledge logic to complete switch definition.
<code>set_repeater</code>	Specifies a repeater (buffer) strategy.
<code>set_retention</code>	Specifies a retention strategy.
<code>set_retention_control</code>	Specifies the control signals and assertions for a previously defined retention strategy.
<code>set_retention_elements</code>	Creates a named list of elements whose collective state shall be maintained if retention is applied to any of the elements in the list.
<code>set_scope</code>	Specifies the current scope.
<code>set_simstate_behavior</code>	Specifies the simulation simstate behavior for a model or library.
<code>set_variation</code>	Specifies the variation range for a supply source.
<code>upf_version</code>	Retrieves the version of UPF being used to interpret UPF commands and documents the UPF version for which subsequent commands are written.
<code>use_interface_cell</code>	Specifies the functional model and a list of implementation targets for isolation and level-shifting.

## **add\_domain\_elements**

Adds design elements to a power domain.

### **Support for UPF Standard**

UPF 3.0 — deprecated

UPF 2.1 — deprecated

UPF 2.0 — yes

UPF 1.0 — yes

### **Options**

<b>Option</b>	<b>Comments/Restrictions</b>
< <i>domain_name</i> >	
-elements	

## **add\_port\_state**

Adds states to a port.

### **Support for UPF Standard**

UPF 3.0 — yes, legacy

UPF 2.1 — yes, legacy

UPF 2.0 — yes

UPF 1.0 — yes

### **Options**

<b>Option</b>	<b>Comments/Restrictions</b>
< <i>port_name</i> >	
-state	

## add\_power\_state

Defines power states of an object.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

### Options

Option	3.0	2.1	2.0	Comments/Restrictions
<object_name>	Y	Y	Y	
-supply   -domain   -group   -model   -instance	Y	Y	Y	Introduced in UPF 2.1, the -supply and -domain options can be enabled for UPF 2.0 by using the following argument:  vopt -pa_upfextensions=relaxedsyntax  The -group, -model, and -instance options are introduced in UPF 3.0.
-state	Y	Y	Y	
-supply_expr	Y	Y	Y	
-logic_expr	Y	Y	Y	
-simstate	Y	Y	Y	
-power_expr	N	N/A	N/A	
-legal   -illegal	Y	Y	Y	
-complete	Y	Y	Y	Introduced in UPF 2.1, you can enable the -complete option for UPF 2.0 by using the following argument:  vopt -pa_upfextensions=relaxedsyntax
-update	Y	Y	Y	

## **add\_pst\_state**

Defines the states of each of the supply nets for one possible state of the design.

### **Support for UPF Standard**

UPF 3.0 — yes, legacy

UPF 2.1 — yes, legacy

UPF 2.0 — yes

UPF 1.0 — yes

### **Options**

<b>Option</b>	<b>Comments/Restrictions</b>
<state_name>	
-pst	
-state	

## **add\_state\_transition**

Defines named transitions among power states of an object.

### **Support for UPF Standard**

UPF 3.0 — yes

UPF 2.1 — no

UPF 2.0 — no

UPF 1.0 — no

### **Options**

<b>Option</b>	<b>Comments/Restrictions</b>
-supply   -domain   -group   -model   -instance	
-update	
-transition	

Option	Comments/Restrictions
-from	
-to	
-paired	
-legal   -illegal	
-complete	

## Usage Notes

- If the UPF file does not contain the [describe\\_state\\_transition](#) or [add\\_state\\_transition](#) command, the simulator reports all transitions in the coverage reports, and follows the given semantics to determine the legality of transitions:
  - For ports and PSTs, the transitions to and from an undefined state are illegal transitions
  - All other transitions are legal transitions
- If the UPF file contains the [describe\\_state\\_transition](#) or [add\\_state\\_transition](#) command, the simulator reports the following transitions in the coverage reports:
  - The transitions that you specify using the [describe\\_state\\_transition](#) or [add\\_state\\_transition](#) command

Use the -legal or -illegal option of the commands to specify the legality of transitions

  - The undefined transitions if you use the vopt -pa\_coverage=undeftrans value.

The simulator follows the given semantics to determine the legality of transitions, and flags appropriate messages for an illegal transition:

  - For ports and PSTs, all transitions to and from an undefined state are illegal transitions
  - For power states, the transitions to and from an undefined state that you specify with the -complete option of the [add\\_state\\_transition](#) command are illegal transitions

## [add\\_supply\\_state](#)

Adds states to a supply port, a supply net, or a supply set function.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — not applicable

UPF 2.0 — not applicable

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;object_name&gt;</i>	
-state	

## apply\_power\_model

Binds system-level IP power models to instances in the design and connects the interface supply set handles of a previously loaded power model defined with begin\_power\_model and end\_power\_model.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes

UPF 2.0 — not applicable

UPF 1.0 — not applicable

## Options

Option	3.0	2.1	Comments/Restrictions
<i>&lt;power_model_name&gt;</i>	Y	Y	
-elements	Y	Y	
-supply_map	Y	Y	
-parameters	N	N/A	

## associate\_supply\_set

Associates two or more supply sets.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;supply_set_name_list&gt;</i>	
-handle	

## **[begin\\_power\\_model](#)**

Begins the definition of a power model. This command is used in conjunction with the `end_power_model` command.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — not applicable

UPF 1.0 — not applicable

## Options

Option	3.0	2.1	Comments/Restrictions
<i>&lt;power_model_name&gt;</i>	Y	Y	It is an alphanumeric string and spaces are not allowed.
-for	Y	Y	

## **[bind\\_checker](#)**

Inserts checker modules and bind them to instances.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.0 — yes

UPF 2.1 — yes

UPF 1.0 — yes

## Options

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
<instance_name>	Y	Y	Y	Y	
-module	Y	Y	Y	Y	
-elements	Y	Y	Y	Y	
-bind_to [-arch]	Y	Y	Y	N/A	
-ports	Y	Y	Y	Y	<net_name> accepts symbolic references. See “ <a href="#">Usage Notes</a> ”.
-parameters	Y	N/A	N/A	N/A	

## Usage Notes

---

### Note

---

 The simulator supports the bind\_checker command only in the vopt flow.

---

### <net\_name> Option to -ports

The -ports option accepts the following symbolic references for <net\_name>:

- isolation\_signal

<scope\_name>. <pd\_name>. <iso\_stratgy\_name>. isolation\_signal

- isolation\_supply\_set

<scope\_name>. <pd\_name>. <iso\_stratgy\_name>. isolation\_supply\_set

- isolation\_power\_net

<scope\_name>. <pd\_name>. <iso\_stratgy\_name>. isolation\_power\_net

- isolation\_ground\_net

<scope\_name>. <pd\_name>. <iso\_stratgy\_name>. isolation\_ground\_net

- <supply\_set>.<function\_name>
- save\_signal of retention strategies
 

```
<scope_name>.<pd_name>.<retention_stratgy_name>.save_signal
```
- restore\_signal of retention strategies
 

```
<scope_name>.<pd_name>.<retention_stratgy_name>.restore_signal
```
- retention\_supply\_set
 

```
<scope_name>.<pd_name>.<retention_stratgy_name>.retention_supply_set.<supply_function_name>
```
- retention\_power\_net
 

```
<scope_name>.<pd_name>.retention_power_net
```
- retention\_ground\_net
 

```
<scope_name>.<pd_name>.retention_ground_net
```

## **connect\_logic\_net**

Connects a logic net to a logic port.

### **Support for UPF Standard**

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

### **Options**

<b>Option</b>	<b>Comments/Restrictions</b>
<net_name>	
-ports	
-reconnect	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: <code>vopt -pa_upfextensions=relaxedsyntax</code>

## connect\_supply\_net

Connects a supply net to the supply ports.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — yes, partial

### Options

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
<net_name>	Y	Y	Y	Y	
-elements	Y	N/A	N/A	N/A	
-ports	Y	Y	Y	Y	
-pg_type	Y	Y (parti al)	Y (parti al)	Y	Unsupported argument (UPF 2.1, 2.0): <element_list>
-vct	Y	Y	Y	Y	
-cells	Y	Y	Y	Y	
-domain	Y	Y	Y	Y	
-pins	N/A	N/A	Y	Y	
-rail_connection	N/A	N/A	N	N	

### Usage Notes

Refer to “[UPF Supply Connections](#)” for more information.

## connect\_supply\_set

Connects a supply set to particular elements.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — not applicable

## Options

Option	3.0	2.1	2.0	Comments/Restrictions
<supply_set_ref>	Y	Y	Y	
-connect	Y	Y	Y	
-elements	Y	Y	Y	
-exclude_elements	N	N	N	
-transitive	Y	Y	Y	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: vopt -pa_upfextensions=relaxedsyntax

## Restrictions

- Supply sets specified in strategy context are not automatically connected to the design elements.
- Supply nets in supply sets functioning as predefined supply set functions are not automatically connected according to their predefined function. You must specify explicit automatic connections.

## [\*\*create\\_composite\\_domain\*\*](#)

Defines a composite domain comprised of one or more subdomains.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.0 — yes

UPF 2.1 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<code>&lt;composite_domain_name&gt;</code>	
<code>-subdomains</code>	
<code>-supply</code>	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: <code>vopt -pa_upfextensions=relaxedsyntax</code>
<code>-update</code>	

## Usage Notes

The UPF commands applied on a composite domain are applied only to those domains that are present at that time in the subdomain tree of the composite domain—they are not applied to the subdomains added later.

### Implementation of Supply Sets

The simulator performs the following tasks:

- Searches each supply set handle name provided with the `-supply` option in the existing list of supply sets for that composite domain. If found, the simulator checks the presence of supply set for that handle.
- Displays a warning message if the reference is present, but its name does not match the current the supply set reference name.
- Searches the current scope if the supply set handle did not contain a reference before, but now a reference name is provided, and updates the handle with this reference. The simulator displays an error if it does not find any matching reference.
- Creates a new handle and populates the reference if the supply set handle does not exist, and adds the new handle to the list of supply sets for the current composite domain and to each subdomain.

You can add new subdomains and supply set handles to a composite domain subject to the checks mentioned above.

### Supported Commands for Composite Domains

- `add_power_state`

You can define power states for a composite domain using the `add_power_state` command.

- `associate_supply_set`

The simulator supports the associate\_supply\_set command only for the primary supply. For other supplies, you can associate them by using either of the following ways:

- associate\_supply\_set on each subdomain individually
- using create\_composite\_domain -update -supply {supply\_set\_handle [supply\_set\_ref]}

The simulator displays an error if the primary supply handle already exists in a subdomain and points to a different supply set.

- create\_power\_switch

The simulator creates the power switch in the creation scope of the composite domain.

- create\_supply\_net

The simulator creates the supply net in the creation scope of the composite domain, and applies the command with a -reuse option to each subdomain. If the subdomain belongs to a different creation scope, the simulator applies the command without the -reuse option.

- create\_supply\_port

The simulator creates the supply port in the creation scope of the composite domain.

- set\_isolation

This command is transitively applied to all the subdomains of a composite domain.

The following are also in effect:

- The set\_isolation -domain <composite\_domain> command cannot use the -elements, -exclude\_elements, or -instance options. This may cause conflict in the subdomains.
- You can update the elements individually for each strategy in all the subdomains.
- If there is an error while transitively applying this command, the simulator does not apply the command to that subdomain.

For example:

```
create_composite_domain cd -subdomains {pd2 pd3} -supply {primary
pdsup_ss}
set_isolation cd_iso -domain cd -clamp_value 0 -applies_to outputs -
isolation_signal iso -isolation_sense high -location parent
set_isolation cd_iso -domain pd2 -update -elements {hier_inst/
leaf_inst2/localout_leaf}
set_isolation cd_iso -domain pd3 -update -elements {hier_inst/
leaf_inst3/localout_leaf}
```

- set\_isolation\_control

This command is applied to all the subdomains of a composite domain.

- `set_level_shifter`

This command is applied to all the subdomains of a composite domain.

The following options are not supported (they can also refer to `composite_domain`):

- `-source <domain_name>`
- `-sink <domain_name>`

The following are also in effect:

- The `set_level_shifter -domain <composite_domain>` command cannot use the `-elements`, `-exclude_elements`, or `-instance` options. This may cause conflict in the subdomains.
- You can update the elements individually for each strategy in all the subdomains.
- If any error occurred in transitively applying this command on a subdomain, it is not applied to that subdomain.

The usage example is similar to the `set_isolation` command.

- `set_port_attributes`

If a composite domain is included in the `-domains` option of this command, it is replaced by its subdomains, listed transitively.

- `set_retention`

This command is applied to all the subdomains of a composite domain.

The following shall also apply:

- The `set_isolation -domain <composite_domain>` command cannot have `-elements`, `-exclude_elements`, or `-instance` options. This may cause conflict in the subdomains.
- You can update the elements individually for each strategy in all the subdomains.
- If any error occurred in transitively applying this command on a subdomain, it is not applied to that subdomain.

The usage is similar to the `set_isolation` command.

- `set_retention_control`

This command is applied to all the subdomains of a composite domain.

- `save_upf`

- Interpreted mode — Power Aware dumps the instances of the `create_composite_domain` command once for each composite domain with all the updates included.

All the commands applied on a composite domain are applied to all the subdomains down to the leaf-level power domains. Because a composite domain can contain only subdomains, supply sets, and power states, the simulator does not store strategies, nets, or other objects on it. As a result, these commands appear as multiple commands applied to each power domain that was a part of the subdomain hierarchy of a composite domain.

- Uninterpreted mode —The simulator saves the command texts as-is, and the `create_composite_domain` command may appear multiple times with `-update`. Also, the commands applied on a composite domain are directly dumped.

## **create\_hdl2upf\_vct**

Defines a VCT that can be used in converting HDL logic values into state type values.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — yes, partial

### Options

Option	Comments/Restrictions
<code>&lt;vct_name&gt;</code>	
<code>-hdl_type</code>	Unsupported type: user-defined
<code>-table</code>	

## **create\_logic\_net**

Defines a logic net.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;net_name&gt;</i>	Can be used as a control signal.

## **create\_logic\_port**

Defines a logic port.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;port_name&gt;</i>	
-direction	

## **create\_power\_domain**

Defines a power domain and its characteristics.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — yes

## Options

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
<domain_name>	Y	Y	Y	Y	
-simulation_only	N/A	N	N	N/A	
-atomic	Y	Y	Y	N/A	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: vopt -pa_upfextensions=relaxedsyntax
-elements	Y	Y	Y	Y	Explicit instance names take precedence over those from wildcard expansion.
-subdomains	Y	N/A	N/A	N/A	
-exclude_elements	Y	Y	Y	N/A	
-supply	Y	Y	Y	N/A	
-available_supplies	Y	Y	Y	N/A	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: vopt -pa_upfextensions=relaxedsyntax
-define_func_type	Y	Y	Y	N/A	
-update	Y	Y	Y	N/A	
-include_scope	N/A	N/A	Y	Y	
-scope	N/A	N/A	Y	Y	

## Usage Notes

You cannot explicitly add an instance of a SystemVerilog interface to the extent of a power domain. For example, the simulator displays a warning message for the following UPF command, where `interface_inst` is an instance of a SystemVerilog interface:

```
create_power_domain PD_I -elements {interface_inst}
                     -supply {primary ss_i}
```

## create\_power\_state\_group

Creates a name for a group of related power states.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — not applicable

UPF 2.0 — not applicable

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<group_name>	

## create\_power\_switch

Defines a power switch.

## Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

## Options

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
<switch_name>	Y	Y	Y	Y	
-switch_type	N	N/A	N/A	N/A	
-output_supply_port	Y	Y	Y	Y	
-input_supply_port	Y	Y	Y	Y	
-control_port	Y	Y	Y	Y	
-on_state	Y	Y	Y	Y	
-off_state	Y	Y	Y	Y	
-supply_set	Y	Y	Y	N/A	

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
-on_partial_state	Y	Y	Y	Y	
-ack_port	Y	Y	Y	Y	
-ack_delay	Y	Y	Y	Y	
-error_state	Y	Y	Y	Y	
-domain	Y	Y	Y	Y	
-instances	Y	Y	Y	N/A	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: vopt -pa_upfextensions=relaxedsyntax
-update	Y	N	N/A	N/A	
-output_voltage	Y	Y	Y	Y	 <b>Note:</b> The option is not a part of the IEEE Std 1801. See " <a href="#">Usage Notes</a> ".

## Usage Notes

The `create_power_switch` command supports multiple ON states. The `-output_voltage` option associates every ON state with a voltage value.

```
create_power_switch
...
    [-output_voltage {state_name voltage_value}]*
...

```

## Examples

```
create_power_switch
...
    -output_voltage {SWT0 0.6}
    -output_voltage {SWT1 0.7}
    -output_voltage {SWT2 0.8}
    -output_voltage {SWT3 0.8}
    -output_voltage {SWT4 1.0}
    -output_voltage {SWT5 1.1}
...

```

## create\_pst

Creates a power state table (PST).

## Support for UPF Standard

UPF 3.0 — yes, legacy

UPF 2.1 — yes, legacy

UPF 2.0 — yes

UPF 1.0 — yes

## Options

Option	Comments/Restrictions
<table_name>	
-supplies	

## create\_supply\_net

Creates a supply net.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

## Options

Option	Comments/Restrictions
<net_name>	
-domain	
-reuse	
-resolve [<unresolved   one_hot   parallel   parallel_one_hot   either   weak   strong>]	The either, weak and strong options are not a part of the IEEE Std 1801. See “ <a href="#">Usage Notes on -resolve either, weak, and strong Options</a> ”. You can create a custom resolution function. See “ <a href="#">Usage Notes on Custom Resolution</a> ”.

## Usage Notes on -resolve either, weak, and strong Options

Use the -resolve either, weak and strong options to resolve the state and voltage of a supply net that is driven by multiple supply sources. The output voltage is governed by the voltage divider rule.

The difference between the -resolve strong and weak options is whether the output voltage is resolved to the higher one of the supply sources or the lower one when multiple supply sources are ON. Simulate the design as the best case scenario using the -resolve strong option, and as the worst case scenario using the -resolve weak option.

The difference between the -resolve either and weak options is whether the output supply is resolved to UNDETERMINED or ON when one source supply is UNDETERMINED and the other one is ON. This is another best-case and worst-case scenario consideration.

---

### Note

 If there are more than two supply sources, the resolution mechanism is similar to a binary tree.

---

### Resolution Mechanism of the -resolve either Option

- If both supply sources are FULL\_ON, the supply net state shall be FULL\_ON, and the minimum supply source voltage shall be assigned to the supply net.
- If a supply source is FULL\_ON and the other one is OFF, the supply net state shall be FULL\_ON, and the corresponding supply source voltage shall be assigned to the supply net.
- If both supply sources are OFF, the supply net state shall be OFF, and the supply net voltage shall be unspecified.
- If a supply source is FULL\_ON and the other one is PARTIAL\_ON, the supply net state shall be FULL\_ON, and the minimum supply source voltage shall be assigned to the supply net.
- If a supply source is PARTIAL\_ON and the other one is OFF, the supply net state shall be PARTIAL\_ON, and the corresponding supply source voltage shall be assigned to the supply net.
- If both supply sources are PARTIAL\_ON, the supply net state shall be PARTIAL\_ON, and the minimum supply source voltage shall be assigned to the supply net.
- If any source is UNDETERMINED, the supply net state shall be UNDETERMINED, and the supply net voltage shall be unspecified.

### Resolution Mechanism of the -resolve weak Option

The resolution mechanism of the -resolve weak option is similar to the -resolve either option, except when one source is UNDETERMINED and the other one is FULL\_ON or PARTIAL\_ON. In such cases, the following resolution mechanism is followed:

- If a supply source is UNDETERMINED and the other one is FULL\_ON, the supply net state shall be FULL\_ON, and the corresponding supply source voltage shall be assigned to the supply net.
- If a supply source is UNDETERMINED and the other one is PARTIAL\_ON, the supply net state shall be PARTIAL\_ON, and the corresponding supply source voltage shall be assigned to the supply net.

### Resolution Mechanism of the -resolve strong Option

- If both supply sources are FULL\_ON, the supply net state shall be FULL\_ON, and the maximum supply source voltage shall be assigned to the supply net.
- If a supply source is FULL\_ON and the other one is OFF, the supply net state shall be FULL\_ON, and the corresponding supply source voltage shall be assigned to the supply net.
- If both supply sources are OFF, the supply net state shall be OFF, and the supply net voltage shall be unspecified.
- If a supply source is FULL\_ON and the other one is PARTIAL\_ON, the supply net state shall be FULL\_ON, and the maximum supply source voltage shall be assigned to the supply net.
- If a supply source is PARTIAL\_ON and the other one is OFF, the supply net state shall be PARTIAL\_ON, and the corresponding supply source voltage shall be assigned to the supply net.
- If both supply sources are PARTIAL\_ON, the supply net state shall be PARTIAL\_ON, and the maximum supply source voltage shall be assigned to the supply net.
- If any source is UNDETERMINED, the supply net state shall be UNDETERMINED, and the supply net voltage shall be unspecified.

### Usage Notes on Custom Resolution

Create a custom resolution function of supply\_nets using the following use model:

1. Define a SystemVerilog resolution function with an input type as an array of supply\_net\_type. For example:

```
function supply_net_type resolution_func(supply_net_type bus[]);
```

- Refer to this resolution function in the UPF create\_supply\_net command with a fully qualified name. For example:

```
create_supply_net N1 -resolve sv_package::resolution_function
```

## create\_supply\_port

Creates a supply port on an instance.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — yes, partial

### Options

Option	Comments/Restrictions
<port_name>	
-domain	
-direction	Unsupported argument: inout

## create\_supply\_set

Creates or updates a supply set, or updates a supply set handle.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	3.0	2.1	2.0	Comments/Restrictions
<set_name>	Y	Y	Y	
-function	Y	Y	Y	
-reference_gnd	N/A	Y	Y	The simulator accepts the -reference_gnd option during parsing. However, it has no impact on simulation.
-update	Y	Y	Y	

## **[create\\_upf2hdl\\_vct](#)**

Defines VCT that can be used in converting UPF supply\_net\_type values into HDL logic values.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — yes, partial

## Options

Option	Comments/Restrictions
<vct_name>	
-hdl_type	Unsupported type: user-defined
-table	

## **[describe\\_state\\_transition](#)**

Describes a state transition's legality.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<transition_name>	
-object	Supported objects: port, power state, power switch, and PST
-from	
-to	
-paired	
-legal   -illegal	

## Usage Notes

---

### Note

---

 The describe\_state\_transition command is deprecated in UPF 3.0. Use the add\_state\_transition command instead.

---

To use the describe\_state\_transition command on the power state tables (PST) and ports, specify the following argument to the vopt command:

```
vopt -pa_enable=statetransition
```

## end\_power\_model

Terminates the definition of a power model. It is used in conjunction with the begin\_power\_model command.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

## Options

None

# load\_simstate\_behavior

Loads the simstate behavior defaults for a library.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;lib_name&gt;</i>	
-file	Unsupported argument: <i>&lt;file_list&gt;</i> Load only one file per occurrence of the command.

# load\_upf

Executes commands from the specified UPF file in the current scope or in the scope of each specified instance.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

## Options

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
<upf_file_name>	Y	Y	Y	Y	
-scope	Y	Y	Y	Y	
-version	Y	Y	Y	Y	The option is deprecated in UPF 3.0, but the simulator supports it.
-hide_globals	Y	N/A	N/A	N/A	
-parameters	Y	N/A	N/A	N/A	

## load\_upf\_protected

Loads a UPF file in a protected environment that prevents corruption of existing variables.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

## Options

Option	Comments/Restrictions
<upf_file_name>	
-hide_globals	
-scope	
-version	
-params	

### Note

 The load\_upf\_protected command is not available in the UPF 1.0 standard, and is deprecated in the UPF 3.0 standard. However, the simulator supports the command for all UPF standards.

---

## map\_isolation\_cell

Maps a particular isolation strategy to a library cell or range of library cells.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — deprecated

UPF 2.0 — yes

UPF 1.0 — yes

### Options

Option	Comments/Restrictions
<i>&lt;isolation_name&gt;</i>	
-domain	
-elements	
-lib_cells	
-lib_cell_type	
-lib_model_name	The cell or module must be present in a library visible to the Power Aware vopt command.
-port	Connects <i>&lt;port_name&gt;</i> to <i>&lt;net_name&gt;</i> . See “ <a href="#">Usage Notes</a> ”.

### Usage Notes

This command has an effect only if isolation cells have been inserted. See “[set\\_isolation](#)”.

*<net\_name>* can have any of the following values:

- A logic net name
- A supply net name
- One of the following symbolic references:
  - UPF\_ISO\_ENABLE (Specific to the simulator)  
Refers to the isolation control signal of associated isolation strategy.
  - UPF\_ISO\_PWR (Specific to the simulator)

- Refers to the isolation power net of associated isolation strategy.
- UPF\_ISO\_GND (Specific to the simulator)
  - Refers to the isolation ground net of associated isolation strategy.
- UPF\_GENERIC\_DATA
  - Refers to the port on which the cell is to be placed.
- UPF\_GENERIC\_OUTPUT
  - Refers to the output of the isolation cell.
- isolation\_signal
  - Refers to the isolation control signal of associated isolation strategy.
- isolation\_supply\_set.function\_name
  - The function\_name extension refers to the supply net corresponding to the function it provides to the isolation\_supply\_set.

## **map\_level\_shifter\_cell**

Maps a level-shifter strategy to a simulation or implementation model.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — deprecated

UPF 2.0 — yes

UPF 1.0 — yes

### Options

Option	Comments/Restrictions
<level_shifter_strategy>	
-domain	
-lib_cells	
-elements	

## map\_retention\_cell

Constrains implementation alternatives, or specifies a functional model, for retention strategies.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — yes, partial

### Options

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
<retention_name_list>	Y	Y	Y	Y	
-domain	Y	Y	Y	Y	
-elements	Y	Y	Y	Y	
-exclude_elements	N	N	N	N/A	
-lib_cells	Y	Y	Y	Y	
-lib_cell_type	Y	Y	Y	Y	
-lib_model_name -port_map	Y (parti al)	Y (parti al)	Y (parti al)	Y (parti al)	Unsupported option: -port_map Verification semantics of lib_model are not honored for UDP retention, where flip-flops and latches are written as UDPs and retention is applied on them.

## name\_format

Defines the format for constructing names of implicitly created objects.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — no

## Options

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
-isolation_prefix	Y	Y	Y	N	
-isolation_suffix	Y	Y	Y	N	
-level_shift_prefix	Y	Y	Y	N	
-level_shift_suffix	Y	Y	Y	N	
-implicit_supply_suffix	N	N	N	N	
-implicit_logic_prefix	N	N	N	N	
-implicit_logic_suffix	N	N	N	N	

## save\_upf

Creates a UPF file of the structures relative to the active or specified scope.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

## Options

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
<upf_file_name>	Y	Y	Y	Y	
-scope	Y	Y	Y	Y	The simulator does not support the -scope option in the uninterpreted mode. See “ <a href="#">Usage Notes</a> ”.
-version	N/A	N/A	Y	Y	The simulator does not support the -version option in the uninterpreted mode. See “ <a href="#">Usage Notes</a> ”.
-u	Y	Y	Y	N/A	Enables uninterpreted mode. See “ <a href="#">Usage Notes</a> ”.   <b>Note:</b> The option is not a part of IEEE Std 1801.

## Usage Notes

Power Aware simulation supports the following modes for using the save\_upf command:

- Interrupted Mode (default)

Any command or option that is not supported by Power Aware simulation is written as a comment at the end of the saved UPF file. This file contains supported commands as interpreted by Power Aware simulation, which are written after performing various operations such as semantic checks, resolving net-port connections, and resolving design objects related to a command.

- Uninterrupted Mode

All the UPF commands are saved in the output file without any processing (even if the commands are not supported by Power Aware simulation). This mode filters out any Tcl-specific constructs in the UPF and writes only the UPF commands to the output UPF file.

To write the output file in uninterpreted mode, do either of the following:

- Specify save\_upf -u. In uninterpreted mode, the -scope and -version options of the save\_upf command are not supported. Also, the saved UPF file is a complete replica of the original UPF file (but without any Tcl constructs).
- Specify vopt -pa\_dumpupf <filename>. This saves the UPF file in uninterpreted mode to the <filename> file.

---

### Tip

 Specify UPF commands in a separate file, which is loaded by specifying the filename as the value to the -pa\_tclfile command line argument. Specifically, use this functionality to avoid putting non-standard commands (or commands with non-standard options) in the main UPF file, which enables you to keep the file portable. Also, you must include navigation commands, such as set\_scope, in the -pa\_tclfile file to ensure that UPF commands in that file are applied in the appropriate scope.

---

## set\_design\_attributes

Applies attributes to models or instances.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — not applicable

## Options

Option	3.0	2.1	2.0	Comments/Restrictions
-models	Y	Y	Y	
-elements	Y	Y	Y	
-exclude_elements	N	N	N	
-attribute	Y	Y	Y	See “ <a href="#">Table C-4</a> ” for the list of supported UPF attributes.
-is_leaf_cell	N/A	Y	Y	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: vopt -pa_upfextensions=relaxedsyntax
-is_macro_cell	N/A	Y	Y	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: vopt -pa_upfextensions=relaxedsyntax
-is_soft_macro	Y	N/A	N/A	
-is_hard_macro	Y	N/A	N/A	
-switch_cell_type	N	N/A	N/A	

## set\_design\_top

Specifies the design top module.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

## Options

Option	Comments/Restrictions
<i>&lt;design_top_module&gt;</i>	

## Usage Notes

Refer to “[set\\_scope](#)” for information on functionality specific to UPF 2.1.

