

# **Power Aware Simulation User's Manual**

Software Version 2020.4

---

Unpublished work. © Siemens 2020

This document contains information that is confidential and proprietary to Mentor Graphics Corporation, Siemens Industry Software Inc., or their affiliates (collectively, "Siemens"). 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 confidential and proprietary information.

This document is for information and instruction purposes. Siemens 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 Siemens to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Siemens products are set forth in written agreements between Siemens and its customers. **End User License Agreement** — You can print a copy of the End User License Agreement from: [mentor.com/eula](http://mentor.com/eula).

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 Siemens whatsoever.

SIEMENS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

SIEMENS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, LOST DATA OR PROFITS, EVEN IF SUCH DAMAGES WERE FORESEEABLE, ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF SIEMENS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**LICENSE RIGHTS APPLICABLE TO THE U.S. GOVERNMENT:** This document explains the capabilities of commercial products that were developed exclusively at private expense. If the products are acquired directly or indirectly for use by the U.S. Government, then the parties agree that the products and this document are considered "Commercial Items" and "Commercial Computer Software" or "Computer Software Documentation," as defined in 48 C.F.R. §2.101 and 48 C.F.R. §252.227-7014(a)(1) and (a)(5), as applicable. Software and this document may only be used under the terms and conditions of the End User License Agreement referenced above as required by 48 C.F.R. §12.212 and 48 C.F.R. §227.7202. The U.S. Government will only have the rights set forth in the End User License Agreement, which supersedes any conflicting terms or conditions in any government order document, 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 Siemens or other parties. No one is permitted to use these Marks without the prior written consent of Siemens or the owner of the Marks, as applicable. The use herein of third party Marks is not an attempt to indicate Siemens as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A list of Siemens' trademarks may be viewed at: [www.plm.automation.siemens.com/global/en/legal/trademarks.html](http://www.plm.automation.siemens.com/global/en/legal/trademarks.html) and [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.

Support Center: [support.sw.siemens.com](http://support.sw.siemens.com)

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

# Table of Contents

---

## Chapter 1

|   |           |
|---|-----------|
| <b>Introduction to Power Aware Simulation</b> ..... | <b>15</b> |
| Power Aware Simulation Overview .....               | 15        |
| Power Aware Simulation in Your Design Flow .....    | 16        |
| Documentation Scope .....                           | 18        |

## Chapter 2

|  |           |
|--|-----------|
| <b>Getting Started With Power Aware Simulation</b> ..... | <b>19</b> |
| Power Aware Simulation Inputs .....                      | 19        |
| Power Aware Simulation Commands .....                    | 19        |
| Power Aware Simulation Flows .....                       | 21        |
| Using the Three-Step Flow .....                          | 21        |
| Using the PDU-Based Simulation Flow .....                | 23        |
| Using the Power Aware Interactive Mode .....             | 24        |
| Using the Two-Step Flow .....                            | 25        |

## Chapter 3

|   |           |
|---|-----------|
| <b>Power Aware Simulation Behavior</b> .....                      | <b>27</b> |
| Power Aware Default Settings .....                                | 27        |
| Power Aware Isolation Support .....                               | 40        |
| Power Aware Level Shifter Support .....                           | 41        |
| Power Aware Retention Support .....                               | 43        |
| Power Aware Retention Support Overview .....                      | 43        |
| Retention Model Selection .....                                   | 43        |
| Level Sensitive Retention Model Example .....                     | 44        |
| Isolation and Level Shifting Behavior on a Port .....             | 46        |
| Examples of Isolation on a Port .....                             | 47        |
| Power States .....  | 52        |
| Power State Tables .....  | 52        |
| Cross-PST Analysis .....  | 53        |
| Power State Composition .....                                     | 54        |
| State Dependency Determination With add_power_state Options ..... | 56        |
| X Propagation and Drivers .....                                   | 57        |
| Checker Module and the bind_checker Command .....                 | 59        |
| Reasons For Disabling the Simulation Semantics .....              | 63        |
| Assertion Control .....   | 63        |
| Value Conversion Tables .....                                     | 65        |
| Questa-Specific Value Conversion Tables .....                     | 65        |
| Path-Based Semantics for Power Aware Strategies .....             | 67        |
| Precedence Resolution for Power Aware Strategies .....            | 77        |
| Specifying Files Using the Vopt Command .....                     | 81        |

**Chapter 4**

|   |           |
|---|-----------|
| <b>Power Aware Checks</b>                                     | <b>83</b> |
| Power Aware Checks Overview                                   | 84        |
| Isolation Checks  | 85        |
| Level Shifter Checks  | 86        |
| Path Analysis Checks  | 87        |
| Retention Checks  | 87        |
| Back-to-Back Checks   | 87        |
| Power Aware Checks Command Reference                          | 89        |
| Static RTL Checks   | 90        |
| Static RTL Isolation Checks                                   | 90        |
| Static RTL Level Shifter Checks                               | 92        |
| Static RTL Path Analysis Checks                               | 94        |
| Static RTL Back-to-Back Checks                                | 95        |
| Static GLS Checks   | 97        |
| Static GLS Isolation Checks                                   | 98        |
| Static GLS Level Shifter Checks                               | 101       |
| Static GLS Retention Checks                                   | 104       |
| Static GLS Back-to-Back Checks                                | 104       |
| Static GLS ELS Checks   | 105       |
| Static GLS Switch Checks                                      | 106       |
| Static GLS Miscellaneous Checks                               | 106       |
| Dynamic Checks  | 108       |
| Dynamic Isolation Checks                                      | 108       |
| Dynamic Level Shifter Checks                                  | 111       |
| Dynamic Retention Checks                                      | 112       |
| Dynamic Miscellaneous Checks                                  | 113       |
| Quick Reference of Static RTL and Dynamic Checks              | 115       |
| Power Aware Checks Control                                    | 118       |
| Controlling Checks With the pa_checks Command                 | 119       |
| Controlling Checks With the pa msg Command                    | 121       |
| Precedence Order of Arguments                                 | 123       |
| Pathname Convention in Arguments                              | 125       |
| Power Aware Static Check Display                              | 126       |
| Viewing Static Checks in the Results Analysis Window          | 126       |
| GUI Elements of the Results Analysis Window for Static Checks | 127       |

**Chapter 5**

|   |            |
|---|------------|
| <b>Power Aware Reports and Messages</b> | <b>131</b> |
| Power Aware Reports                     | 132        |
| Generating Power Aware Reports          | 135        |
| Power Aware Messages                    | 136        |

**Chapter 6**

|  |            |
|--|------------|
| <b>Power Aware Coverage</b>                              | <b>137</b> |
| Power Aware Coverage Overview                            | 137        |
| Power Aware Coverage Flow                                | 140        |
| Generating Coverage Reports Using Power Aware Simulation | 141        |

## Table of Contents

---

|   |            |
|---|------------|
| Generating Coverage Reports Using Non-Power Aware Simulation. . . . . | 143        |
| Analyzing Coverage Using the Power Aware Test Plan . . . . .          | 145        |
| Coverage Support of UPF Objects. . . . .                              | 149        |
| Excluding Objects From Power Aware Coverage. . . . .                  | 152        |
| <br><b>Chapter 7</b>  |            |
| <b>Power Aware Information Model . . . . .</b>                        | <b>155</b> |
| Tcl Interface. . . . .  | 156        |
| Running the Tcl APIs . . . . .  | 156        |
| Supported Tcl APIs. . . . .   | 157        |
| upf_object_in_class . . . . .   | 157        |
| upf_query_object_pathname . . . . .                                   | 157        |
| upf_query_object_properties . . . . .                                 | 158        |
| upf_query_object_type. . . . .  | 158        |
| HDL Interface . . . . .   | 160        |
| Supported HDL Package Functions . . . . .                             | 160        |
| Supported Native HDL Representation . . . . .                         | 162        |
| Power Aware Coverage Using HDL Package Functions . . . . .            | 163        |
| Power Aware Assertions Using HDL Package Functions . . . . .          | 165        |
| <br><b>Chapter 8</b>  |            |
| <b>Power Aware Simulation Display in Questa TKGUI . . . . .</b>       | <b>167</b> |
| UPF Object Display . . . . .  | 168        |
| UPF Objects . . . . .   | 169        |
| Displaying UPF Objects in the GUI . . . . .                           | 169        |
| Power Aware Source Window. . . . .                                    | 171        |
| UPF Color Scheme in the Source Window . . . . .                       | 172        |
| UPF-Specific Context Menu in the Source Window . . . . .              | 173        |
| Show Drivers Control Bar . . . . .                                    | 174        |
| Power Aware Schematic Display . . . . .                               | 175        |
| Top-Down Debugging (From the Test Bench). . . . .                     | 175        |
| Bottom-Up Debugging (From the DUT) . . . . .                          | 176        |
| Schematic Window Features for Debugging. . . . .                      | 177        |
| Power Aware Waveform Display . . . . .                                | 180        |
| Power Aware Waveform Concepts. . . . .                                | 180        |
| Using Power Aware Highlighting. . . . .                               | 181        |
| Power State and Transition Display. . . . .                           | 182        |
| Power State and Transition Concepts . . . . .                         | 182        |
| Displaying Power Aware State Machines . . . . .                       | 183        |
| Power Aware State Machine List Window. . . . .                        | 185        |
| Power Aware State Machine Viewer Window . . . . .                     | 187        |
| <br><b>Chapter 9</b>  |            |
| <b>Power Aware Mixed RTL and Gate-Level Simulation . . . . .</b>      | <b>191</b> |
| Hard Macro Model. . . . .   | 193        |
| Hard Macro Model Overview. . . . .                                    | 194        |
| Non-Power Aware HDL Model . . . . .                                   | 195        |
| Power Aware HDL Model . . . . .                                       | 196        |