# set\_domain\_supply\_net

Sets the default power and ground supply nets for a power domain.

## Support for UPF Standard

UPF 3.0 — yes, legacy

UPF 2.1 — yes, legacy

UPF 2.0 — yes

UPF 1.0 — yes

## Options

Option	Comments/Restrictions
<i>&lt;domain_name&gt;</i>	
-primary_power_net	
-primary_ground_net	

# set\_equivalent

Declares that supply nets or supply sets are electrically or functionally equivalent.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — not applicable

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
-function_only	
-nets	
-sets	

## set\_isolation

Specifies an isolation strategy.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — yes

## Options

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
<strategy_name>	Y	Y	Y	Y	
-domain	Y	Y	Y	Y	
-elements	Y	Y	Y	Y	
-exclude_elements	Y	Y	Y	N/A	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: vopt -pa_upfextensions=relaxedsyntax
-source	Y	Y	Y	N/A	
-sink	Y	Y	Y	N/A	

<b>Option</b>	<b>3.0</b>	<b>2.1</b>	<b>2.0</b>	<b>1.0</b>	<b>Comments/Restrictions</b>
-diff_supply_only	Y	Y	Y	N/A	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument:  vopt -pa_upfextensions=relaxedsyntax
-use_equivalence	Y	Y	Y	N/A	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument:  vopt -pa_upfextensions=relaxedsyntax
-applies_to	Y	Y	Y	Y	
-applies_to_boundary	Y	N/A	N/A	N/A	
-applies_to_clamp	Y	Y	Y	N/A	
-applies_to_sink_off_clamp	Y	Y	Y	N/A	
-applies_to_source_off_clamp	Y	Y	Y	N/A	
-no_isolation	Y	Y	Y	Y	
-force_isolation	Y	Y	Y	N/A	
-location	Y	Y (parti al)	Y (parti al)	N/A	Unsupported arguments: <ul style="list-style-type: none"> <li>• automatic</li> <li>• sibling</li> </ul> The simulator supports the <instance_name> option. See “locationinstance” in <a href="#">Table A-4</a> .
-clamp_value	Y	Y (parti al)	Y	Y	<ul style="list-style-type: none"> <li>• Unsupported arguments:&lt;any&gt;</li> <li>• The &lt;value&gt; option does not support ports of unpacked type.</li> </ul>
-isolation_signal	Y	Y	Y	N/A	
-isolation_sense	Y	Y	Y	N/A	
-isolation_supply	Y	Y	Y	N/A	The simulator does not support a list.  The name of the option in UPF 2.1 is isolation_supply_set.
-name_prefix	Y	Y	Y	N/A	

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
-name_suffix	Y	Y	Y	N/A	
-instance	Y	Y	Y	N/A	
-update	Y	Y	Y	N/A	
-sink_off_clamp	N/A	N/A	N	N/A	
-source_off_clamp	N/A	N/A	N	N/A	
-isolation_power_net	N/A	N/A	Y	Y	
-isolation_ground_net	N/A	N/A	Y	Y	
-transitive	N/A	N/A	N	N/A	

## set\_isolation\_control

Specifies the control signals for a previously defined isolation strategy.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — deprecated

UPF 2.0 — yes, partial

UPF 1.0 — yes, partial

### Options

Option	Comments/Restrictions
<isolation_name>	
-domain	
-isolation_signal	
-isolation_sense	
-location	Unsupported argument: sibling

## set\_level\_shifter

Specifies a level-shifter strategy.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — yes

## Options

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
<strategy_name>	Y	Y	Y	Y	
-domain	Y	Y	Y	Y	
-elements	Y	Y	Y	Y	
-exclude_elements	Y	Y	Y	N/A	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: <code>vopt -pa_upfextensions=relaxedsyntax</code>
-source	Y	Y	Y	N/A	
-sink	Y	Y	Y	N/A	
-use_equivalence	Y	Y	Y	N/A	Introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument: <code>vopt -pa_upfextensions=relaxedsyntax</code>
-applies_to	Y	Y	Y	Y	
-applies_to_boundary	Y	N/A	N/A	N/A	
-rule	Y	Y	Y	Y	
-threshold	Y	Y	Y (parti al)	Y	Unsupported argument (UPF 2.0): <list>
-no_shift	Y	Y	Y	Y	
-force_shift	Y	Y	Y	N/A	

<b>Option</b>	<b>3.0</b>	<b>2.1</b>	<b>2.0</b>	<b>1.0</b>	<b>Comments/Restrictions</b>
-location	Y	Y (parti al)	Y (parti al)	Y	<ul style="list-style-type: none"> <li>If you specify the fanout option, a level shifter is identified for placement at all fanout locations (for valid level shifters) and the count is incremented accordingly in the report.</li> <li>If you specify the self, parent, sibling, or automatic option, then only one level shifter is identified for placement at a fanout location. Thus, the report may show a count of level shifters in some paths as 0.</li> </ul> <p>Unsupported arguments:</p> <ul style="list-style-type: none"> <li>automatic</li> <li>sibling</li> </ul>
-input_supply	Y	Y	Y	N/A	<p>Supported simstates:</p> <ul style="list-style-type: none"> <li>CORRUPT</li> <li>NORMAL</li> </ul> <p>The name of the option in UPF 2.1 is input_supply_set.</p>
-output_supply	Y	Y	Y	N/A	<p>Supported simstates:</p> <ul style="list-style-type: none"> <li>CORRUPT</li> <li>NORMAL</li> </ul> <p>The name of the option in UPF 2.1 is output_supply_set.</p>
-internal_supply	Y	Y	Y	N/A	<p>Supported simstates:</p> <ul style="list-style-type: none"> <li>CORRUPT</li> <li>NORMAL</li> </ul> <p>The name of the option in UPF 2.1 is internal_supply_set.</p>
-name_prefix	Y	Y	Y	Y	
-name_suffix	Y	Y	Y	Y	
-instance	Y	Y	Y	N/A	
-update	Y	Y	Y	N/A	
-transitive	N/A	N/A	N	N/A	

## **set\_partial\_on\_translation**

Defines the translation of PARTIAL\_ON.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

### Options

Option	3.0	2.1	2.0	Comments/Restrictions
OFF   FULL_ON	Y	Y	Y	
-full_on_tools	Y	Y	Y	Defines a list of tools (products) for which translation of PARTIAL_ON to FULL_ON take place.
-off_tools	Y	Y	Y	Defines a list of tools for which translation of PARTIAL_ON to OFF take place.

### Usage Notes

By default, the simulator translates PARTIAL\_ON as OFF. Therefore, you must use this command to change the default translation behavior to FULL\_ON.

### Enabling Checks for Supply Nets Defined with -resolve parallel

For cases where you create a supply net with the create\_supply\_net command and use the -resolve parallel option, use the vopt argument -pa\_rslvnetcheck to enable a check for cases where different states are driven by active drivers of the supply net. The check may issue a warning when any of the following combinations of states occur in the supply sources for the supply net.

- All sources are FULL\_ON — Does not issue a warning due to synchronized sources.
- All sources are OFF — Does not issue a warning due to synchronized sources.
- All sources are PARTIAL\_ON — Refer to the rules below about PARTIAL\_ON collapsing.
- Any source is UNDETERMINED — Issues a warning due to undetermined setting.

- A mix of PARTIAL\_ON and FULL\_ON — Refer to the rules below about PARTIAL\_ON collapsing.
- A mix of PARTIAL\_ON and OFF — Refer to the rules below about PARTIAL\_ON collapsing.
- A mix of FULL\_ON and OFF — Issues a warning because sources are not synchronized.
- a mix of FULL\_ON, PARTIAL\_ON, and OFF — Refer to the rules below about PARTIAL\_ON collapsing.

Depending upon your setting for the UPF command set\_partial\_on\_translation, PARTIAL\_ON may be treated as FULL\_ON or OFF.

When using the default (set\_partial\_on\_translation OFF) any combinations containing PARTIAL\_ON collapses as follows:

- When all sources are PARTIAL\_ON the resulting combination becomes one where all sources are OFF.
- When there is a mix of PARTIAL\_ON and OFF the resulting combination becomes one where all sources are OFF.
- When there is a mix of PARTIAL\_ON, FULL\_ON and OFF the resulting combination becomes a mix of FULL\_ON and OFF.

If you change the default (set\_partial\_on\_translation FULL\_ON) combinations collapses as follows:

- When all sources are PARTIAL\_ON the resulting combination becomes one where all sources are FULL\_ON.
- When there is a mix of PARTIAL\_ON and FULL\_ON the resulting combination becomes one where all sources are FULL\_ON.
- When there is a mix of PARTIAL\_ON, FULL\_ON, and OFF the resulting combination becomes a mix of FULL\_ON and OFF.

If you also specify the -pa\_nopartialontrans argument to vopt, do not apply any PARTIAL\_ON translation for this check. In this case, the simulator issues a warning for any combination where at least one source is UNDETERMINED or when all sources are not synchronized.

### Defining Tool-specific Defaults

You can also define a list of tools for which translation of PARTIAL\_ON to FULL\_ON takes place and the tools for which PARTIAL\_ON to OFF takes place. All tool names are case-insensitive. You can also specify default behavior for an unlisted tool.

**Note**

- To define PARTIAL\_ON translation behavior for Power Aware simulation, specify 'questa' in the tools list. In Power Aware simulation, PARTIAL\_ON has an enum value of 2. The supply\_partial\_on\_translation function also assigns the same enum value of 2.
- 

The following example sets the translation of PARTIAL\_ON to FULL\_ON only for Questa SIM:

```
set_partial_on_translation OFF -full_on_tools questa
```

The following example sets the translation of PARTIAL\_ON to OFF only for Questa SIM:

```
set_partial_on_translation FULL_ON -off_tools questa
```

## set\_pin\_related\_supply

Defines the related power and ground pair for a library cell.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — deprecated

UPF 2.0 — yes

UPF 1.0 — yes

### Options

Option	Comments/Restrictions
<library_cell>	
-pins	
-related_power_pin	
-related_ground_pin	

### Usage Notes

The simulator assumes the driver/receiver logic supply to be the same as specified related supplies—that is, corruption, isolation, and level-shifting behavior is in accordance with the specified related supplies. When the supply specified using set\_pin\_related\_supply (using

-related\_power\_pin or -related\_ground\_pin is a different supply than that of actual driver or receiver logic, Power Aware simulation gives a vopt error message (vopt-9814).

Use the -warning argument of vopt to change the severity of this message to a warning so that simulation may continue:

```
vopt -warning 9814
```

Refer to “[Arguments to Control Power Aware Message Severity](#)” for more information on changing the level of message severity.

## **set\_port\_attributes**

Defines information on ports.

### **Support for UPF Standard**

UPF 3.0 — yes, partial

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — not applicable

### **Options**

<b>Option</b>	<b>3.0</b>	<b>2.1</b>	<b>2.0</b>	<b>Comments/Restrictions</b>
-model	Y	Y	Y	
-elements	Y	Y	Y	
-exclude_elements	N	N	N	
-ports	Y	Y	Y	See “ <a href="#">Usage Notes</a> ”.
-exclude_ports	Y	Y	Y	
-applies_to	Y	Y	Y	
-attribute	Y	Y	Y	

<b>Option</b>	<b>3.0</b>	<b>2.1</b>	<b>2.0</b>	<b>Comments/Restrictions</b>
-clamp_value	Y	Y	Y	<p>The -clamp_value option is reported under the isolation strategy section in the <a href="#">Architecture Report</a>.</p> <ul style="list-style-type: none"> <li>Specifying clamp value to the complete array, &lt;my_array&gt;:</li> </ul> <pre>set_port_attributes -ports my_array -clamp_value 3'b101</pre> <ul style="list-style-type: none"> <li>Specifying clamp value to the individual bits of the array, &lt;my_array[n]&gt;:</li> </ul> <pre>set_port_attributes -ports my_array[0] -clamp_value 1set_port_attributes -ports my_array[1] -clamp_value 0set_port_attributes -ports my_array[2] -clamp_value 1</pre> <p> <b>Note:</b> The clamp_value option supports a hex or a binary value only.</p>
-sink_off_clamp	Y	Y	Y	
-source_off_clamp	Y	Y	Y	
-driver_supply	Y	Y	Y	<p>Supported simstates:</p> <ul style="list-style-type: none"> <li>CORRUPT</li> <li>NORMAL</li> <li>CORRUPT_ON_ACTIVITY</li> </ul>
-receiver_supply	Y	Y	Y	<p>Supported simstates:</p> <ul style="list-style-type: none"> <li>CORRUPT</li> <li>NORMAL</li> <li>CORRUPT_ON_ACTIVITY</li> </ul>
-literal_supply	N	N/A	N/A	
-pg_type	Y	Y	Y	The simulator does not support -pg_type for UPF-created supply port.
-related_power_port	Y	Y	Y	
-related_ground_port	Y	Y	Y	
-related_bias_ports	Y	Y	Y	
-feedthrough	Y	Y	N/A	
-unconnected	Y	Y	N/A	
-is_analog	Y	N/A	N/A	
-is_isolated	Y	N/A	N/A	

<b>Option</b>	<b>3.0</b>	<b>2.1</b>	<b>2.0</b>	<b>Comments/Restrictions</b>
-domains	N/A	N/A	Y	
-exclude_domains	N/A	N/A	N	
-repeater_supply	N/A	N/A	Y	
-transitive	N/A	N/A	N	

## Usage Notes

- The -clamp\_value, -source\_off\_clamp, and -sink\_off\_clamp options affect the filtering of ports specified by the [set\\_isolation](#) command.
- Power Aware simulation assumes the driver/receiver logic supply to be the same as specified related supplies—that is, corruption, isolation, and level-shifting behavior is in accordance with the specified related supplies. When the supply specified by set\_port\_attributes command (using -related\_power\_port or -related\_ground\_port) is a different supply than that of actual driver or receiver logic, Power Aware simulation gives a vopt error message (vopt-9814).

Use the -warning argument of vopt to change the severity of this message to a warning so that simulation may continue:

```
vopt -warning 9814
```

Refer to “[Arguments to Control Power Aware Message Severity](#)” for more information on changing the severity of message.

## Priority of Commands

It is possible that multiple set\_port\_attributes commands attempt to override the same attribute of a specific port. In this case, the simulator resolves the priority of commands based on the following precedence order:

---

### Note

 The set\_port\_attributes command can specify the over-ridable attribute in the UPF file explicitly, or imply it through the HDL or Liberty attribute specifications.

---

1. Command with a part of the port, specified by the -ports option, and without the -model option. For example:

```
set_port_attributes -ports top_tb/ff_in1/d[0] -attribute
UPF_clamp_value Z
```

2. Command with the entire port, specified by the -ports option, and without the -model option. For example:

```
set_port_attributes -ports top_tb/ff_in1/d -attribute
UPF_clamp_value 1
```

3. Command with the port implied by an instance in the -elements option, and with a direction. For example:

```
set_port_attributes -elements top_tb/ff_in2 -applies_to outputs  
-attribute UPF_clamp_value 1
```

4. Command with the port implied by an instance in the -elements option. For example:

```
set_port_attributes -elements top_tb/ff_in2 -attribute  
UPF_clamp_value 1
```

5. Command with a part of the port of a module or library cell, specified by the -ports option, and with the -model option. For example:

```
set_port_attributes -ports out1[0] -model comb -attribute  
UPF_clamp_value Z
```

6. Command with the entire port of a module or library cell, specified by the -ports option, and with the -model option. For example:

```
set_port_attributes -ports out1 -model comb -attribute  
UPF_clamp_value 1
```

## **set\_power\_switch**

Extends an HDL model containing no more than acknowledge logic to complete switch definition.

### **Support for UPF Standard**

UPF 3.0 — deprecated

UPF 2.1 — deprecated

UPF 2.0 — yes

UPF 1.0 — yes

### **Options**

<b>Option</b>	<b>2.0</b>	<b>1.0</b>	<b>Comments/Restrictions</b>
<switch_name>	Y	Y	
-output_supply_port	Y	Y	
-input_supply_port	Y	Y	
-control_port	Y	Y	
-on_state	Y	Y	

<b>Option</b>	<b>2.0</b>	<b>1.0</b>	<b>Comments/Restrictions</b>
-supply_set	Y	N/A	
-on_partial_state	Y	Y	
-off_state	Y	Y	
-error_state	Y	Y	

## **set\_repeater**

Specifies a repeater (buffer) strategy.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — not applicable

UPF 1.0 — not applicable

### Options

<b>Option</b>	<b>3.0</b>	<b>2.1</b>	<b>Comments/Restrictions</b>
<strategy_name>	Y	Y	
-domain	Y	Y	
-elements	Y	Y	
-exclude_elements	Y	Y	
-source	Y	Y	
-sink	Y	Y	
-use_equivalence	Y	Y	
-applies_to	Y	Y	
-applies_to_boundary	Y	N/A	
-repeater_supply	Y	Y	Supported simstates: • CORRUPT • NORMAL • CORRUPT_ON_ACTIVITY The name of the option in UPF 2.1 is repeater_supply_set.

<b>Option</b>	<b>3.0</b>	<b>2.1</b>	<b>Comments/Restrictions</b>
-name_prefix	Y	Y	
-name_suffix	Y	Y	
-instance	Y	Y	
-update	Y	Y	

## **set\_retention**

Specifies a retention strategy.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — yes

### Options

<b>Option</b>	<b>3.0</b>	<b>2.1</b>	<b>2.0</b>	<b>1.0</b>	<b>Comments/Restrictions</b>
<set_retention>	Y	Y	Y	Y	
-domain	Y	Y	Y	Y	
-elements	Y	Y	Y	Y	
-exclude_elements	Y	Y	Y	N/A	
-retention_supply	Y	Y	Y	N/A	The name of the option in UPF 2.1 is retention_supply_set.
-no_retention	Y	Y	Y	N/A	
-save_signal	Y	Y	Y	N/A	
-restore_signal	Y	Y	Y	N/A	
-save_condition	Y	Y	Y	N/A	
-restore_condition	Y	Y	Y	N/A	
-retention_condition	Y	Y	Y	N/A	
-use_retention_as_primary	Y	Y	Y	N/A	
-parameters	Y	Y	Y	N/A	

Option	3.0	2.1	2.0	1.0	Comments/Restrictions
-transitive	N	N	N	N/A	
-instance	Y	Y	Y	N/A	See “ <a href="#">Usage Notes</a> ”.
-update	Y	Y	Y	N/A	
-retention_power_net	N/A	N/A	Y	Y	
-retention_ground_net	N/A	N/A	Y	Y	
-assert_r_mutex	Y	Y	Y	Y	 <b>Note:</b> The option is not a part of the IEEE Std 1801.
-assert_s_mutex	Y	Y	Y	Y	 <b>Note:</b> The option is not a part of the IEEE Std 1801.
-assert_rs_mutex	Y	Y	Y	Y	 <b>Note:</b> The option is not a part of the IEEE Std 1801.

## Usage Notes

Restrictions:

- Corruption of retention element and saved value is not supported.
- The implicit corruption semantics are also applied to the shadow latch used to preserve the data during retention period. In order to remove the shadow latch from corruption, you must specify it in an exclude file (vopt -pa\_excludefile).

### The -instance option

- If there is a port of type pg\_type present on the instance, then Power Aware simulation automatically connects the primary\_power and primary\_ground pins with primary power and primary ground nets of the power domain. It connects the backup power and backup ground pin specified on the instance with the retention power and ground nets specified in the strategy.
- If there is no port of type pg\_type present on the instance, then Power Aware simulation applies implicit corruption semantics according to the primary\_power and ground nets specified for the power domain.

### Configuration Notes

- Master-slave (slave-alive) configuration — Applied when both -save\_signal and -restore\_signal options are absent and -retention\_condition option is present for retention strategy.

When the retention condition is enabled and the register retains the value, then the clock, set, and reset operations are blocked. The corruption happens at power down. However, during power down if the retention condition is false, then the retained value is corrupted. At power up, retained value is put at the register output if retention condition holds. Normal flop operation resumes once retention condition is disabled.

See “[Master-slave \(slave-alive\) Retention Protocol](#)”.

- Balloon-latch configuration — Applied when both -save\_signal and -restore\_signal options are present for retention strategy.

At the save event, the event register output is saved if -save\_condition is true. At power down, the register output is corrupted. At the restore event, the retained value is restored at the register output if the -restore\_condition is true.

## **set\_retention\_control**

Specifies the control signals and assertions for a previously defined retention strategy.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — deprecated

UPF 2.0 — yes

UPF 1.0 — yes

### Options

Option	2.0	1.0	Comments/Restrictions
<retention_name>	Y	Y	
-domain	Y	Y	
-save_signal	Y	Y	
-restore_signal	Y	Y	
-assert_r_mutex	Y	Y	
-assert_s_mutex	Y	Y	
-assert_rs_mutex	Y	Y	

## **set\_retention\_elements**

Creates a named list of elements whose collective state shall be maintained if retention is applied to any of the elements in the list.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes, partial

UPF 1.0 — not applicable

## Options

Option	3.0	2.1	2.0	Comments/Restrictions
<retention_list_name>	Y	Y	Y	
-elements	Y	Y	Y	Specify only instances as value to the option. The simulator does not support processes, sequential registers, or signal names.
-applies_to	Y	Y	Y	
-exclude_elements	Y	Y	Y	Specify only instances as value to the option. The simulator does not support processes, sequential registers, or signal names.
-retention_purpose	Y	Y	Y	
-transitive	Y	Y	Y	
-expand	N/A	N/A	Y (parti al)	Default behavior is equivalent to “-expand TRUE”. The simulator assumes this value is set to TRUE. The simulator does not support setting the option to FALSE.

## set\_scope

Specifies the current scope.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

## Options

Option	Comments/Restrictions
<instance>   <pathname>	

## Usage Notes

In UPF 2.1 and newer versions, use the -pa\_enable=upf2.1hierarchynavigation argument with the vopt command to specify <instance> in the set\_scope command as the following types:

---

### Note

---

 The -pa\_enable=upf2.1hierarchynavigation argument also impacts the [set\\_design\\_top](#) command.

---

- Design-relative hierarchical name, which is interpreted relative to the current design top instance.
- A slash character (/), specifically a hierarchical name that starts with a leading '/' character is a design-relative hierarchical name.

Consider the following example:

```
-----test.sv-----
module tb();
top top1();
top top2();
endmodule

module top;
mid mid1();
mid mid2();
endmodule

module mid;
bot bot1();
bot bot2();
endmodule

module bot;
endmodule
-----

-----test.upf-----
upf_version 2.1
set_design_top tb
create_power_domain pd
set_scope top1
load_upf mid.upf -scope mid1

/*Comment - In mid.upf, if design_top is set and scope specified in
load_upf is not an instance of that design top then the simulator displays
a warning*/
-----

-----mid.upf-----
set_design_top mid      /*Comment - Here set_design_top is not
ignored and is the root scope for mid.upf*/
#set_scope ..           /*Comment - The simulator displays an error
because current scope is mid, which is the
instance of design_top mid, and you can not go
further up from here*/
set_scope bot1          /*Comment - This command returns scope name
mid1, which is the previous scope set*/
set_scope ..            /*Comment - The simulator sets the scope as
instance of the design top i.e mid1*/
-----
```

## **set\_simstate\_behavior**

Specifies the simulation simstate behavior for a model or library.

### **Support for UPF Standard**

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	3.0	2.1	2.0	Comments/Restrictions
ENABLE   DISABLE	Y	Y	Y	
-lib	Y	Y	Y	The -elements option has higher priority over -model and -lib options.
-models	Y	Y	Y	
-elements	Y	Y	Y	
-exclude_elements	N	N	N/A	

## set\_variation

Specifies the variation range for a supply source.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — not applicable

UPF 2.0 — not applicable

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
-supply	
-range	

## upf\_version

Retrieves the version of UPF being used to interpret UPF commands and documents the UPF version for which subsequent commands are written.

### Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — yes

### Options

Option	Comments/Restrictions
<string>	

## use\_interface\_cell

Specifies the functional model and a list of implementation targets for isolation and level-shifting.

### Support for UPF Standard

UPF 3.0 — yes, partial

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — not applicable

### Options

Option	3.0	2.1	2.0	Comments/Restrictions
<interface_implementation_name>	Y	Y	Y	
-strategy	Y	Y	Y	
-domain	Y	Y	Y	
-lib_cells	Y	Y	Y	

<b>Option</b>	<b>3.0</b>	<b>2.1</b>	<b>2.0</b>	<b>Comments/Restrictions</b>
-port_map	Y	Y	Y	
-elements	Y	Y	Y	
-exclude_elements	Y	Y	Y	
-applies_to_clamp	Y	Y	Y	
-update_any	N	N	N	
-force_function	Y	Y	Y	
-inverter_supply_set	N	N	N	

# Supported Query Commands

This section provides information on the query commands that you can use to query your design.

## Note

 The query\_\* commands are deprecated in the IEEE Std P1801-2015 standard, but the simulator supports them. Use the upf\_query\_\* commands instead of the query\_\* commands. See “[Supported Tcl Commands](#)”.

**Table C-3. List of Supported Query Commands**

Command	Description
<a href="#">find_objects</a>	Finds logical hierarchy objects within a scope.
<a href="#">query_cell_instances</a>	Queries the instances of a mapped cell within the active scope.
<a href="#">query_cell_mapped</a>	Queries the cell that is mapped to a specified instance.
<a href="#">query_design_attributes</a>	Queries attributes for a design element or model.
<a href="#">query_isolation</a>	Queries information about previously defined isolation strategies for the specified power domain.
<a href="#">query_port_state</a>	Queries the state information for a specified port.
<a href="#">query_power_domain</a>	Queries a power domain.
<a href="#">query_power_domain_element</a>	Queries the domain membership information for a design element.
<a href="#">query_power_state</a>	Queries the state information for a power domain or supply set.
<a href="#">query_power_switch</a>	Queries information for a UPF power switch.
<a href="#">query_pst</a>	Queries a power state table.
<a href="#">query_pst_state</a>	Queries state information for a PST.
<a href="#">query_retention</a>	Queries the retention strategy information for a domain.
<a href="#">query_retention_control</a>	Queries the retention control information for a domain.
<a href="#">query_supply_net</a>	Queries a supply net.
<a href="#">query_supply_port</a>	Queries a supply port.

## find\_objects

Finds logical hierarchy objects within a scope.

## Support for UPF Standard

UPF 3.0 — yes

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<scope>	
-pattern	
-object_type	
-direction	
-transitive	
-regexp   -exact	
-ignore_case	
-non_leaf   -leaf_only	

## Usage Notes

### Note

 The simulator does not support the search for a multi-dimensional array of instances.

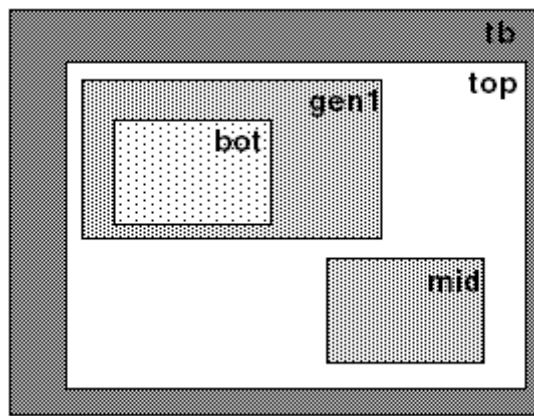
---

Write the output of the `find_objects` command to an external file with the `-pa_dbgfindobj=<filename>` argument to the `vopt` command.

## Examples

- The following example shows the output of the `find_objects` command with and without the `-pa_upfextensions=genblk` argument to the `vopt` command. See “genblk” in [Table A-4](#).

**Figure C-1. Design Hierarchy**



- o Default output

```
find_objects top -pattern {*} -object_type inst
Returns: top/mid
```

```
find_objects top -pattern {*} -object_type inst -transitive TRUE
Returns: top/mid top/gen1/bot
```

```
find_objects top/gen1 -pattern {*} -object_type inst
Returns: Invalid Scope Error.
```

- o Output with the -pa\_upfextensions=genblk argument to the vopt command

```
find_objects top -pattern {*} -object_type inst
Returns: top/mid top/gen1
```

```
find_objects top -pattern {*} -object_type inst -transitive TRUE
Returns: top/mid top/gen1 top/gen1/bot
```

```
find_objects top/gen1 -pattern {*} -object_type inst
Returns: top/gen1/bot
```

## query\_cell\_instances

Queries the instances of a mapped cell within the active scope.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<cell_name>	
-domain	

## query\_cell\_mapped

Queries the cell that is mapped to a specified instance.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<instance_name>	

## query\_design\_attributes

Queries attributes for a design element or model.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
-element	
-model	
-detailed	

## query\_isolation

Queries information about previously defined isolation strategies for the specified power domain.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;isolation_name&gt;</i>	
-domain	
-detailed	

## query\_port\_state

Queries the state information for a specified port.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;port&gt;</i>	
-detailed	

## query\_power\_domain

Queries a power domain.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes, partial

UPF 2.0 — yes, partial

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;domain_name&gt;</i>	
-non_leaf   -all   -no_elements	Unsupported option: -all
-detailed	

## query\_power\_domain\_element

Queries the domain membership information for a design element.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;design_element&gt;</i>	

## **query\_power\_state**

Queries the state information for a power domain or supply set.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;object_name&gt;</i>	
-state	
-detailed	

## **query\_power\_switch**

Queries information for a UPF power switch.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;switch_name&gt;</i>	
-detailed	

## query\_pst

Queries a power state table.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes, legacy

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<i>&lt;table_name&gt;</i>	
-detailed	

## query\_pst\_state

Queries state information for a PST.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes, legacy

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<state_name>	
-pst	
-detailed	

## query\_retention

Queries the retention strategy information for a domain.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<retention_name>	
-domain	
-detailed	

## query\_retention\_control

Queries the retention control information for a domain.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — deprecated

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<retention_name>	
-domain	
-detailed	

## **query\_supply\_net**

Queries a supply net.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<net_name>	
-domain	
-is_supply	
-detailed	

## **query\_supply\_port**

Queries a supply port.