|   |            |
|---|------------|
| Extended Power Aware HDL Model .....                        | 197        |
| Liberty Model of a Hard Macro .....                         | 197        |
| Specifying Power Intent of a Hard Macro .....               | 200        |
| Power Management Cell .....                                 | 203        |
| Gate-Level Cell .....                                       | 214        |
| Sequential UDP .....  | 215        |
| Liberty Model .....   | 217        |
| Liberty Model Overview .....                                | 217        |
| Specifying Liberty to a Power Aware Simulation .....        | 218        |
| Liberty Attribute Library Management .....                  | 219        |
| Creating a Liberty Attribute Library .....                  | 219        |
| Updating a Liberty Attribute Library .....                  | 219        |
| Refreshing a Liberty Attribute Library .....                | 220        |
| <b>Chapter 10</b>   |            |
| <b>Power Aware Simulation Debug .....</b>                   | <b>221</b> |
| Debugging With Questa TKGUI .....                           | 222        |
| Creating and Loading a Debug Database in Questa TKGUI ..... | 222        |
| Power Aware Debug Support in Questa TKGUI .....             | 222        |
| Debugging With Visualizer Debug Environment .....           | 225        |
| Creating and Loading a Debug Database in Visualizer .....   | 225        |
| Power Aware Debug Support in Visualizer .....               | 227        |
| Debugging With Power Aware Apps .....                       | 230        |
| Power Aware Debugging Apps Overview .....                   | 230        |
| Running the Power Aware Apps .....                          | 230        |
| Finding the Source of a Corrupt Signal .....                | 231        |
| Reporting Selected Power Aware Information .....            | 233        |
| <b>Appendix A</b>   |            |
| <b>UPF and Tcl Commands .....</b>                           | <b>235</b> |
| UPF Commands and Reference .....                            | 236        |
| Supported UPF Versions .....                                | 236        |
| Supported UPF Commands .....                                | 238        |
| add_domain_elements .....                                   | 241        |
| add_port_state .....  | 241        |
| add_power_state .....                                       | 242        |
| add_pst_state .....   | 243        |
| add_state_transition .....                                  | 243        |
| add_supply_state .....                                      | 244        |
| apply_power_model .....                                     | 244        |
| associate_supply_set .....                                  | 245        |
| begin_power_model .....                                     | 245        |
| bind_checker .....  | 246        |
| connect_logic_net .....                                     | 247        |
| connect_supply_net .....                                    | 247        |
| connect_supply_set .....                                    | 248        |
| create_composite_domain .....                               | 249        |
| create_hdl2upf_vct .....                                    | 253        |

## Table of Contents

---

|                                  |     |
|----------------------------------|-----|
| create_logic_net .....           | 253 |
| create_logic_port .....          | 254 |
| create_power_domain .....        | 254 |
| create_power_state_group .....   | 255 |
| create_power_switch .....        | 255 |
| create_pst .....                 | 257 |
| create_supply_net .....          | 257 |
| create_supply_port .....         | 260 |
| create_supply_set .....          | 260 |
| create_upf2hdl_vct .....         | 261 |
| define_power_model .....         | 261 |
| describe_state_transition .....  | 262 |
| end_power_model .....            | 262 |
| load_simstate_behavior .....     | 263 |
| load_upf .....                   | 263 |
| load_upf_protected .....         | 264 |
| map_isolation_cell .....         | 265 |
| map_level_shifter_cell .....     | 266 |
| map_power_switch .....           | 267 |
| map_retention_cell .....         | 267 |
| name_format .....                | 268 |
| save_upf .....                   | 268 |
| set_design_attributes .....      | 270 |
| set_design_top .....             | 270 |
| set_domain_supply_net .....      | 271 |
| set_equivalent .....             | 271 |
| set_isolation .....              | 272 |
| set_isolation_control .....      | 274 |
| set_level_shifter .....          | 274 |
| set_partial_on_translation ..... | 275 |
| set_pin_related_supply .....     | 277 |
| set_port_attributes .....        | 278 |
| set_power_switch .....           | 282 |
| set_repeater .....               | 282 |
| set_retention .....              | 283 |
| set_retention_control .....      | 284 |
| set_retention_elements .....     | 284 |
| set_scope .....                  | 285 |
| set_simstate_behavior .....      | 288 |
| set_variation .....              | 288 |
| sim_assertion_control .....      | 289 |
| sim_corruption_control .....     | 290 |
| sim_replay_control .....         | 291 |
| upf_version .....                | 291 |
| use_interface_cell .....         | 291 |
| Supported Query Commands .....   | 293 |
| find_objects .....               | 293 |
| query_cell_instances .....       | 296 |
| query_cell_mapped .....          | 296 |

|  |            |
|--|------------|
| query_design_attributes .....                              | 296        |
| query_isolation .....                                      | 297        |
| query_port_state .....                                     | 297        |
| query_power_domain .....                                   | 298        |
| query_power_domain_element .....                           | 298        |
| query_power_state .....                                    | 299        |
| query_power_switch .....                                   | 299        |
| query_pst .....  | 300        |
| query_pst_state .....                                      | 300        |
| query_retention .....                                      | 301        |
| query_retention_control .....                              | 301        |
| query_supply_net .....                                     | 302        |
| query_supply_port .....                                    | 302        |
| Supported Attributes .....                                 | 302        |
| Tcl Commands .....   | 312        |
| describe_state_cross_coverage .....                        | 313        |
| getenv .....   | 314        |
| pa_checks .....  | 315        |
| save_checks_config .....                                   | 323        |
| set_corruption_extent .....                                | 324        |
| set_feedthrough_object .....                               | 325        |
| set_pgpin_vct .....  | 326        |
| set_related_supply_net .....                               | 327        |
| <b>Appendix B</b>  |            |
| <b>Model Construction for Power Aware Simulation .....</b> | <b>331</b> |
| Basic Model Structure .....                                | 331        |
| Named Events in Power Aware .....                          | 333        |
| Attributes .....   | 334        |
| Model Interface Ports .....                                | 335        |
| Model Construction Examples .....                          | 337        |
| <b>Appendix C</b>  |            |
| <b>Supplemental Information .....</b>                      | <b>341</b> |
| Power Aware Verification of ARM-Based Designs .....        | 342        |
| Abstract .....   | 342        |
| Introduction .....   | 343        |
| Active Power Management .....                              | 343        |
| Power Management Techniques .....                          | 343        |
| Power Management Specification .....                       | 344        |
| Power Management Architecture .....                        | 345        |
| Power Managed Behavior .....                               | 352        |
| Power Control Logic .....                                  | 352        |
| Power Aware Verification Flow .....                        | 353        |
| Summary .....  | 356        |