### Support for UPF Standard

UPF 3.0 — deprecated

UPF 2.1 — yes

UPF 2.0 — yes

UPF 1.0 — not applicable

## Options

Option	Comments/Restrictions
<port_name>	
-domain	
-is_supply	
-detailed	

## Supported UPF Attributes

This section provides information on the UPF attributes that you can use to express the power intent in a UPF, design, and Liberty file.

The simulator supports the UPF attributes listed in [Table C-4](#). These attributes are a part of the IEEE Std 1801.

**Table C-4. Supported UPF Attributes**

UPF Attribute	Value(s)	Equivalent UPF Commands
UPF_clamp_value	0   1   Z   latch   any   <value>	<ul style="list-style-type: none"> <li>• set_isolation -clamp_value</li> <li>• set_port_attributes -clamp_value</li> </ul>
UPF_pg_type	<pg_type_value>	set_port_attributes -pg_type
UPF_related_bias_pins	<port_name>	set_port_attributes -related_bias_ports
UPF_related_ground_pin	<port_name>	<ul style="list-style-type: none"> <li>• set_pin_related_supply -related_ground_pin</li> <li>• set_port_attributes -related_ground_port</li> </ul>
UPF_related_power_pin	<port_name>	<ul style="list-style-type: none"> <li>• set_pin_related_supply -related_power_pin</li> <li>• set_port_attributes -related_power_port</li> </ul>
UPF_retention	required   optional	set_retention_elements -retention
UPF_simstate_behavior	ENABLE   DISABLE	set_simstate_behavior
UPF_sink_off_clamp_value	0   1   Z   latch   any   <value>	<ul style="list-style-type: none"> <li>• set_isolation -sink_off_clamp_value</li> <li>• set_port_attributes -sink_off_clamp_value</li> </ul>

**Table C-4. Supported UPF Attributes (cont.)**

UPF Attribute	Value(s)	Equivalent UPF Commands
UPF_source_off_clamp_value	0   1   Z   latch   any   <value>	<ul style="list-style-type: none"> <li>• set_isolation -source_off_clamp_value</li> <li>• set_port_attributes -source_off_clamp_value</li> </ul>
UPF_switch_cell_type	<fine_grain   coarse_grain>	set_design_attributes -switch_type

The simulator supports the UPF attributes listed in [Table C-5](#). These attributes are not a part of the IEEE Std 1801.

**Table C-5. Supported Simulator-specific UPF Attributes**

UPF Attribute	Value(s)	Equivalent UPF Commands
qpa_attr_inactive_reset_duration	<Tmax>	set_design_attributes -attribute
qpa_attr_active_reset_duration	<Twidth>	set_design_attributes -attribute
qpa_dont_replay_init	TRUE	set_design_attributes -attribute
qpa_override_pbp_corruption	TRUE   FALSE	set_design_attributes -attribute
qpa_replay_init	TRUE	set_design_attributes -attribute
upf_dont_touch	true   TRUE	<ul style="list-style-type: none"> <li>• set_port_attributes -attribute</li> <li>• set_design_attributes -attribute</li> </ul>

### **qpa\_attr\_inactive\_reset\_duration**

Sets the maximum time unit (Tmax) within which the asynchronous reset must be asserted after power-up. The UPF attribute is not a part of the IEEE Std 1801.

Usage:

```
set_design_attributes -attribute qpa_attr_inactive_reset_duration Tmax
```

where Tmax represents integer time units.

For example:

```
set_design_attributes -attribute qpa_attr_inactive_reset_duration 10
```

Refer to “*Non-Retention Register Reset*” check in [Table 5-18](#) for more information.

### **qpa\_attr\_active\_reset\_duration**

Sets the minimum time unit (Twidth) for which the asynchronous reset must remain asserted. The UPF attribute is not a part of the IEEE Std 1801.

Usage:

```
set_design_attributes -attribute qpa_attr_active_reset_duration Twidth
```

where Twidth represents integer time units.

For example:

```
set_design_attributes -attribute qpa_attr_active_reset_duration 5
```

Refer to “*Non-Retention Register Reset*” check in [Table C-4](#) for more information.

---

**Note**

-  The attribute qpa\_attr\_active\_reset\_duration is valid only when the attribute qpa\_attr\_inactive\_reset\_duration is specified with a non-zero value.
- 

### **qpa\_dont\_replay\_init**

Disables re-triggering of a Verilog initial block at power up. The UPF attribute is not a part of the IEEE Std 1801.

---

**Note**

-  The attribute qpa\_dont\_replay\_init is valid only when you define the attribute qpa\_replay\_init.
- 

Usage:

```
set_design_attributes -attribute qpa_dont_replay_init TRUE  
[-elements <element_list>] [-models <model_list>] [-transitive  
<TRUE|FALSE>]
```

Assume you include the following two commands in a UPF file:

```
set_design_attributes -attribute qpa_replay_init TRUE  
set_design_attributes -attribute qpa_dont_replay_init TRUE  
-elements PD_Core
```

In this case, the simulator re-triggers all initial blocks present in the design, except the initial blocks present in the power domain, PD\_Core.

### **qpa\_override\_pbp\_corruption**

When the attribute is set to TRUE, the simulator applies Liberty-based corruption to all macro cells, even when the models are Power Aware. The UPF attribute is not a part of the IEEE Std 1801.

Usage:

```
set_design_attributes -attribute {qpa_override_pbp_corruption TRUE}
```

When the attribute is set to FALSE, the simulator does not apply Liberty-based corruption to all macro cells, even when the models are non-Power Aware.

Usage:

```
set_design_attributes -attribute {qpa_override_pbp_corruption FALSE}
```

## qpa\_replay\_init

Enables re-triggering of a Verilog initial block at power up. The UPF attribute is not a part of the IEEE Std 1801.

---

### Note

 The attribute qpa\_replay\_init does not re-trigger an initial block containing a “forever”, “fork join”, “break continue”, “wait”, “disable” or “rand seq” statement.

---

Usage:

```
set_design_attributes -attribute qpa_replay_init TRUE  
[-elements <element_list>] [-models <model_list>] [-transitive  
<TRUE|FALSE>]
```

Examples:

```
set_design_attributes -attribute qpa_replay_init TRUE -models { t* }  
set_design_attributes -attribute qpa_replay_init TRUE  
-elements { top/bot1/sram } -transitive TRUE
```

## Priority of Commands

If a UPF file contains multiple [set\\_design\\_attributes](#) commands with the qpa\_replay\_init attribute set to TRUE, then the simulator resolves the priority of these commands based on the precedence of the options of the set\_design\_attributes command.

The precedence of the options is as follows:

1. Named initial block in a module — A set\_design\_attributes command with a named initial block in a module.

For example, the set\_design\_attributes command with a named initial block, NamedInitialBlock, in the module, ModuleForRetrigg:

```
set_design_attributes -attribute qpa_replay_init TRUE  
-elements NamedInitialBlock -models ModuleForRetrigg
```

2. Design instance — A set\_design\_attributes command with a design instance.

For example, the set\_design\_attributes command with a design instance, DesignInstance:

```
set_design_attributes -attribute qpa_replay_init TRUE  
-elements DesignInstance
```

3. Design module — A set\_design\_attributes command with a design module.

For example, the set\_design\_attributes command with a design module, DesignModule:

```
set_design_attributes -attribute qpa_replay_init TRUE  
-models DesignModule
```

4. Power domain — A set\_design\_attributes command with a power domain.

For example, the set\_design\_attributes command with a power domain, PD\_Core:

```
set_design_attributes -attribute qpa_replay_init TRUE  
-elements PD_Core
```

5. Entire design — A set\_design\_attributes command without any module or instance.

For example:

```
set_design_attributes -attribute qpa_replay_init TRUE
```

## upf\_dont\_touch

Excludes Power Aware behavior on certain instances, modules, or signals. The UPF attribute is not a part of the IEEE Std 1801.

Following are some of the reasons to exclude Power Aware behavior:

- The items are self instantiated isolation or level shifter cells.
- They have their own power model or behavioral model.
- They are power output or inout pins.

Use the upf\_dont\_touch attribute with the set\_design\_attributes or set\_port\_attributes commands. You can specify upf\_dont\_touch attribute on objects such as module, architecture, or entity instance or any signal through a UPF or HDL attribute. The attribute disables Power Aware instrumentation (for example, corruption, retention, isolation, level\_shifter, or buffers) on the specified object.

If an instance or a model is identified with a upf\_dont\_touch attribute then the functionality is applied recursively to its children down the hierarchy. The following limitations exist for this functionality:

- The upf\_dont\_touch attribute supports only the value “true” (or TRUE).
- Bitwise signal exclusion is not honored for isolation, level shifter, or buffer cell insertion, but it does affect corruption and retention.

The following are some UPF examples using the upf\_dont\_touch attribute, and the resulting notes from the simulator:

```
set_design_attributes -attribute upf_dont_touch TRUE -models fpu_*
** Note: src/complex4/dta.tcl(2): (vopt-9716) Excluding power aware module
'fpu_mod' in path '/tb/top/inst1'.

set_design_attributes -attribute upf_dont_touch TRUE -elements top/inst2
** Note: src/complex4/dta.tcl(3): (vopt-9716) Excluding power aware module
'mid3' in path '/tb/top/inst2'.
set_design_attributes -attribute upf_dont_touch TRUE
-elements inst1/blk/*/*s*

** Note:exclude.upf(1): (vopt-9719) Excluding signal 's14' in power aware
module 'mid' in path '/tb/top/inst1/blk/fg(5)/s14'.

** Note:exclude.upf(1): (vopt-9719) Excluding signal 's11' in power aware
module 'mid' in path '/tb/top/inst1/blk/fg(5)/s11'.

set_design_attributes -attribute upf_dont_touch TRUE
-elements top/inst2/sig1[2:0]
** Note:exclude.upf(1): (vopt-9719) Excluding signal 'sig1[2:0]' in power
aware module 'mid' in path '/tb/top/inst1/sig1'.

set_port_attributes -attribute upf_dont_touch TRUE -models mid
-ports {p1 p2}
** Note:exclude.upf(1): (vopt-9719) Excluding signal 'p1' in power aware
module 'mid' in path '/tb/top/inst1/p1'.
** Note:exclude.upf(1): (vopt-9719) Excluding signal 'p2' in power aware
module 'mid' in path '/tb/top/inst1/p2'.
```

The following is an HDL example using the attribute:

```
(*upf_dont_touch = "TRUE"*)
module mid();
...
endmodule
```

# **Appendix D**

## **Supplemental Information**

---

This appendix provides supplemental information on applications of Power Aware.

**Power Aware Verification of ARM-Based Designs .....** **434**

# Power Aware Verification of ARM-Based Designs

---

This section contains an application note written by Ping Yeung and Erich Marschner of Mentor Graphics Corporation.

<b>Abstract</b> .....	<b>434</b>
<b>Introduction</b> .....	<b>435</b>
<b>Active Power Management</b> .....	<b>435</b>
<b>Power Management Techniques</b> .....	<b>435</b>
<b>Power Management Specification</b> .....	<b>436</b>
<b>Power Management Architecture</b> .....	<b>437</b>
<b>Power Managed Behavior</b> .....	<b>444</b>
<b>Power Control Logic</b> .....	<b>444</b>
<b>Power Aware Verification Flow</b> .....	<b>445</b>
<b>Summary</b> .....	<b>448</b>

## Abstract

Power dissipation has become a key constraint for the design of today's complex chips. Minimizing power dissipation is essential for battery-powered portable devices, as well as for reducing cooling requirements for non-portable systems. Such minimization requires active power management built into a device.

In a System-on-Chip (SoC) design with active power management, various subsystems can be independently powered up or down, and/or powered at different voltage levels. It is important to verify that the SoC works correctly under active power management. When a given subsystem is turned off, its state is lost, unless some or all of the state is explicitly retained during power down. When that subsystem is powered up again, it must either be reset, or it must restore its previous state from the retained state, or some combination thereof. When a subsystem is powered down, it must not interfere with the normal operation of the rest of the SoC.

Power Aware verification is essential to verify the operation of a design under active power management, including the power management architecture, state retention and restoration of subsystems when powered down, and the interaction of subsystems in various power states. This presentation summarizes the challenges of power aware verification and describes the use of IEEE Std 1801™-2009 UPF to define power management architecture. It also outlines the requirements and essential coverage goals for verifying a power-managed ARM-based SoC design.

## Introduction

The continual scaling of transistors and the end of voltage scaling has made power one of the critical design constraints in the design flow.

Trying to maintain performance levels and achieve faster speeds by scaling supply and threshold voltages increases the subthreshold leakage current due to its exponential relationship with the threshold voltage [1]. Leakage currents lead to power dissipation even when the circuit is not doing any useful work, which limits operation time between charges for battery-operated devices, and creates a heat dissipation problem for all devices.

Minimizing power dissipation starts with minimizing the dynamic power dissipation associated with the clock tree, by turning off the clock for subsystems that are not in use. This technique has been in use for many years. But at 90nm and below, static leakage becomes the dominant form of power dissipation. Active power management minimizes static leakage through various techniques, such as shutting off the power to unused subsystems or varying the supply voltage or threshold voltage for a given component to achieve the functionality and performance required with minimum power.

## Active Power Management

Active power management can be thought of as having three major aspects:

- the power management architecture, which involves the partitioning of the system into separately controlled power domains, and the logic required to power those domains; mediate their interactions, and control their behavior;
- the power managed behavior of the design, which involves the dynamic operation of power domains as they are powered up and down under active power management, as well as the dynamic interactions of those power domains to achieve system functionality;
- the power control logic that ultimately drives the control inputs to the power management architecture, which may be implemented in hardware or software or a combination thereof.

All three of these aspects need to be verified to ensure that the design works properly under active power management. Ideally such verification should be done at the RTL stage. This enables verification of the active power management capability much more efficiently than would be possible at the gate level, which in turn allows more time for consideration of alternative power management architectures and simplifies debugging.

## Power Management Techniques

Several power management techniques are used to minimize power dissipation: clock gating, power gating, voltage scaling, and body biasing are four of them.

Clock gating disables the clock of an unused device, to eliminate dynamic power consumption by the clock tree. Power gating uses a current switch to cut off a circuit from its power supply rails during standby mode, to eliminate static leakage when the circuit is not in use. Voltage scaling changes the voltage and clock frequency to match the performance required for a given operation so as to minimize leakage. Body biasing changes the threshold voltage to reduce leakage current at the expense of slower switching times.

Power gating is one of the most common active power management techniques. Switching off the power to a subsystem when it is not in use eliminates the leakage current in that subsystem when it is powered down, and hence the overall leakage power dissipation through that subsystem is reduced. However, this technique also results in loss of state in the subsystem when it is switched off. Also, the outputs of a power domain can float to unpredictable values when they are powered down.

Another common technique is the use of different supply voltage levels for different subsystems. A subsystem that has a higher voltage supply can change state more quickly and therefore operate with higher performance, at the expense of higher static leakage and dynamic power. A subsystem with a lower voltage supply cannot change state as quickly, and consequently operates with lower performance, but also with less static leakage and dynamic power. This technique allows designers to minimize static leakage in areas where higher performance is not required.

Multiple voltage supplies can also be used for a single subsystem, for example, by enabling it to dynamically switch between a higher voltage supply and a lower voltage supply. This allows the system to select higher performance for that subsystem when necessary, but minimize static leakage when high performance is not required. Multi-voltage and power gating techniques can be combined to give a range of power/performance options.

All of these power management techniques must be implemented in a manner that preserves the intended functionality of the design. This requires creation of power management logic to ensure that the design operates correctly as the power supplies to its various components are switched on and off or switched between voltage levels. Since this power management logic could potentially affect the functionality of the design, it is important to verify the power management logic early in the design cycle, to avoid costly respins.

## Power Management Specification

The power management architecture for a given design could be defined as part of the design, and ultimately it is a part of the design's implementation. A better approach, however, is to specify the power management architecture separate from the design.

This simplifies exploration of alternative power management architectures, reduces the likelihood of unintended changes to the golden design functionality, and maintains the reusability of the design data. This is the approach supported by IEEE Std 1801™-2009, "Standard for Design and Verification of Low Power Integrated Circuits."

This standard is also known as the Unified Power Format (UPF) version 2.0. Initially developed by Accellera, UPF is currently supported by multiple vendors and is in use worldwide [5].

UPF provides the concepts and notation required to define the power management architecture for a design. A UPF specification can be used to drive the implementation of power management for a given design, during synthesis or subsequent implementation steps. A UPF specification can also be used to drive verification of power management, during RTL simulation, gate-level simulation, or even via static verification methods. The ability to use UPF in conjunction with RTL simulation enables early verification of the power management architecture. The ability to use UPF across all of these applications eases implementation and validation by enabling reuse of power management specifications throughout the flow.

UPF syntax is defined as an extension of Tcl [8], which enables UPF descriptions to leverage all of the control features of Tcl. UPF captures the power management architecture in a portable form for use in simulation, synthesis, and routing, reducing potential omissions during translation of that intent from tool to tool. Because it is separate from the HDL description and can be read by all of the tools in the flow, the UPF side file is as portable and interoperable as the logic design's HDL code.

The concepts introduced in the following sections are illustrated with the UPF commands used to specify them.

## Power Management Architecture

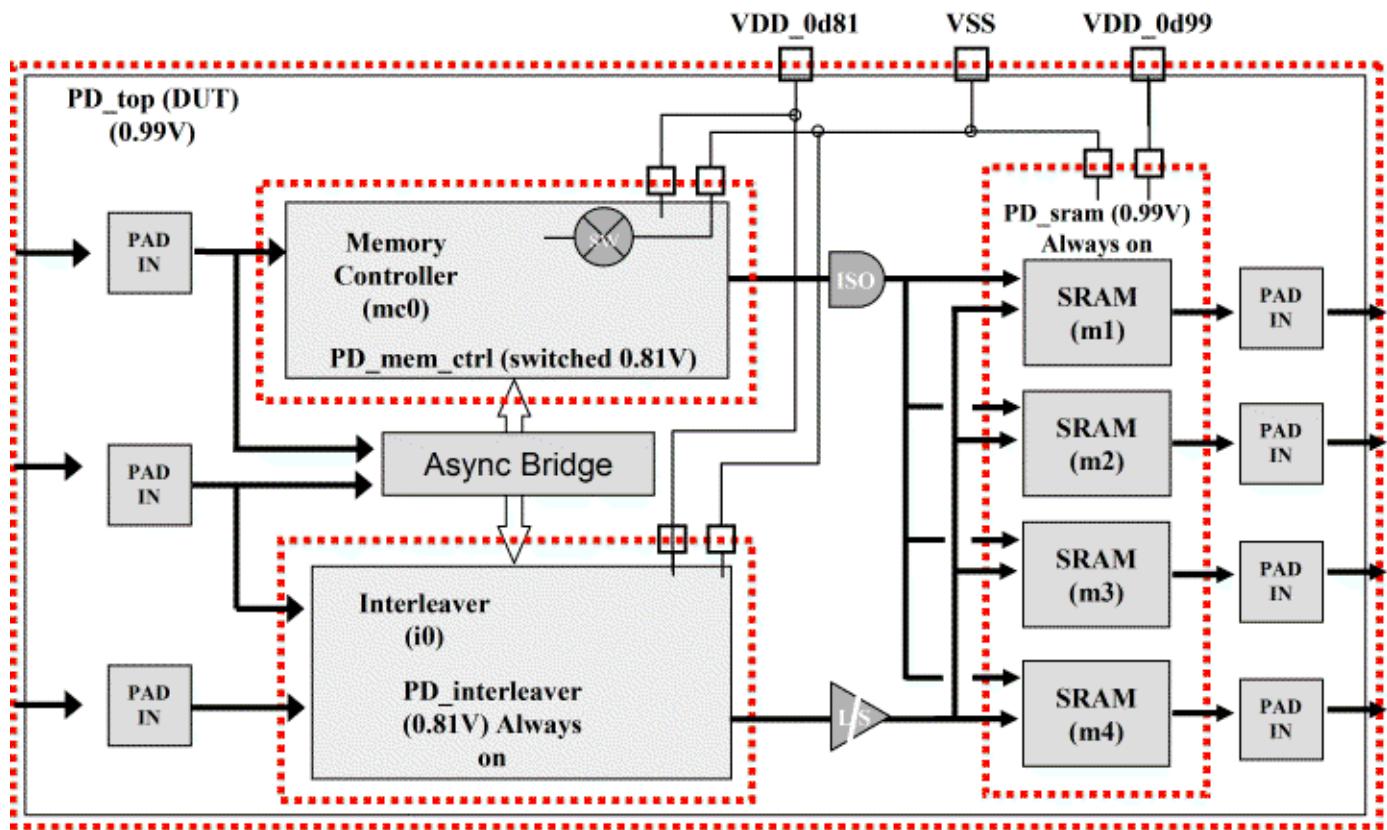
In order to employ active power management techniques such as power gating and multiple voltage supplies, the design must be partitioned into separate functional areas that can be independently powered. Additional logic must be inserted into the design to perform special functions such as power switching, state retention, isolation, and level shifting. These additional components constitute the power management architecture for a given system.

### Operating Modes

Designing the power management architecture for a given system starts with characterization of the functions and operating modes of the system. Since the goal of active power management is to optimize the use of power based on the function and performance required of the system at any given time, the first step involves identifying the distinct combinations of functionality and performance that is required of the device in use. Analysis of the set of distinct operating modes enables the designer to determine how to partition the design into independently powered subsystems or subcomponents, so that any given operating mode can be supported by providing the necessary subset of system components with the appropriate power.

For example, [Figure D-1](#) shows a block diagram of a design that has two operating modes: ON and SLEEP.

**Figure D-1. A Power-Managed Design**



In the ON mode, it reads input data streams, interleaves them, and stores them in the memory before driving them onto outputs. In the SLEEP mode, it monitors inputs and maintains the state of its memory, but it does not process inputs.

#### Power Domains

Each independently powered subsystem or subcomponent is called a power domain. At the RTL stage, a power domain is typically somewhat abstract, consisting of some or all of the RTL logic within a given portion of the design hierarchy. At the logical netlist stage, a power domain consists of a collection of cells that share the same primary power and ground supplies. At the physical level, the cells associated with a given power domain may be placed in a contiguous region of a chip or distributed over multiple discontiguous regions of the chip.

The design in [Figure D-1](#) has several major components. These include the interleaver block, the memory controller block, and the memory itself. Each of these can be defined as a separate power domain. The top-level of the design is also a separate power domain. The dotted lines in

[Figure D-1](#) indicate power domain boundaries. The following UPF commands is used to define these power domains:

```
#-----
# Create power domains
#-----
create_power_domain PD_top
create_power_domain PD_interleaver -elements {io}
create_power_domain PD_mem_ctrl -elements {mc0}
create_power_domain PD_sram -elements {m1 m2 m3 m4}
```

The `-elements` argument on each of these commands lists the instance names of the elements to be included in the specified power domain.

### Power Distribution

Each power domain may have one or more power supplies. The primary supply provides power for most of the functional elements in that domain. Additional supplies may provide power for retention, isolation, or level shifting cells associated with the power domain.

The primary supply may be a switched supply, which can be turned on and off via a control input to the switch. Either the VDD or VSS supply may be switched. A supply may be driven by multiple switches connected to the same voltage source. The switches are turned on incrementally, to minimize rush currents when the supply is switched on. A supply may also be driven by multiple switches connected to different voltage sources, so that the supply voltage level delivered to elements of the power domain may be varied. Switches may be on-chip or off-chip.

Power is distributed to power domains via supply ports interconnected by supply nets. Supply ports may represent external supplies or may be driven by internal supply sources. Supply ports are connected to supply nets, each of which is ultimately connected to a power domain. Each supply port has one or more supply states defined. The port may drive only one state at any given time. That state is propagated by the supply net connected to the port.

For the example in [Figure D-1](#), the following UPF commands could be used to define top-level supply ports, and to define and connect supply nets to those ports:

```
#-----
# Create top level power domain supply ports
#-----
create_supply_port VDD_0d99 -domain PD_top
create_supply_port VDD_0d81 -domain PD_top
create_supply_port VSS -domain PD_top

#-----
# Create top level power domain supply nets
#-----
create_supply_net VDD_0d99 -domain PD_top
create_supply_net VDD_0d81 -domain PD_top
create_supply_net VSS -domain PD_top

#-----
# Connect top level power domain supply ports
# to supply nets
#-----
connect_supply_net VDD_0d99 -ports VDD_0d99
connect_supply_net VDD_0d81 -ports VDD_0d81
connect_supply_net VSS -ports VSS
```

The following UPF command would be used to identify a particular pair of supply nets as the primary power and ground supplies for a given power domain:

```
#-----
# Set the default for top level power domain
#-----
set_domain_supply_net PD_top \
    -primary_power_net VDD_0d99 \
    -primary_ground_net VSS
```

Additional UPF commands could be used to propagate the top-level supply nets into subordinate power domains and to define a power switch to create a switched ground (VSS\_SW) for one of the power domains:

```
#-----
# Create sub domain supply nets
#-----
create_supply_net VDD_0d81 -domain PD_interleaver -reuse
create_supply_net VDD_0d81 -domain PD_mem_ctrl -reuse
create_supply_net VDD_0d99 -domain PD_sram -reuse

create_supply_net VSS -domain PD_interleaver -reuse
create_supply_net VSS -domain PD_mem_ctrl -reuse
create_supply_net VSS -domain PD_sram -reuse

#-----
# Create supply net for switch output
#-----
create_supply_net VSS_SW -domain PD_mem_ctrl

#-----
```

```
# Create power switch for memory controller domain
# - switch on ground side of supply network
#-----
create_power_switch mem_ctrl_sw \
    -domain PD_mem_ctrl \
    -output_supply_port {vout_p VSS_SW} \
    -input_supply_port {vin_p VSS} \
    -control_port {ctrl_p mc_pwr_c} \
    -on_state {normal_working vin_p {ctrl_p}} \
    -off_state {off_state {!ctrl_p}}
```

Power distribution logic may also include on-chip analog components such as regulators and sensors. A regulator takes an input supply voltage and generates a specific output voltage. A sensor monitors a supply rail and signals when the voltage has stabilized at its nominal value with respect to ground. Sensors enable construction of a feedback loop so that power control logic can determine when a power rail has completed transitioning. Analog components such as these are not specifiable in UPF, but can be modeled in HDL code using UPF package functions to model the ramp-up and ramp-down of power supplies as they switch on and off.

### Power States

UPF provides commands for defining a power state table that captures the possible power states of the system. The power state table defines system power states in terms of the states of supply ports or nets.

For the example in [Figure D-1](#), the following UPF commands define the possible states of the supply ports VDD\_0d81, VDD\_0d99, and VSS, as well as the switched ground supply VSS\_SW:

```
#-----
# Define power states
#-----
add_port_state VDD_0d99 -state {ON 0.99 1.10 1.21}
add_port_state VDD_0d99 -state {OFF off}

add_port_state VDD_0d81 -state {ON 0.81 0.90 0.99}
add_port_state VDD_0d81 -state {OFF off}

add_port_state VSS -state {ON 0 0 0}
add_port_state VSS_SW -state {ON 0} -state {OFF off}

#-----
# Create power state table
#-----
create_pst top_pst \
    -supplies { VDD_0d99 VDD_0d81 VSS VSS_SW }

add_pst_state ON \
    -pst top_pst -state { ON ON ON ON }
add_pst_state SLEEP \
    -pst top_pst -state { ON ON OFF OFF }
add_pst_state OFF \
    -pst top_pst -state { OFF OFF ON OFF }
```

The power states of the system in [Figure D-1](#) are defined in the above power state table. Note that these power states are the same as the operating modes of the system, plus the state in which the system is completely turned off.

### Isolation and Level Shifting

Even though each power domain may be independently powered on and off, their logical and physical connections to other power domains remain; therefore, when one domain is turned off, it is still connected logically and electrically to other domains. These connections between power domains require special cells to mediate the interaction between domains as their respective power states change. Two kinds of cells are involved: isolation cells, and level shifting cells.

Isolation cells ensure that signals coming from unpowered domains are clamped to a well-defined logic value while the source domain is powered down, so that any sink domain that is powered up sees reliable inputs. Depending upon the architecture of the design, and the particular characteristics of a signal that crosses from one power domain to another (for example, how many power domains it fans out to, and when those power domains are on or off with respect to the source domain), it may be appropriate to insert isolation cells at either the source of the signal or at its sink(s). However, since the isolation cell must be powered on when the source domain is powered off, isolation cells are typically powered by a separate, “always-on” supply voltage.