**Index**

**End-User License Agreement  
with EDA Software Supplemental Terms**



# List of Figures

---

|   |     |
|---|-----|
| Figure 1-1. Power Aware Simulation in Your Design Flow .....  | 17  |
| Figure 3-1. Supply Paths to Power Domains .....   | 48  |
| Figure 3-2. Multiple Isolation Cells .....  | 50  |
| Figure 3-3. VCT Information in the Wave Window .....  | 65  |
| Figure 3-4. Tool inserted isolation cell for iso1 -location self. ....                                    | 69  |
| Figure 3-5. Tool inserted isolation cell for iso1 -location parent .....                                  | 70  |
| Figure 3-6. Tool inserted isolation cell for iso1 -location fanout .....                                  | 71  |
| Figure 3-7. Tool inserted isolation cell for iso2 -location self. ....                                    | 72  |
| Figure 3-8. Tool inserted isolation cell for iso2 -location parent .....                                  | 73  |
| Figure 3-9. Tool inserted isolation cell for iso2 -location fanout .....                                  | 74  |
| Figure 3-10. Tool inserted isolation cell for iso3 -location self. ....                                   | 75  |
| Figure 3-11. Tool inserted isolation cell for iso3 -location parent .....                                 | 76  |
| Figure 3-12. Tool inserted isolation cell for iso3 -location fanout .....                                 | 77  |
| Figure 4-1. Power Aware Checks .....  | 85  |
| Figure 4-2. Results Analysis Window With Static Check Results .....                                       | 128 |
| Figure 6-1. Power Aware Test Plan (Flat View) in the Verification Management Tracker Window .....         | 147 |
| Figure 6-2. Power Aware Test Plan (Flat View) in an XML Format .....                                      | 147 |
| Figure 6-3. Power Aware Test Plan (Hierarchical View) in the Verification Management Tracker Window ..... | 148 |
| Figure 6-4. Power Aware Test Plan (Hierarchical View) in an XML Format .....                              | 148 |
| Figure 8-1. UPF Objects in the Structure (sim), Objects, and Wave Windows .....                           | 170 |
| Figure 8-2. Color-Coded HDL Design Elements .....   | 178 |
| Figure 8-3. Gray Area Indicates a Power Domain That is Off .....  | 178 |
| Figure 8-4. UPF Source File: Right-Click and Choose Power Domain .....                                    | 179 |
| Figure 8-5. UPF Source File: Hover the Mouse and View Tool Tip .....                                      | 179 |
| Figure 8-6. Power Aware Highlighting in the Wave Window .....   | 181 |
| Figure 8-7. Power Aware State Machine List Example .....  | 185 |
| Figure 8-8. Power Aware State Machine Viewer Window Example .....   | 187 |
| Figure 9-1. Design Consisting of Hard Macros .....  | 201 |
| Figure A-1. Design Hierarchy .....  | 290 |
| Figure A-2. Design Hierarchy .....  | 294 |
| Figure C-1. A Power-Managed Design .....  | 346 |
| Figure C-2. An ARM-based SoC with Active Power Management .....   | 354 |



## List of Tables

---

|  |     |
|--|-----|
| Table 2-1. Power Aware Simulation Flows .....                          | 21  |
| Table 3-1. Power Aware Default Settings .....                          | 28  |
| Table 3-2. Tool Behavior for iso1 Strategy .....                       | 69  |
| Table 3-3. Tool Behavior for iso2 Strategy .....                       | 71  |
| Table 3-4. Tool Behavior for iso3 Strategy .....                       | 74  |
| Table 4-1. Power Aware Checks Command Reference .....                  | 89  |
| Table 4-2. Static RTL Checks .....                                     | 90  |
| Table 4-3. Static RTL Isolation Checks .....                           | 91  |
| Table 4-4. Static RTL Level Shifter Checks .....                       | 93  |
| Table 4-5. Static RTL Path Analysis Checks .....                       | 94  |
| Table 4-6. Static RTL Back-to-Back Checks .....                        | 96  |
| Table 4-7. Static GLS Checks .....                                     | 97  |
| Table 4-8. Static GLS Isolation Checks .....                           | 98  |
| Table 4-9. GLS Static Level Shifter Checks .....                       | 101 |
| Table 4-10. Static GLS Retention Checks .....                          | 104 |
| Table 4-11. Static GLS Back-to-Back Checks .....                       | 105 |
| Table 4-12. Static GLS ELS Checks .....                                | 106 |
| Table 4-13. Static GLS Switch Checks .....                             | 106 |
| Table 4-14. Miscellaneous Checks .....                                 | 107 |
| Table 4-15. Dynamic Checks .....                                       | 108 |
| Table 4-16. Dynamic Isolation Checks .....                             | 108 |
| Table 4-17. Dynamic Level Shifter Checks .....                         | 111 |
| Table 4-18. Dynamic Retention Checks .....                             | 112 |
| Table 4-19. Dynamic Miscellaneous Checks .....                         | 113 |
| Table 4-20. Argument Values for Static RTL and Dynamic Checks .....    | 115 |
| Table 5-1. Power Aware Reports .....                                   | 132 |
| Table 6-1. Coverage Support of UPF Objects .....                       | 149 |
| Table 7-1. Supported Tcl Commands .....                                | 157 |
| Table 7-2. Unsupported Classes .....                                   | 161 |
| Table 7-3. Unsupported Properties .....                                | 161 |
| Table 8-1. Power Aware State Machines Menu .....                       | 185 |
| Table 8-2. Power Aware State Machine List Window Columns .....         | 186 |
| Table 8-3. Power Aware State Machine List Window Popup Menu .....      | 186 |
| Table 8-4. Power Aware State Machine Viewer Window GUI Elements .....  | 187 |
| Table 8-5. Power Aware State Machine Viewer Window Popup Menu .....    | 189 |
| Table 8-6. FSM View Menu, Specific to Power Aware State Machines ..... | 190 |
| Table 9-1. Identifying Criteria .....                                  | 204 |
| Table 9-2. Matching Criteria .....                                     | 204 |
| Table A-1. UPF Command Syntax and Semantics .....                      | 236 |
| Table A-2. Supported UPF Commands .....                                | 238 |

---

|   |     |
|---|-----|
| Table A-3. List of Supported Query Commands .....   | 293 |
| Table A-4. Supported UPF Attributes .....           | 303 |
| Table A-5. Supported Liberty Attributes .....       | 303 |
| Table A-6. Tool-Specific Attributes .....           | 304 |
| Table A-7. Tool-Specific Tcl Commands .....         | 312 |
| Table A-8. Supported Options — Dynamic Checks ..... | 319 |

# Chapter 1


## Introduction to Power Aware Simulation

---

Power Aware simulation enables you to perform functional verification of low-power designs together with the power management structure defined by your power intent. The tool supports Power Aware simulation for VHDL and Verilog designs in both register transfer level (RTL) and gate-level simulation (GLS).

---

### Note

 If you are using ModelSim® SE™, you must have an additional license (qpasim) to use the Power Aware functionality. Refer to “[License Feature Names](#)” in the *Installation and Licensing Guide* for more information.

---

|   |           |
|---|-----------|
| <b>Power Aware Simulation Overview .....</b>            | <b>15</b> |
| <b>Power Aware Simulation in Your Design Flow .....</b> | <b>16</b> |
| <b>Documentation Scope .....</b>                        | <b>18</b> |

## Power Aware Simulation Overview

Verilog and VHDL designs 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 the complex systems designed today. In these systems, power is typically provided to a given portion of a chip only when that part has to function. To do so, additional hardware is included in the design 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 are implemented in the design. To do so, additional logic is included in the simulation model. The 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 a Power Aware simulation, the normal build process for constructing the simulation model is modified to add the additional logic.

Power Aware simulation applies appropriate power intent to the design. 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 “[Model Construction for Power Aware Simulation](#)” for more information on Power Aware modeling.

---

## Power Aware Simulation in Your Design Flow

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

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

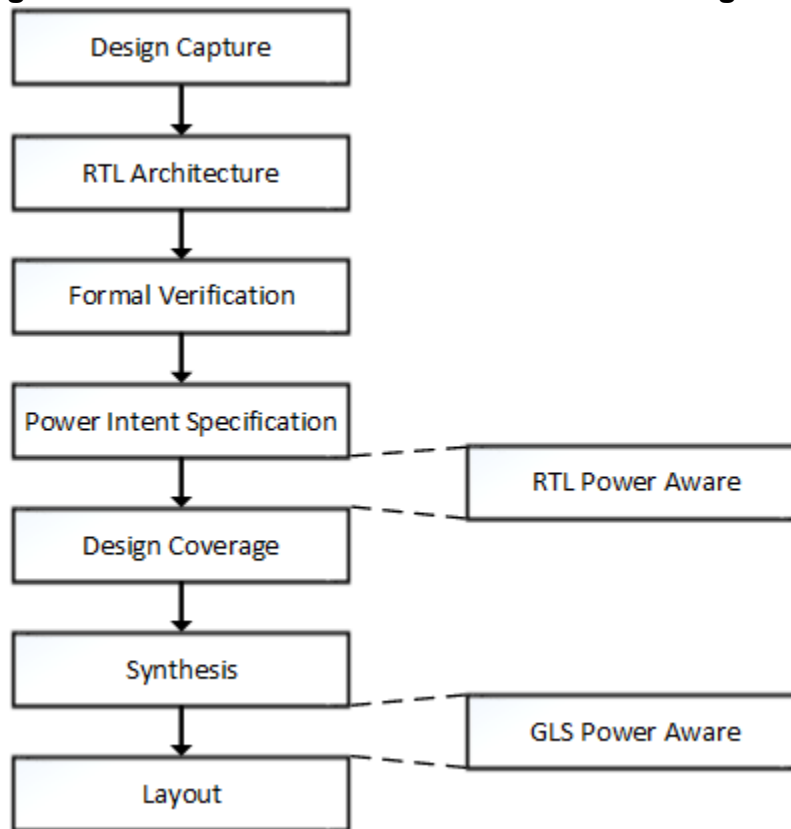
Before running Power Aware simulation, work on the following stages of your design:

- Create your design
- Define your RTL architecture
- Perform formal verification
- Specify 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.



**Figure 1-1. Power Aware Simulation in Your Design Flow**



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 Enterprise Verification Platform that can be used for Power Aware Verification, including Questa PASim, Questa ADMS, Questa Formal, Questa Codelink, Questa Verification Management, Veloce Emulator platform, and Symphony AMS.
- **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 products that are involved in the design, verification, and implementation of low-power IP, chips, and systems, such as PowerPro, and Catapult HLS.

## Documentation Scope

The purpose of the manual is to provide information on how to run a Power Aware simulation in the Questa SIM simulator.

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

- Unified Power Format (UPF) Commands — The [UPF and Tcl Commands](#) appendix lists the UPF commands and options that are currently supported for version 3.1, 3.0, 2.1, 2.0, and 1.0 of the IEEE Std 1801. However, the manual does not duplicate the usage information from the standards.
- Power Aware Arguments and Values — Refer to the *Command Reference Manual* for details on the power-specific arguments and values of various Questa SIM commands.
- Details of the Questa SIM simulator — Refer to the other manuals of the Questa SIM simulator for information on the graphical user interface (GUI), design optimization, waveform analysis, and so on.

# Chapter 2

## Getting Started With Power Aware Simulation

---

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.

|  |           |
|--|-----------|
| <b>Power Aware Simulation Inputs</b> .....   | <b>19</b> |
| <b>Power Aware Simulation Commands</b> ..... | <b>19</b> |
| <b>Power Aware Simulation Flows</b> .....    | <b>21</b> |
| Using the Three-Step Flow .....              | 21        |
| Using the PDU-Based Simulation Flow .....    | 23        |
| Using the Power Aware Interactive Mode ..... | 24        |
| Using the Two-Step Flow .....                | 25        |

## Power Aware Simulation Inputs

Power Aware simulation requires your design, power intent, and optionally, Liberty details.

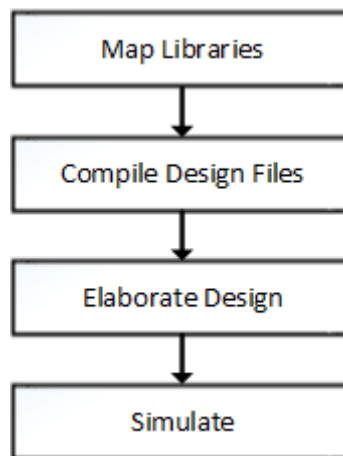
- **Design** — You can express your design in Verilog, SystemVerilog, VHDL code, or any combination of these languages. Your design can be synthesizable RTL code, behavioral RTL code, gate-level netlist, or any combination of these forms.
- **Liberty** — For designs that are, or that include, a gate-level netlist, and for designs that include instances of hard macros, you need to specify the Liberty library. The Liberty library defines the characteristics of the liberty or power management cells (such as isolation, level shifter, and retention cell) or hard macros used in the design. Power Aware simulation reads Liberty libraries and considers it in building and executing a Power Aware simulation model.
- **Power Intent** — Specify the power intent for the design using the IEEE Std 1801 in the Unified Power Format (UPF) file.

## 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.

- **vcom** or **vlog** — 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 processes the power intent specification (specified in the UPF file).
- **vsim** — Runs the simulation of the HDL design. For Power Aware simulation, the vsim command applies Power Aware simulation semantics to the HDL design.

## 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 power intent to the design:

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

4. Simulate the Power Aware version of your design:

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

## Power Aware Simulation Flows

Use the standard three-step, two-step, and PDU-based simulation flow of the tool for Power Aware simulation. Apart from the standard flows, you can use the Power Aware interactive mode.

### Note



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

Use the flows to perform Power Aware simulation for RTL, gate-level, or mixed RTL and gate-level designs.

**Table 2-1. Power Aware Simulation Flows**

| Flow   | Description  |
|--|--|
| <a href="#">Using the Three-Step Flow</a>              | In a three-step flow, you compile, optimize, and simulate your design in three separate steps.   |
| <a href="#">Using the PDU-Based Simulation Flow</a>    | In a PDU-based simulation flow, you optimize your IP or macro so that you can use the optimized model (pre-optimized design unit, or PDU) in larger blocks.  |
| <a href="#">Using the Power Aware Interactive Mode</a> | During development or debugging, if you make any change in the UPF file, you need to rerun the vopt command. Rerunning the vopt command requires reloading the design, which is time consuming. Power Aware simulation supports the Power Aware interactive mode, which enables you to load a design once and then apply the UPF specification multiple times.                 |
| <a href="#">Using the Two-Step Flow</a>                | In a two-step flow, you compile and simulate your design. The simulator implicitly optimizes the design before simulation. 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. You can pass optimization or UPF processing options to the optimization engine via the simulator. |

## Using the Three-Step Flow

In a three-step flow, you compile, optimize, and simulate your design in three separate steps.

### Prerequisites

- Create your UPF file either with respect to the top of the DUT or test bench.

## Procedure

1. Compile your design by running either the `vcom` (for VHDL) or the `vlog` (for Verilog) command:

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

2. Optimize your design:

When the UPF is written with respect to the top of the DUT:

```
vopt TB -pa_top TB/dut -pa_upf <upf_file> -pa_lib <library_name>  
-o <optimized_output> [other vopt args]
```

When the UPF is written with respect to the top of the test bench:

```
vopt TB_top -pa_upf <upf_file> -pa_lib <library_name>  
-o <optimized_output> [other vopt args]
```

- `-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 UPF file containing the power intent specification. The tool reads the UPF file and generates information required to run Power Aware simulation.
- `-pa_lib` — (Optional) Specifies the destination library in which the Power Aware information is stored. 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 resultant optimized design.

3. Simulate your design:

```
vsim <optimized_output> -pa -pa_lib <library_name> [other vsim args]
```

- `-pa` — Enables Power Aware simulation, including features for gate-level simulation.
- `-pa_lib` — (Optional) Loads Power Aware information from the specified library. If unspecified, the default is the current directory.

## 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 PDU-Based Simulation Flow

In a PDU-based simulation flow, you optimize your IP or macro so that you can use the optimized model (pre-optimized design unit, or PDU) in larger blocks.

### Prerequisites

- Compile your design.

### Procedure

1. Optimize your IP or macro with the `-pdu` argument to create a PDU, *my\_small\_pdu*:

```
vopt DUT -pa_upf my_small.upf -pdu my_small_pdu -o my_small_opt  
[other vopt args]
```

The command exposes the following at the boundary of the PDU:

- Top-level user-defined supply and logic ports
- Driver supply of top-level output ports and receiver supply of top-level input ports

The top-level ports of *my\_small.upf* are as following:

```
create_supply_port VDD_SMALL  
create_supply_port VSS_SMALL
```

2. (Optional) Optimize your larger IP, with the `-pdu` argument to create a PDU, *my\_large\_pdu*. The larger PDU, *my\_large\_pdu*, loads the smaller PDU, *my\_small\_pdu*:

```
vopt -pa_upf my_large.upf -pdu my_large_pdu -o my_large_opt
```


- Only the top-level user-defined supply and logic ports of the smaller PDU are visible in the larger PDU, and hence only the top-level ports can be connected in the larger PDU. The internals of the smaller PDU, *my\_small\_pdu*, is not visible in the larger PDU, *my\_large\_pdu*, and the tool displays an error if you access the internals of the smaller PDU in the larger PDU.
- The driver and receiver supply of top-level ports of the smaller PDU, *my\_small\_pdu* are exposed for source-sink analysis in the larger PDU, *my\_large\_pdu*.

The connections to the smaller PDU in *my\_large.upf* are as following:

```
create_supply_port VDD_LARGE  
create_supply_port VSS_LARGE  
connect_supply_net VDD_LARGE -ports { small_inst/VDD_SMALL}  
connect_supply_net VSS_LARGE -ports { small_inst/VSS_SMALL}
```

---

#### Note

 You can use *my\_large\_pdu* in a further larger block, and step 2 can be repeated multiple times.