The following UPF commands specify the addition of isolation for the PD\_mem\_ctrl power domain in the example in [Figure D-1](#). The first command defines the supplies powering the isolation cell and specifies its clamp value. The second command defines the control signal for the isolation cell.

```
#-----
# Setup isolation strategy for memory controller
#-----

# Mem ctrl chip & write enables: clamp to '1'

set_isolation mem_ctrl_iso_1 \
    -domain PD_mem_ctrl \
    -isolation_power_net VDD_0d99 \
    -isolation_ground_net VSS \
    -clamp_value 1 \
    -elements {mc0/ceb mc0/web}

set_isolation_control mem_ctrl_iso_1 \
    -domain PD_mem_ctrl \
    -isolation_signal mc_iso_c \
    -isolation_sense high \
    -location parent
```

Level shifting cells ensure that a signal coming from a power domain operating at one voltage is correctly interpreted when it is received by a power domain operating at a different voltage. Depending upon the relative voltage levels of the two power domains, a level shifter may increase or decrease the operating voltage of the signal. As with isolation cells, level shifters

may have separate power supplies that are always on, or they may be powered by the primary supplies of the source and sink domains, respectively.

```
#-----
# Define level shifters
#-----

set_level_shifter interleaver_ls_in \
    -domain PD_interleaver \
    -applies_to inputs \
    -location self

set_level_shifter interleaver_ls_out \
    -domain PD_interleaver \
    -applies_to outputs \
    -location parent
```

The UPF commands above specify addition of level shifters for the PD\_interleaver power domain in the example in [Figure D-1](#). The first command specifies addition of level shifters for inputs; the second command specifies addition of level shifters for outputs. Whether level shifters are actually inserted or not depends upon the respective supply voltages of the source and sink domains involved.

### State Retention

When a power domain is powered down, any normal state elements within the power domain loses their state. When the power domain is powered on again, the power domain must be brought to a predictable state again. This may involve resetting all state elements in the domain, or resetting some subset that is sufficient to cause the rest of the domain to reach a well-defined state after a few clock cycles. Another alternative is to save the state of certain state elements before the domain is powered down, and restore those statements to their saved state after the power domain is powered up again.

Retention cells are special memory elements that preserve their data during power down. Such cells involve extra logic and possibly complex timing to save and restore their values across powered-down periods [2]. Various kinds of retention cells have been designed [2], [3], [4]. Some of these use balloon latch mechanisms [2], which are made up of high threshold transistors to minimize leakage through them. They are separated from the critical path of the design by transmission gates and thus are not required to be timing critical. Others depend on complex sequences of different controls to achieve data retention.

The following UPF command specifies where retention should occur in the example in [Figure D-1](#).

```
#-----
# Setup retention strategy for mem cntrl
#-----
set_retention mem_ctrl_ret \
    -domain PD_mem_ctrl \
    -retention_power_net VDD_0d81 \
    -save_signal {mc_save_c high} \
    -restore_signal {mc_restore_c low}
```

This command identifies the power domain (PD\_mem\_ctrl) within which retention registers should be used, specifies the power supply used to maintain retained values, and specifies the control signals required for saving and restoring the values of retention registers.

## Power Managed Behavior

With the power management architecture implemented on top of the design, it should be possible to repeatedly power up each power domain and later power it down again. Each time a domain is powered up, it should reach a well-defined state from which to continue its operations. That state may be the initial reset state, or a state saved before the power domain was last powered down, or a combination of the two.

While a power domain is powered down, its elements cannot drive their outputs to well-defined logic values. As a result, those outputs may float to 1 or float to 0, or may be at an intermediate value. In this situation, those outputs are considered corrupted. This is not a problem as long as no other active power domain sinks those corrupted values; otherwise logical and/or electrical problems could result. During the power down process, it is essential for the isolation cells in the design to be enabled before the domain's primary supply is shut off, for those isolation cells to clamp signals from the powered-down domain to appropriate values, and for the isolation cells to remain enabled until the shut-off domain's primary supply is turned on again.

Similarly, when two interconnected power domains have been put into respective power states that involve different supply voltages, the level shifters in the design must convert logic 1 signal voltage levels in the source domain to logic 1 signal voltage levels in the sink domain. Although level shifters function continuously and therefore do not need to be enabled, dynamic changes in the supply voltages for the respective power domains may result in unexpected situations.

## Power Control Logic

Power management may involve both software and hardware control.

For example, a power control unit (PCU) can be specified in RTL internal to the SoC. The PCU may be under software control by an embedded processor. The combination of these power management controls drive the signals that define the PCN, based on the system's power management strategy—signaling power domains to retain state, enable isolation, power down (turn off switches), power up (turn on switches), disable isolation, and restore state.

Correct operation of the power management architecture depends upon correct sequencing of power control signals. For example, outputs of a domain must be isolated before the power is shut off, and must remain isolated until after power is turned on again. Thus the control signal initiating isolation must come before the control signal that turns off the power switch. Similarly, the control signal that turns on the power switch must occur before the control signal that terminates isolation. In fact, turning power on and off may involve handshaking between the PCU and the supply source (or a sensor monitoring the supply) to ensure that voltage-dependent delays in ramping up or down the power supply are factored into control signal sequencing.

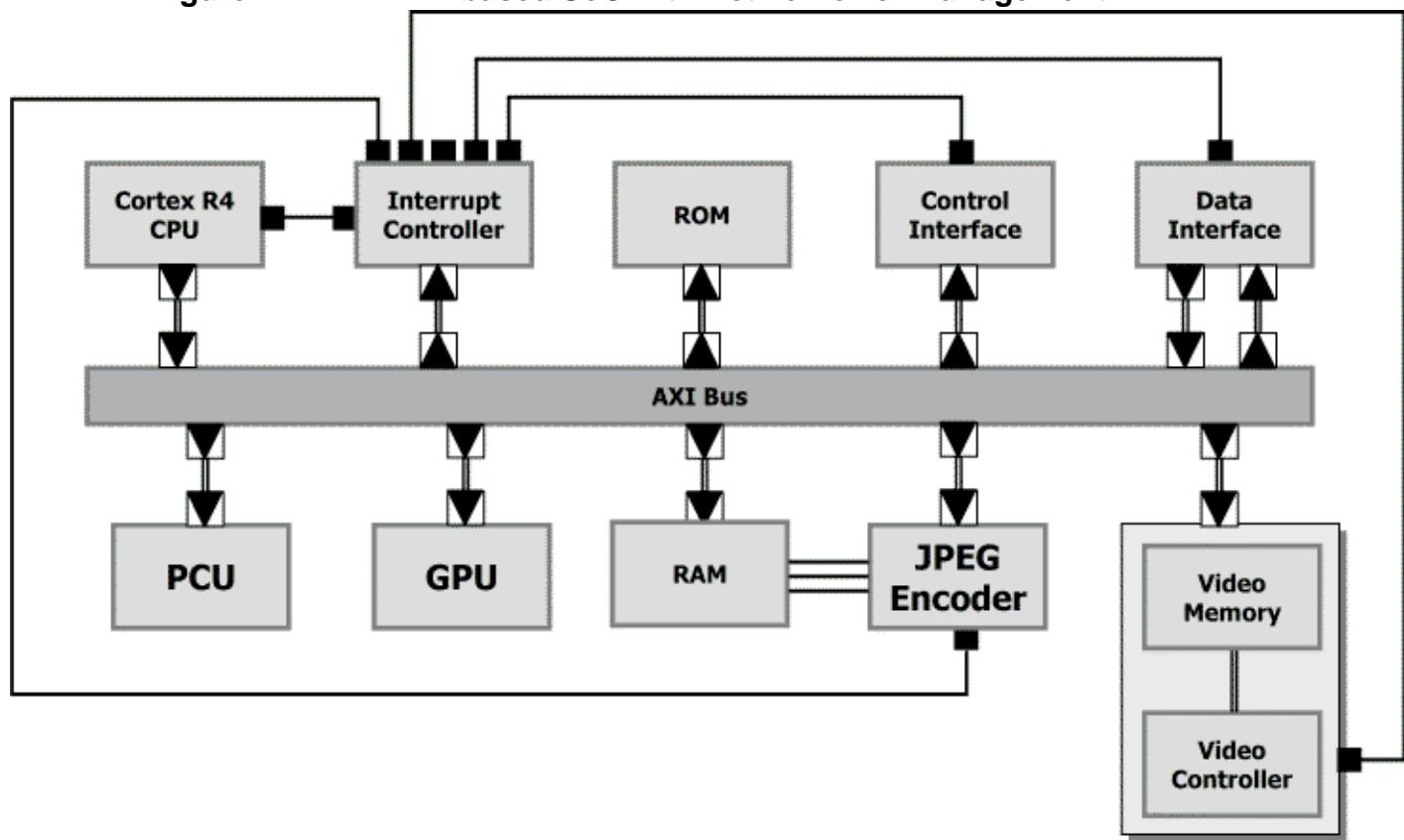
## Power Aware Verification Flow

Verifying RTL-level specification of active power management for a given design involves several steps.

First, you need to verify that the power management architecture is correctly structured, given the operating modes of the device and the power states that have been defined to reflect those modes. Second, you need to verify that the design (both each power domain individually and all of them collectively) behave correctly when power management control signals are given in the correct sequence. Third, you need to verify that the power control logic always generate power control signals in the correct sequence.

[Figure D-2](#) shows the high-level design of an ARM-based SoC with active power management. The above verification steps as applied to this example are described below.

**Figure D-2.** An ARM-based SoC with Active Power Management



This design consists of multiple functional units communicating over the AXI bus. Each functional unit may be defined as a separate power domain, or even as a collection of power domains. A UPF file for this design would specify the power management architecture for the whole system, including the specific requirements for power distribution, switching, and state retention for each power domain, and the requirements for isolation and level shifting between interacting power domains. The Power Control Unit (PCU) is a hardware implementation of power control logic that drives power control signals for each domain in the correct order. The Cortex R4 CPU is an embedded ARM processor that drives system-level power state changes by sending transactions to the PCU.

# Verifying the Power Management Architecture

Verifying the sufficiency of the power management architecture can be done in part through static analysis. Given a complete definition of the power domains and power states for a given design, it is relatively straightforward to verify that the necessary isolation cells and level shifters are present (or implied by a UPF specification) to ensure that the power domains interact correctly and is not adversely affected when their neighboring power domains are powered down. Static analysis can also ensure that the necessary supply structures are present to provide the ability to control power to each power domain.

However, static analysis is not always possible. Depending upon the sequencing of power state changes, and the ramping of power supplies as they transition, there may be a requirement for level shifters that is not obvious from the power state table. Also, the power state table may not be complete, and power states that are not defined might actually occur during operation of the device. Finally, external supply sources may be switched or may vary in voltage beyond what is defined in the power state table. For these and other reasons, simulation is often required. In this case, Power Aware simulation is necessary, to ensure that the power management architecture and its controls are taken into account during simulation.

Power Aware simulation enables functional verification of power management in the context of an RTL design. Power Aware simulation run does the following:

- Compiles the design and UPF specifications
- Infers sequential elements from the RTL design (registers, latches and memories)
- Applies the UPF-specified power management architecture to the RTL design
- Augments the simulation model with appropriate power aware models
- Dynamically modifies the RTL behavior to reflect the impact of active power management.

Using the UPF and sequential element information, the simulator is able to augment the normal RTL behavior with the UPF-specified power aware behavior (power distribution and control, retention, corruption, and isolation). This involves selecting the appropriate simulation models to implement the UPF-specified power management architecture. It may also involve recognizing and integrating user-supplied Power Aware simulation models.

### Verifying Power Managed Behavior

Power Aware simulation can be used to visualize the effects of active power management on the dynamic behavior of the design, as well as visualizing the behavior of the power management architecture itself under control of power management logic. In a Power Aware simulation, the internal state and outputs of a power domain is set to X to reflect the corruption of those signals when the primary supply to that domain is turned off. When the supply is turned on again, the X values are replaced as the power domain reinitializes or has its state restored. Signals driven by outputs of a powered-down domain should be clamped to 0 or 1, so that downstream power domains see a well-defined value and will not be affected by corrupted outputs of a power domain that has been powered down. Retention should be evident in that the state of signals following power up corresponds to the state of signals prior to the previous power down.

Visualizing the effects of active power management helps the designer confirm that all of the necessary power domains and power states required to implement the operation modes of this device have been defined, and that all the necessary isolation, level-shifting, and retention cells necessary to enable power management have been added. If there are errors in the power management architecture, they are very likely cause signal corruption that does not go away after power up, which in turn leads to functional errors in the design.

Debugging power management errors can be performed by tracking corruption of signals in the waveform view, but that method is tedious and error-prone. A much more effective method is the use of assertions to check for correct operation of the design under active power management. For example, an assertion to check that an output of a power domain is clamped to the correct value when the power domain is powered down, immediately catches any error related to the clamp value, or the powering of the isolation cell involved, rather than just generating an X and letting it propagate. Such assertions can be automatically generated by the Power Aware simulator.

### Verifying Power Control Logic

Power Aware simulation can also be used to verify the control logic driving the power management architecture, provided that the control logic is part of the design rather than being implemented in a test bench. For software-based power control logic, simulation is the only method available. In particular, hardware/software co-simulation is necessary if the power control logic is split between hardware and software components, as is often the case. For hardware-based power control logic, such as a power control unit, another alternative is available.

Formal verification is particularly suited to verifying complex control logic. In contrast to simulation, which runs one input sequence at a time to test a device, formal verification considers all valid input sequences in one pass. A formal verification tool can therefore identify all possible behaviors of the power control logic, which enables it to automatically find any corner cases in which the generated control sequences may not be complete or in the correct order. Formal verification is driven by assertions, so use of formal verification requires creation of assertions about the expected behavior of the power control unit. Although this takes some effort, the ability to thoroughly verify the power control logic makes it worthwhile.

## Summary

Active power management is becoming a necessary part of today's SoC designs. To add active power management to a design and verify that it works correctly, it is critical to have a well-defined methodology that addresses all aspects of active power management. The methodology needs to support defining and verifying an appropriate power management architecture, verifying that the design behaves correctly under the power management architecture, and verifying that the power control signals controlling the power management architecture are generated correctly. IEEE Std 1801™-2009 UPF supports such a methodology, as does static analysis of power management architecture, Power Aware simulation of power-managed designs, and formal verification of power control logic. These methods provide a comprehensive solution for defining and verifying active power management.

## Acknowledgments

The authors would like to acknowledge the thoughtful commentary and suggestions for improvement provided by Barry Pangrle on the penultimate draft of this paper.