---

3. You can specify UPF to create connections to the PDUs:

```
vopt -pa_upf top.upf TB -o TB_opt
```

The contents of *top.upf* are as following:

```
create_supply_port VDD_TOP
create_supply_port VSS_TOP
connect_supply_net VDD_TOP -ports { large_inst/VDD_LARGE}
connect_supply_net VSS_TOP -ports { large_inst/VSS_LARGE}
```

4. Simulate your design:

```
vsim -pa TB_opt [other vsim args]
```

5. To generate a report for a specific PDU, use the following command:

```
pa report -scope=/TB/my_large_pdu
```

## Using the Power Aware Interactive Mode

During development or debugging, if you make any change in the UPF file, you need to rerun the vopt command. Rerunning the vopt command requires reloading the design, which is time consuming. Power Aware simulation supports the Power Aware interactive mode, which enables you to load a design once and then apply the UPF specification multiple times.


### Procedure

1. Enter the interactive mode:

```
vopt tb -pa_interactive [other vopt args]
```

The prompt on the command line changes to PA> after the design has loaded.

#### Tip

-  You can also specify a do file that contains a list of commands that runs sequentially (from top to bottom) at the PA> prompt:

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

2. Enter vopt commands with new or different specifications without reloading the design:


```
PA> vopt -o opt -pa_upf src/test1.upf -pa_checks -pa_genrpt=de
```

After the previous vopt command has finished, you can enter another vopt command with different specifications:

```
PA> vopt -o opt -pa_upf src/test2.upf -pa_checks -pa_genrpt=pa+de
```



### Tip

 Power Aware interactive mode is a Tcl session, and you can enter any Tcl or shell commands.

3. Simulate your design in the interactive mode:

```
PA> vsim opt -pa -c
```

4. To exit the interactive mode and return to the Questa SIM command line, enter either of the following commands:

```
PA> exit
PA> quit
```

## Using the Two-Step Flow

In a two-step flow, you compile and simulate your design. The simulator implicitly optimizes the design before simulation. 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. You can pass optimization or UPF processing options to the optimization engine via the simulator.

### Prerequisites

- Create your UPF file either with respect to the top of the DUT or test bench.

### Procedure

1. Compile your design by running either the **vcom** (for VHDL) or the **vlog** (for Verilog) command:

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

2. Simulate your design:

```
vsim TB_top -pa -voptargs="-pa_upf <upf_file> -pa_lib <library_name>
[other vopt args]" [other vsim args]
```

- **-pa** — Enables Power Aware simulation, including features for gate-level simulation.
- **-voptargs** — Applies arguments to the vopt command.

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.



# Chapter 3

## Power Aware Simulation Behavior

---

Power Aware simulation exhibits features, which are either enabled by default or you need to enable them using commands and arguments.

|   |           |
|---|-----------|
| <b>Power Aware Default Settings</b> .....                         | <b>27</b> |
| <b>Power Aware Isolation Support</b> .....                        | <b>40</b> |
| <b>Power Aware Level Shifter Support</b> .....                    | <b>41</b> |
| <b>Power Aware Retention Support</b> .....                        | <b>43</b> |
| Power Aware Retention Support Overview .....                      | 43        |
| Retention Model Selection .....                                   | 43        |
| Level Sensitive Retention Model Example .....                     | 44        |
| <b>Isolation and Level Shifting Behavior on a Port</b> .....      | <b>46</b> |
| <b>Examples of Isolation on a Port</b> .....                      | <b>47</b> |
| <b>Power States</b> .....   | <b>52</b> |
| Power State Tables .....  | 52        |
| Cross-PST Analysis .....  | 53        |
| Power State Composition .....                                     | 54        |
| State Dependency Determination With add_power_state Options. .... | 56        |
| X Propagation and Drivers .....                                   | 57        |
| <b>Checker Module and the bind_checker Command</b> .....          | <b>59</b> |
| <b>Reasons For Disabling the Simulation Semantics</b> .....       | <b>63</b> |
| <b>Assertion Control</b> .....                                    | <b>63</b> |
| <b>Value Conversion Tables</b> .....                              | <b>65</b> |
| Questa-Specific Value Conversion Tables .....                     | 65        |
| <b>Path-Based Semantics for Power Aware Strategies</b> .....      | <b>67</b> |
| <b>Precedence Resolution for Power Aware Strategies</b> .....     | <b>77</b> |
| <b>Specifying Files Using the Vopt Command</b> .....              | <b>81</b> |

## Power Aware Default Settings

Use the -pa\_enable and -pa\_disable arguments of the vopt command to enable or disable the Power Aware features.

---

### Note



The default features of the tool are specific to a release and may vary from release to release.

---

The syntax of the `-pa_enable` and `-pa_disable` arguments to the `vopt` command is as follows:

```
vopt -pa_enable=<value>[+<value>...]
```

```
vopt -pa_disable=<value>[+<value>...]
```

---

**Note**



If a value is enabled by default, use it with the `-pa_disable` argument to disable it. Similarly, if a value is disabled by default, use it with the `-pa_enable` argument to enable it.

---

**Table 3-1. Power Aware Default Settings**



| <b>vopt -pa_enable and -pa_disable Value</b> | <b>Feature</b>   | <b>Default Status</b> |
|--|--|-----------------------|
| ackportbehavior                              | Reverts to UPF 1.0 behavior related to the <code>-ack_port</code> of the power switches without a supply set.<br><br>According to UPF 1.0, the <code>-ack_port</code> of a power switch without a supply set uses an always-on supply set.   | Enable                |
| alwaysonbuf                                  | Turns on corruption of buf primitives.   | Enable                |
| aoncellcrpt                                  | Disables corruption of always-on Liberty cells.<br><br>Corruption semantics of always-on cells are disabled when the Liberty model for these cells is not present.   | Enable                |
| anonymoustopsupply                           | Considers the top domain supply instead of anonymous supply as the driver and receiver supply for top level ports.   | Enable                |
| assertduringpoweroff                         | Enables assertions present in the test bench or design instance during power-down.<br>See “ <a href="#">Assertion Control</a> ”.<br><br> <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. | Disable               |
| autoconnlis                                  | Connects the unconnected pg pins of the ELS and level shifter cell to the always-on power supply created by the tool.  | Disable               |
| autoconnpacell                               | Connects all unconnected supply ports of a power management cell to the primary supply of the power domain of the cell.  | Disable               |

Table 3-1. Power Aware Default Settings (cont.)

| vopt -pa_enable and -pa_disable Value  | Feature   | Default Status |
|--|---|----------------|
| autotestplan<br> <b>Restriction:</b> Not available for ModelSim SE (Questa SIM only). | Enables generation of Power Aware test plan for Power Aware verification management. See “ <a href="#">Analyzing Coverage Using the Power Aware Test Plan</a> ”.  | Disable        |
| behaveseqcrpt  | Enables corruption of behavioral sequential blocks. Behavioral sequential blocks are those always blocks that are edge sensitive and cannot be synthesized.   | Disable        |
| biascorruption   | Enables recognition of bias supplies (such as pwell, nwell, deeppwell, and deepnwell) for Power Aware corruption.   | Disable        |
| bndrypacell  | Enables placement of isolation, level shifter, and buffer cells on the boundary ports of switch and retention cells.  | Disable        |
| bufdebug   | Displays the buffer in the Schematic window of the tool, which the following commands insert: <ul style="list-style-type: none"> <li>• set_related_supply_net</li> <li>• set_port_attributes -driver_supply/-receiver_supply/-repeater_supply</li> </ul> If you do not enable bufdebug, you cannot view the buffer in the Schematic window, but the buffer information is still available in the <i>report.pa.txt</i> file. | Disable        |
| cellcrpt   | Disables corruption of internal combinational signals of a Verilog cell.  | Enable         |
| cellonsimdisable   | Enables placement of cells (such as isolation, level shifter, and repeaters) on the simstate disabled instances.  | Disable        |
| coadebug   | Displays debug messages for CORRUPT_ON_ACTIVITY-based simstate corruption.  | Disable        |
| codecoverage   | Enables code coverage in the Power Aware design. By default, only toggle coverage is enabled.   | Disable        |
| conninsidemacro  | Connects the unconnected pg pins of the cells present inside a macro cell to the always-on power supply created by the tool.  | Disable        |

**Table 3-1. Power Aware Default Settings (cont.)**



| <b>vopt -pa_enable and -pa_disable Value</b>  | <b>Feature</b>   | <b>Default Status</b> |
|---|--|-----------------------|
| constasfeedthru   | Enables corruption of signals driven by constants, and ports that are tied high or low.  | Enable                |
| contassignasfeedthru  | Enables corruption of signals that are driven by constants.  | Enable                |
| controlrs   | Enables receiver supply of control signals according to strategy supply.   | Disable               |
| crosscoverage<br> <b>Note:</b> The value is deprecated in the current release, and the tool will not support it in the next release. Use -pa_coverageoff=crosscov. | Disables cross coverage for Power Aware simulation.  | Enable                |
| crptbinds   | Considers the SystemVerilog bind hierarchies in the extent of a power domain and applies Power Aware semantics on them.                        | Disable               |
| csnopt  | Disables optimization of <a href="#">connect_supply_net</a> UPF command.   | Enable                |
| debug   | Displays UPF objects created in the Structure, Object, and Wave windows.   | Disable               |
| defaultlssupplies   | Enables corruption of level shifter with default supplies.   | Disable               |
| defaulttoff   | Changes the default state of supply nets and ports to FULL_ON. By default, the tool changes the default state of supply nets and ports to OFF. | Enable                |
| detectaon   | Disables the detection of the always-on cells present in the design.   | Enable                |
| detectiso   | Disables the detection of the isolation cells present in the design.   | Enable                |
| detectls  | Disables the detection of the level shifter cells present in the design.   | Enable                |
| detectret   | Disables the detection of the retention cells present in the design.   | Enable                |

Table 3-1. Power Aware Default Settings (cont.)

| vopt -pa_enable and -pa_disable Value | Feature   | Default Status |
|---------------------------------------|---|----------------|
| detectsw                              | Disables the detection of the switches instantiated in the design and their association with a UPF <a href="#">create_power_switch</a> command.   | Enable         |
| donttouchassertinbind                 | Prevents the <code>assertduringpoweroff</code> and the <code>forceasrtoffpwrdown</code> value from affecting the assertions present in the bind instances and modules during power-down.  | Disable        |
| feedthroughbuf                        | Prevents treating a buf primitive as a source or sink, and disables the path analysis.  | Enable         |
| forceasrtoffpwrdown                   | Disables assertions present in the test bench and design instance during power-down, even if the test bench or design instance has assertion control directives, such as <code>asserton</code> , <code>assertoff</code> , and <code>assertkill</code> .<br>See “ <code>assertduringpoweroff</code> ”. | Disable        |
| fsmbasedcov                           | Enables FSM based Power Aware coverage. See “ <a href="#">Power State and Transition Display</a> ”.   | Disable        |
| generic_clk_edge                      | Enables edge triggering of the UPF_GENERIC_CLOCK. See “ <a href="#">UPF Generics</a> ”.   | Disable        |
| gluelogiciso                          | Disables insertion of path-based isolation strategy on the glue logic path of ports and the tool displays a warning message. For details on path-based-semantics, see “ <a href="#">Path-Based Semantics for Power Aware Strategies</a> ”.  | Enable         |
| highlight                             | Enables highlighting in the Wave window.  | 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.   | Disable        |

**Table 3-1. Power Aware Default Settings (cont.)**

| <b>vopt -pa_enable and -pa_disable Value</b> | <b>Feature</b>   | <b>Default Status</b> |
|--|--|-----------------------|
| ignoregroundsupplyconn                       | Disables application of isolation and level shifter strategies on the ports connected to the ground supplies, and the tool performs static RTL isolation, static RTL level shifter, or dynamic checks on the ports.  | Enable                |
| ignorehangingoutput                          | <p> Ignores static/dynamic analysis and 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> <b>Note:</b> 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> | Disable               |
| ignorepdu                                    | Ignores preoptimized design unit (PDU), and if you specify the -pa_upf argument, the PDU is loaded and processed in vopt for Power Aware processing.   | Disable               |
| ignorepowersupplyconn                        | Disables application of isolation and level shifter strategies on the ports connected to the power supplies, and the tool performs static RTL isolation, static RTL level shifter, or dynamic checks on the ports.   | Enable                |



**Table 3-1. Power Aware Default Settings (cont.)**

| <b>vopt -pa_enable and<br/>-pa_disable Value</b> | <b>Feature</b>   | <b>Default Status</b> |
|--|--|-----------------------|
| ignorespecialdrivers                             | Prevents 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 checks, or dynamic checks (missing level shifter or missing isolation). | Disable               |
| ignoresupplyconn                                 | Disables static/dynamic analysis and application of isolation/level shifter strategies when ports are connected to power/ground supplies.  | Enable                |
| ignoretielow                                     | Prevents application of isolation and level shifter strategies on the ports tied to 0, and the tool does not perform any static RTL isolation, static RTL level shifter, or dynamic checks on the ports.   | Disable               |
| ignoretiehigh                                    | Prevents application of isolation and level shifter strategies on the ports tied to 1, and the tool does not perform any static RTL isolation, static RTL level shifter, or dynamic checks on the ports.   | Disable               |

**Table 3-1. Power Aware Default Settings (cont.)**


| <b>vopt -pa_enable and<br/>-pa_disable Value</b> | <b>Feature</b>  | <b>Default Status</b> |
|--|---|-----------------------|
| ignoreundriveninput                              | <p> Ignores static/dynamic analysis and 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> <b>Note:</b> 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> | Disable               |
| immedcrt   | Corrupts values in continuous delay assign statements instantly at power-down.  | Disable               |
| inoutsupplyconn                                  | <p>Disables the inout functionality between connections involving a UPF supply net and an HDL inout port.</p> <p>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.</p>   | Disable               |
| insertiso  | Disables insertion of isolation cells.  | Enable                |
| insertls   | Disables insertion of level shifter cells.  | Enable                |
| insertret  | Disables insertion of retention cells.  | Enable                |
| libertycellcrt                                   | 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).  | Disable               |
| libertyinpcorrupt                                | Disables Liberty attribute-based input corruption, but does not affect Liberty attribute-based output corruption.   | Enable                |

Table 3-1. Power Aware Default Settings (cont.)

| vopt -pa_enable and -pa_disable Value | Feature  | Default Status |
|---------------------------------------|--|----------------|
| libertypamodelopt                     | Disables corruption of the following, based on Liberty corruption semantics: <ul style="list-style-type: none"><li>• Outputs of combinational cells</li><li>• Outputs and inputs of sequential cells</li></ul> This provides a more optimized Liberty-based corruption.  | Enable         |
| literalsupply                         | Enables support of UPF_literal_supply attribute in all UPF versions. By default, the tool supports the attribute in UPF 3.0 and above versions only.   | Disable        |
| limitassertports                      | Disables truncation of the dynamic check messages to a threshold value of 16.  | 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.   | Disable        |
| logicportiso                          | Enables support of UPF-created logic ports in the <a href="#">set_isolation</a> UPF command.   | Disable        |
| lowerboundary                         | Disables isolation on the ports present in lower boundary of power domain.   | Enable         |
| lscellcrpt                            | Disables Liberty-based corruption of level shifter cells.<br><br>If the Liberty model for a level shifter cell is not present, then driver-based corruption semantics are applied using <a href="#">input_supply_set</a> , <a href="#">output_supply_set</a> , and <a href="#">internal_supply_set</a> of the specified UPF strategy.<br><br>Corruption semantics of a level shifter cell are disabled if the Liberty model is not present and all three supplies ( <a href="#">input_supply_set</a> , <a href="#">output_supply_set</a> , and <a href="#">internal_supply_set</a> ) are missing from the UPF strategy.<br><br>The tool does not do any default automatic connections for level shifter cells. You must make explicit connections for these cells. | Enable         |

**Table 3-1. Power Aware Default Settings (cont.)**



| <b>vopt -pa_enable and -pa_disable Value</b> | <b>Feature</b>   | <b>Default Status</b> |
|--|--|-----------------------|
| msglimit                                     | Does not limit logging of messages to a default limit of 50 in the vopt log file.  | Enable                |
| multibitcell                                 | Enables inferring of multibit isolation, level shifter, and enable level shifter cells.  | Disable               |
| netsplit                                     | Enables path-based semantics for Power Aware strategies. See “ <a href="#">Path-Based Semantics for Power Aware Strategies</a> ”.  | Disable               |
| orderbysupply                                | Orders isolation and level shifter cells based on their supplies.  | Disable               |
| overridelibrelsupplies                       | Enables the tool to override the related supplies information that you specify in the Liberty with the information that you specify in the set_port_attributes command.  | Disable               |
| pdhiertestplan                               | <p>Enables 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 pdhiertestplan argument, enable generation of the test plan using the autotestplan argument.</p> | Disable               |
| pgoutsynccheck                               | Enables the check for synchronization of all inputs to a resolved parallel supply net that is connected to one or more supply sources that are internal pg pins of direction output.   | Disable               |
| powerunconnnets                              | 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.  | Disable               |
| powerstatesonports                           | Enables you to specify states on ports/nets via <a href="#">add_power_state</a> command in PST.  | Disable               |
| powerstatesemantics                          | Enables UPF 3.0 power states semantics.  | Disable               |
| pstcomp                                      | Disables cross-PST analysis. See “ <a href="#">Cross-PST Analysis</a> ”.   | Enable                |
| reevalinitial                                | Applies initial re-evaluation globally in the complete design.   | Disable               |