## References

1. N.S. Kim, T. Austin, T. Blaauw, T. Mudge, K. Flautner, H.S. Hu, M.J.Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *IEEE Computer*, 36(12):68--75, 2003.
2. S. Shigematsu, S. Mutoh, Y. Matsuya, Y. Tanabe and J. Yamada, "A 1-V High-Speed MTCMOS Circuit Scheme for Power-Down Application Circuits," *IEEE J. Solid-State Circuits*, Vol. 32, No. 6, pp. 861--869, 1997.
3. Hyo-Sig Won; Kyo-Sun Kim; Kwang-Ok Jeong; Ki-Tae Park; Kyu-Myung Choi; Jeong-Taek Kong, "An MTCMOS design methodology and its application to mobile computing," *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on*, vol., no., pp. 110-115, 25-27 Aug. 2003.
4. Zyuban, V.; Kosonocky, S.V., "Low power integrated scan-retention mechanism," *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium on*, vol., no., pp. 98-102, 2002.
5. IEEE 1801™-2009, "Standard for Design and Verification of Low Power Integrated Circuits", IEEE.
6. IEEE 1801™-2013, "Standard for Design and Verification of Low Power Integrated Circuits", IEEE.
7. IEEE 1801™-2015, "Standard for Design and Verification of Low Power Integrated Circuits", IEEE.
8. Tcl/Tk Documentation, Tcl Developer Xchange, <http://www.tcl.tk>.



# Index

---

## — D —

Design flow, [16](#)

## — H —

Hard macro, [341](#)

## — L —

Liberty libraries, [52](#)

Low power, [15](#)

## — M —

Macro, hard, [341](#)

## — N —

Named events, [349](#)

## — P —

PA-GL, [15](#)

PA-RTL, [15](#)

Power Aware, [15](#)

    documentation, [18](#)

Power gating, [15](#)

Power State Table (PST), [332](#), [380](#)

PST, [332](#), [380](#)

## — U —

Unified Power Format (UPF), [357](#)

UPF, [357](#)

## — V —

Value conversion table (VCT), [79](#)

VCT, [79](#)

Verilog

    named events, [349](#)



# End-User License Agreement

The latest version of the End-User License Agreement is available on-line at:  
[www.mentor.com/eula](http://www.mentor.com/eula)

## IMPORTANT INFORMATION

**USE OF ALL SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF SOFTWARE INDICATES CUSTOMER'S COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.**

## END-USER LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Software (as defined in Section 2) and hardware (collectively "Products") between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If Customer does not agree to these terms and conditions, promptly return or, in the case of Software received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

### 1. ORDERS, FEES AND PAYMENT.

- 1.1. To the extent Customer (or if agreed by Mentor Graphics, Customer's appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement (each an "Order"), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement, any applicable addenda and the applicable quotation, whether or not those documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order or presented in any electronic portal or automated order management system, whether or not required to be electronically accepted, will not be effective unless agreed in writing and physically signed by an authorized representative of Customer and Mentor Graphics.
- 1.2. Amounts invoiced will be paid, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice. Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax, consumption tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer's sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer's behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.
- 1.3. All Products are delivered FCA factory (Incoterms 2010), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics retains a security interest in all Products delivered under this Agreement, to secure payment of the purchase price of such Products, and Customer agrees to sign any documents that Mentor Graphics determines to be necessary or convenient for use in filing or perfecting such security interest. Mentor Graphics' delivery of Software by electronic means is subject to Customer's provision of both a primary and an alternate e-mail address.

2. **GRANT OF LICENSE.** The software installed, downloaded, or otherwise acquired by Customer under this Agreement, including any updates, modifications, revisions, copies, documentation, setup files and design data ("Software") are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors, who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Except for Software that is embeddable ("Embedded Software"), which is licensed pursuant to separate embedded software terms or an embedded software supplement, Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form (except as provided in Subsection 4.2); (b) for Customer's internal business purposes; (c) for the term of the license; and (d) on the computer hardware and at the site authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Customer may have Software temporarily used by an employee for telecommuting purposes from locations other than a Customer office, such as the employee's residence, an airport or hotel, provided that such employee's primary place of employment is the site where the Software is authorized for use. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or services purchased, apply to the following: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be technically implemented through the use of authorization codes or similar devices); and (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions. For the avoidance of doubt, if Customer provides any feedback or requests any change or enhancement to Products, whether in the course of receiving support or consulting services, evaluating Products, performing beta testing or otherwise, any inventions, product improvements, modifications or developments made by Mentor Graphics (at Mentor Graphics' sole discretion) will be the exclusive property of Mentor Graphics.

### **3. BETA CODE.**

- 3.1. Portions or all of certain Software may contain code for experimental testing and evaluation (which may be either alpha or beta, collectively “Beta Code”), which may not be used without Mentor Graphics’ explicit authorization. Upon Mentor Graphics’ authorization, Mentor Graphics grants to Customer a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. Mentor Graphics may choose, at its sole discretion, not to release Beta Code commercially in any form.
- 3.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer’s use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer’s evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.
- 3.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on Customer’s feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this Subsection 3.3 shall survive termination of this Agreement.

### **4. RESTRICTIONS ON USE.**

- 4.1. Customer may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. Except for Embedded Software that has been embedded in executable code form in Customer’s product(s), Customer shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. Customer shall not make Products available in any form to any person other than Customer’s employees and on-site contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use Products except as permitted by this Agreement. Customer shall give Mentor Graphics written notice of any unauthorized disclosure or use of the Products as soon as Customer becomes aware of such unauthorized disclosure or use. Customer acknowledges that Software provided hereunder may contain source code which is proprietary and its confidentiality is of the highest importance and value to Mentor Graphics. Customer acknowledges that Mentor Graphics may be seriously harmed if such source code is disclosed in violation of this Agreement. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, disassemble, reverse-compile, or reverse-engineer any Product, or in any way derive any source code from Software that is not provided to Customer in source code form. Log files, data files, rule files and script files generated by or for the Software (collectively “Files”), including without limitation files containing Standard Verification Rule Format (“SVRF”) and Tcl Verification Format (“TVF”) which are Mentor Graphics’ trade secret and proprietary syntaxes for expressing process rules, constitute or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Customer may use Files containing SVRF or TVF only with Mentor Graphics products. Under no circumstances shall Customer use Products or Files or allow their use for the purpose of developing, enhancing or marketing any product that is in any way competitive with Products, or disclose to any third party the results of, or information pertaining to, any benchmark.
  - 4.2. If any Software or portions thereof are provided in source code form, Customer will use the source code only to correct software errors and enhance or modify the Software for the authorized use, or as permitted for Embedded Software under separate embedded software terms or an embedded software supplement. Customer shall not disclose or permit disclosure of source code, in whole or in part, including any of its methods or concepts, to anyone except Customer’s employees or on-site contractors, excluding Mentor Graphics competitors, with a need to know. Customer shall not copy or compile source code in any manner except to support this authorized use.
  - 4.3. Customer agrees that it will not subject any Product to any open source software (“OSS”) license that conflicts with this Agreement or that does not otherwise apply to such Product.
  - 4.4. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense, or otherwise transfer the Products, whether by operation of law or otherwise (“Attempted Transfer”), without Mentor Graphics’ prior written consent and payment of Mentor Graphics’ then-current applicable relocation and/or transfer fees. Any Attempted Transfer without Mentor Graphics’ prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics’ option, result in the immediate termination of the Agreement and/or the licenses granted under this Agreement. The terms of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer’s permitted successors in interest and assigns.
  - 4.5. The provisions of this Section 4 shall survive the termination of this Agreement.
5. **SUPPORT SERVICES.** To the extent Customer purchases support services, Mentor Graphics will provide Customer with updates and technical support for the Products, at the Customer site(s) for which support is purchased, in accordance with Mentor Graphics’ then current End-User Support Terms located at <http://supportnet.mentor.com/supportterms>.
  6. **OPEN SOURCE SOFTWARE.** Products may contain OSS or code distributed under a proprietary third party license agreement, to which additional rights or obligations (“Third Party Terms”) may apply. Please see the applicable Product documentation (including license files, header files, read-me files or source code) for details. In the event of conflict between the terms of this Agreement

(including any addenda) and the Third Party Terms, the Third Party Terms will control solely with respect to the OSS or third party code. The provisions of this Section 6 shall survive the termination of this Agreement.

## 7. LIMITED WARRANTY.

- 7.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Products will meet Customer's requirements or that operation of Products will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. Customer must notify Mentor Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Software under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes or alternate Software under a transaction involving Software re-mix. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification, improper installation or Customer is not in compliance with this Agreement. MENTOR GRAPHICS' ENTIRE LIABILITY AND CUSTOMER'S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) PRODUCTS PROVIDED AT NO CHARGE; OR (C) BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."
  - 7.2. THE WARRANTIES SET FORTH IN THIS SECTION 7 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO PRODUCTS PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.
8. **LIMITATION OF LIABILITY.** TO THE EXTENT PERMITTED UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT RECEIVED FROM CUSTOMER FOR THE HARDWARE, SOFTWARE LICENSE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 8 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

## 9. THIRD PARTY CLAIMS.

- 9.1. Customer acknowledges that Mentor Graphics has no control over the testing of Customer's products, or the specific applications and use of Products. Mentor Graphics and its licensors shall not be liable for any claim or demand made against Customer by any third party, except to the extent such claim is covered under Section 10.
- 9.2. In the event that a third party makes a claim against Mentor Graphics arising out of the use of Customer's products, Mentor Graphics will give Customer prompt notice of such claim. At Customer's option and expense, Customer may take sole control of the defense and any settlement of such claim. Customer WILL reimburse and hold harmless Mentor Graphics for any LIABILITY, damages, settlement amounts, costs and expenses, including reasonable attorney's fees, incurred by or awarded against Mentor Graphics or its licensors in connection with such claims.
- 9.3. The provisions of this Section 9 shall survive any expiration or termination of this Agreement.

## 10. INFRINGEMENT.

- 10.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay costs and damages finally awarded against Customer that are attributable to such action. Customer understands and agrees that as conditions to Mentor Graphics' obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.
- 10.2. If a claim is made under Subsection 10.1 Mentor Graphics may, at its option and expense: (a) replace or modify the Product so that it becomes noninfringing; (b) procure for Customer the right to continue using the Product; or (c) require the return of the Product and refund to Customer any purchase price or license fee paid, less a reasonable allowance for use.
- 10.3. Mentor Graphics has no liability to Customer if the action is based upon: (a) the combination of Software or hardware with any product not furnished by Mentor Graphics; (b) the modification of the Product other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of the Product as part of an infringing process; (e) a product that Customer makes, uses, or sells; (f) any Beta Code or Product provided at no charge; (g) any software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; (h) OSS, except to the extent that the infringement is directly caused by Mentor Graphics' modifications to such OSS; or (i) infringement by Customer that is deemed willful. In the case of (i), Customer shall reimburse Mentor Graphics for its reasonable attorney fees and other costs related to the action.
- 10.4. THIS SECTION 10 IS SUBJECT TO SECTION 8 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS, AND CUSTOMER'S SOLE AND EXCLUSIVE REMEDY, FOR DEFENSE,

SETTLEMENT AND DAMAGES, WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY PRODUCT PROVIDED UNDER THIS AGREEMENT.

**11. TERMINATION AND EFFECT OF TERMINATION.**

- 11.1. If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized term. Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement immediately upon written notice if Customer: (a) exceeds the scope of the license or otherwise fails to comply with the licensing or confidentiality provisions of this Agreement, or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. For any other material breach of any provision of this Agreement, Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement upon 30 days written notice if Customer fails to cure the breach within the 30 day notice period. Termination of this Agreement or any license granted hereunder will not affect Customer's obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination.
- 11.2. Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination of this Agreement and/or any license granted under this Agreement, Customer shall ensure that all use of the affected Products ceases, and shall return hardware and either return to Mentor Graphics or destroy Software in Customer's possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form.
12. **EXPORT.** The Products provided hereunder are subject to regulation by local laws and European Union ("E.U.") and United States ("U.S.") government agencies, which prohibit export, re-export or diversion of certain products, information about the products, and direct or indirect products thereof, to certain countries and certain persons. Customer agrees that it will not export or re-export Products in any manner without first obtaining all necessary approval from appropriate local, E.U. and U.S. government agencies. If Customer wishes to disclose any information to Mentor Graphics that is subject to any E.U., U.S. or other applicable export restrictions, including without limitation the U.S. International Traffic in Arms Regulations (ITAR) or special controls under the Export Administration Regulations (EAR), Customer will notify Mentor Graphics personnel, in advance of each instance of disclosure, that such information is subject to such export restrictions.
13. **U.S. GOVERNMENT LICENSE RIGHTS.** Software was developed entirely at private expense. The parties agree that all Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to U.S. FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. government or a U.S. government subcontractor is subject solely to the terms and conditions set forth in this Agreement, which shall supersede any conflicting terms or conditions in any government order document, except for provisions which are contrary to applicable mandatory federal laws.
14. **THIRD PARTY BENEFICIARY.** Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, Microsoft Corporation and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.
15. **REVIEW OF LICENSE USAGE.** Customer will monitor the access to and use of Software. With prior written notice and during Customer's normal business hours, Mentor Graphics may engage an internationally recognized accounting firm to review Customer's software monitoring system and records deemed relevant by the internationally recognized accounting firm to confirm Customer's compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FlexNet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics' request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement. The provisions of this Section 15 shall survive the termination of this Agreement.
16. **CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION.** The owners of certain Mentor Graphics intellectual property licensed under this Agreement are located in Ireland and the U.S. To promote consistency around the world, disputes shall be resolved as follows: excluding conflict of laws rules, this Agreement shall be governed by and construed under the laws of the State of Oregon, U.S., if Customer is located in North or South America, and the laws of Ireland if Customer is located outside of North or South America or Japan, and the laws of Japan if Customer is located in Japan. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the courts of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply, or the Tokyo District Court when the laws of Japan apply. Notwithstanding the foregoing, all disputes in Asia (excluding Japan) arising out of or in relation to this Agreement shall be resolved by arbitration in Singapore before a single arbitrator to be appointed by the chairman of the Singapore International Arbitration Centre ("SIAC") to be conducted in the English language, in accordance with the Arbitration Rules of the SIAC in effect at the time of the dispute, which rules are deemed to be incorporated by reference in this section. Nothing in this section shall restrict Mentor Graphics' right to bring an action (including for example a motion for injunctive relief) against Customer in the jurisdiction where Customer's place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.
17. **SEVERABILITY.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.
18. **MISCELLANEOUS.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements. Any translation of this Agreement is provided to comply with local legal requirements only. In the event of a dispute between the English and any non-English versions, the English version of this Agreement shall govern to the extent not prohibited by local law in the applicable jurisdiction. This Agreement may only be modified in writing, signed by an authorized representative of each party. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.