Table 3-1. Power Aware Default Settings (cont.)

| vopt -pa_enable and -pa_disable Value | Feature   | Default Status |
|---------------------------------------|---|----------------|
| reportcrossings                       | Writes the results of the static RTL path analysis check to the Static Check Report, <i>report.static.txt</i>   | 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.   | Disable        |
| rslnvnetcheck                         | Enables a check when different states are driven by active drivers of a supply net defined with the -resolve parallel option. The check may issue warnings based on scenarios defined in <a href="#">set_partial_on_translation</a> .   | Disable        |
| scmrelsconn                           | Does not provide precedence to the level shifter strategy supply over the primary supply of the domain for the connection of 2 PG pin level shifter/ELS/combo cell supply pin that has std_cell_main_rail : true attribute.<br><br>By default, the primary supply of the domain takes precedence.                           | Enable         |
| singleisoaon                          | Connects the supply of 1-PG pin isolation cells to the always_on supply.<br><br> <b>Note:</b> For isolation cells, if you specify both singleisoaon and singlerailconn values with the vopt command, then singleisoaon takes precedence. | Disable        |
| singlerailconn                        | Connects the supply of 1-PG pin ELS (enable level shifter), level shifter, and isolation cells to the primary supply of the power domain in which the cell is placed.   | Disable        |
| spaprecedence                         | Enables UPF attribute semantics as per UPF 3.1.   | Disable        |
| supplyattroncells                     | Enables source sink analysis to consider related supplies of Power Aware retention and switch cells that are present in the path.   | Disable        |

**Table 3-1. Power Aware Default Settings (cont.)**

| <b>vopt -pa_enable and -pa_disable Value</b> | <b>Feature</b>  | <b>Default Status</b> |
|--|---|-----------------------|
| supplylogicexpr                              | Disables corruption of the power states if supply and logic expressions are different.  | Enable                |
| supplyequivalence                            | Enables supply equivalence semantics as per UPF 3.1. By default, UPF 3.0 supply equivalence semantics are followed.   | Disable               |
| swcellcrpt                                   | <p>Disables Liberty-based corruption of switch cells.</p> <p>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.</p> <p>Because the tool does not insert default automatic connections for switch cells, you need to make explicit connections for these cells.</p>  | Enable                |
| swoutputportbehavior                         | <p>Enables UPF 3.0 behavior for the output of a power switch.</p> <p>Alternatively, use -pa_upfversion=3.0 argument with the vopt command to enable UPF 3.0 behavior.</p> <p>If you specify -supply_set for a switch, the tool behavior (as per the IEEE Std 1801) is as follows:</p> <ul style="list-style-type: none"> <li>• UPF 3.0 Behavior — The state of the output supply port of a switch does not depend on the simstate of the associated supply set.</li> <li>• UPF 2.0 Behavior — (default) If the supply set simstate is anything other than NORMAL, the state of the output supply port of a switch is UNDETERMINED.</li> </ul> | Disable               |
| syntaxchecks                                 | Disables checking the syntax of the UPF file that you specify with the vopt -pa_upf argument, before loading the design.  | Enable                |
| tiehiloportasfeedthru                        | Enables corruption of ports that are tied high or low.  | Enable                |

**Table 3-1. Power Aware Default Settings (cont.)**

| <b>vopt -pa_enable and -pa_disable Value</b> | <b>Feature</b>   | <b>Default Status</b> |
|--|--|-----------------------|
| trailingedge restore                         | Enables the restore event at the leading edge of a level sensitive restore event of a balloon-style level sensitive retention register when the -restore_condition evaluates to true.<br><br>By default, the restore event occurs at the trailing edge.  | Enable                |
| true inout                                   | Enables an inout supply port defined in HDL to act as an input as well as output port at different times according to the design or test bench.  | Disable               |
| udpno ret                                    | Ignores 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.<br><br>See “ <a href="#">Sequential UDP</a> ”.   | Disable               |
| undetermined state                           | Reverts to UPF 1.0 behavior related to Power Aware switches or supply nets/ports going into an UNDETERMINED state. By default, UPF 2.0 behavior is enabled.  | Enable                |
| upf2.1 hierarchy navigation                  | Enables updated functionality for the <a href="#">set_scope</a> UPF command in UPF 2.1.  | Disable               |
| upfsanity checks                             | Checks the syntax of the UPF file that you specify with the vopt -pa_upf argument, before loading the design. The tool continues to work even if it encounters any syntax errors.  | Disable               |
| upfsim state                                 | Adds the <i>upfsimstate</i> variable in all hierarchies. The variable indicates the simstate value of the primary supply set of the power domain.<br><br>By default, the upsimstate value is disabled. However, the tool enables the upsimstate value when you specify the -designfile argument to the vopt command. | Disable               |

**Table 3-1. Power Aware Default Settings (cont.)**

| <b>vopt -pa_enable and<br/>-pa_disable Value</b> | <b>Feature</b>  | <b>Default Status</b> |
|--|---|-----------------------|
| 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. | Disable               |

## Power Aware Isolation Support

An isolation cell ensures that the logical and electrical interactions between the source of a signal and the receiver are correct. It prevents power-down regions of a design from propagating unknown values to the rest of the design. Power Aware simulation implements isolation by inserting a cell into the design where isolation is required.

Defining an isolation cell depends on whether an isolation cell already exists in the design.

- If a design contains an explicit cell instance that functions as an isolation cell, Power Aware simulation does not insert an additional isolation cell or modify the existing isolation cell. It identifies an explicit cell instance as an isolation cell when you specify the following:
  - RTL design — The set\_isolation -instance command in the UPF file. Power Aware simulation detects these instances and performs isolation checks.
  - GLS design — For cells that are not specified in the UPF file, Power Aware simulation automatically detects the correct UPF strategy, such as is\_isolation\_cell:

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

- If a design does not contain an isolation cell, you must add one by specifying the set\_isolation command (without the -instance option) to provide an isolation strategy. The tool 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.

By default, if Power Aware simulation inserts an isolation cell, it uses a built-in behavioral model for isolation. However, you can insert a different model by using the map\_isolation\_cell UPF command.



---

**Tip**

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

## Results

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

## Related Topics

[Static RTL Isolation Checks](#)

# Power Aware Level Shifter Support

A level shifter cell is required whenever a signal crosses a source power domain operating at a voltage level that is different from the voltage level of the sink power domain.

Defining a level shifter cell depends on whether a level shifter cell already exists in the design.

- If a design contains an explicit cell instance that functions as a level shifter cell, Power Aware simulation does not insert an additional level shifter cell or modify the existing level shifter cell. It identifies an explicit cell instance as a level shifter cell when you specify the following:
  - RTL design — The `set_level_shifter -instance` command in the UPF file. Power Aware simulation detects these instances and performs level shifter checks.
  - GLS design — For cells that are not specified in the UPF file, Power Aware simulation automatically detects the correct UPF strategy, such as `is_level_shifter_cell`:


```
cell (LS_LH) {  
    is_level_shifter_cell : true;  
    ...  
}
```

- If a design does not contain a level shifter cell, you must add one by specifying the `set_level_shifter` command (without the `-instance` option) to provide a level shifter strategy. The tool then implicitly inserts a level shifter cell on any port that satisfies all of the following:
  - Matches the criteria defined in the level shifter strategy.
  - Requires level shifter 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 level shifter cell present and identified.

By default, if the tool inserts a level shifter cell, the tool uses a built-in behavioral model for level shifting. However, you can insert a different model (when required) by using the [map\\_level\\_shifter\\_cell](#) UPF command to identify the level shifter model to be used.

---

**Tip**

 To prevent a redundant level shifter cell inserted on a port, use the `set_level_shifter -instance` command to identify the instance of the existing level shifter cell.

---

### Threshold Control for Level Shifters

Set a global threshold level when a level shifter cell is not required for a particular range of voltage differences using the following command:

```
vopt -pa_lsthreshold <real>
```

where <real> is any numerical value that specifies a voltage threshold.

In this case, Power Aware simulation does not insert any level shifter cell if the source-sink voltage difference is less than the voltage threshold, and does not flag any missing level shifter errors.

# Power Aware Retention Support

Retention saves the value of a design element in a power domain before switching off the power to that element. When power is restored to that element, the value is retained. Based on the retention control signals specified in the UPF specification, Power Aware simulation automatically selects a model for retention.

[Checker Module and the `bind\_checker` Command](#)

|  |           |
|--|-----------|
| <b>Power Aware Retention Support Overview .....</b>  | <b>43</b> |
| <b>Retention Model Selection.....</b>                | <b>43</b> |
| <b>Level Sensitive Retention Model Example .....</b> | <b>44</b> |

## Power Aware Retention Support Overview

Retention saves the value of a design element in a power domain prior to switching off the power to that element. The value is restored when power to the element is turned on.

The `set_retention` (and in UPF 1.0 `set_retention_control`) UPF command determines which registers in a power domain are retention registers. The command sets the corresponding save and restore signals for the retention registers. You can specify the `UPF_GENERIC_CLOCK` and `UPF_GENERIC_ASYNC_LOAD` signals in the save, restore, and retention conditions.

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

Following is the sequence for specifying retention in UPF:

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)
```

## Retention Model Selection

Based on the retention control signals specified in the UPF specification, Power Aware simulation automatically selects a model for retention. A 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.

In simulation, two processes are added for each register, to model the retention behavior:

- One process is sensitive to the `save_signal` in accordance to the save sense.
- One process is sensitive to the `restore_signal` in accordance to the restore sense.

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

There are separate models for single and dual control signals:

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

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.

- **Single control, level-sensitive model** — Power Aware simulation selects this model if both the `save_signal` and the `restore_signal` are level sensitive, and the same signal is used for save and restore.
- **Dual control, level-sensitive model** — Power Aware simulation selects this model if both the `save_signal` and the `restore_signal` are level sensitive, and two different signals are used for save and restore.
- **Single control, edge-sensitive model** — Power Aware simulation selects this model if any control signal (`save_signal` or `restore_signal`) is edge sensitive, and the same signal is used for save and restore.
- **Dual control, edge-sensitive model** — Power Aware simulation selects this model if either the `save_signal` or the `restore_signal` is edge sensitive, and two different signals are used for save and restore.

## Level Sensitive Retention Model Example

Power Aware simulation provides models for retention cells that support both edge-sensitive and level-sensitive detection. Based on the retention control signals in the UPF specification, Power Aware simulation automatically selects the model. An example of a single control, level-sensitive model illustrates the use of the UPF retention command with level-sensitive controls.

## Example Retention Model

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 and the restore\_signal level, the retention register switches between normal and retention operations:

- 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 restored.
- When restore\_signal is de-asserted, the normal operation of the register resumes.

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

## Example Protocol

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

- In the save phase, normal register operation occurs:  $D \rightarrow Q$  at clock edges. At the save event, the register output is 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 a new value of D from the next active edge of clock.
- The register output is corrupted when the primary power or retention power goes off.
- On primary powerup (and 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 are both 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 into retain mode where the register gets the retained value and the effect of clock, and set and reset are disabled.
- Register output is corrupted at power-down.
- At power-up, the register goes into retain mode if `retention_condition` is active; otherwise, normal operation resumes.
- If `retention_condition` goes down during power-down, the retained value is corrupted.

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 the retention cell is corrupted, and the X value is restored when the 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` is added in the Power Aware retention models.

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

- For a user-defined model with an `RETPWR` port and corresponding logic, the 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/blk0/inst0_MyRetModel
```

## Isolation and Level Shifting Behavior 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.

Using the `-source` and `-sink` options affects isolation ([set\\_isolation](#)) and level shifter ([set\\_level\\_shifter](#)) insertion behavior in the following ways:

- Determines all paths passing through a given port.  
To determine the path, all buffers, isolation cells, and level shifter cells are treated as feedthroughs and actual drivers and receivers are determined.
- Inserts 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.
- Determines, 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.

---

## Examples of Isolation on a Port

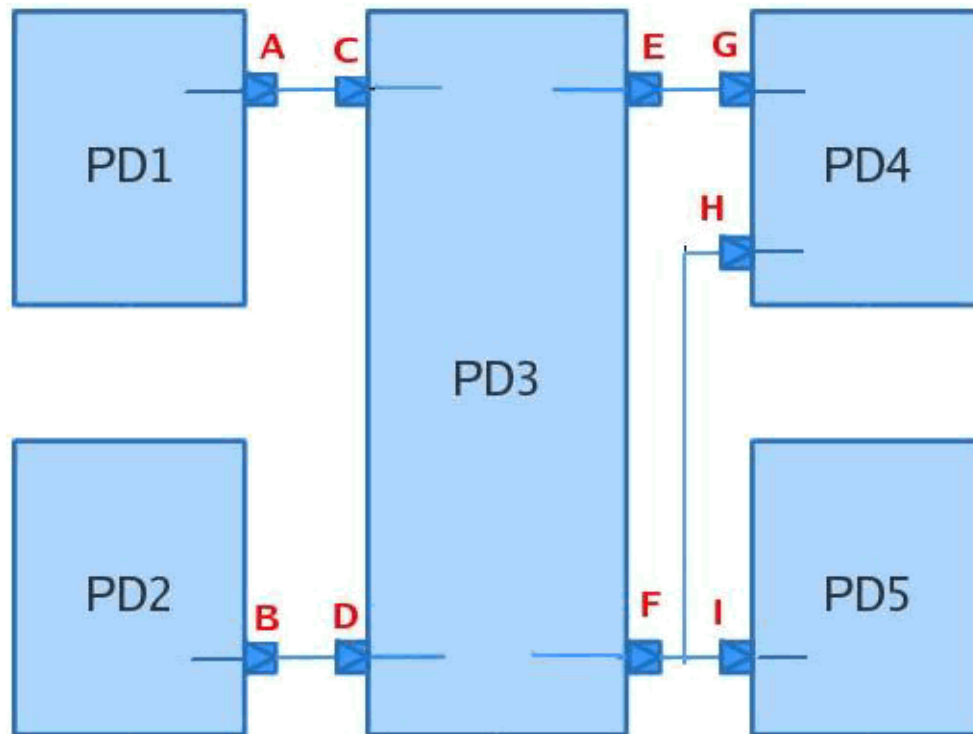
Use the `set_isolation` command to isolate ports.

Figure 3-1 shows a block diagram of power domains, ports, and paths for use in the examples that follow.

The following list shows fragments of UPF commands used to define the diagram of Figure 3-1:

```
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 3-1. Supply Paths to Power Domains**



#### Source Examples

- Place isolation cell at Port C and isolate Path A-C.

```
set_isolation iso1 -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 iso1 -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 3-2](#) 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.

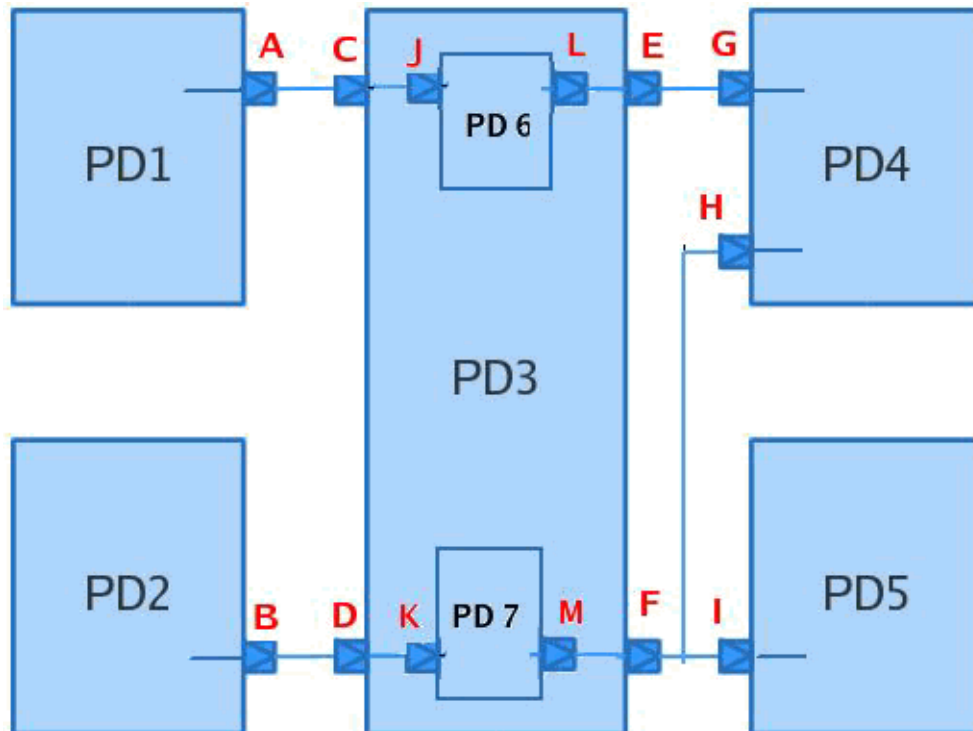
```
set_isolation iso2 -domain PD3 -source PD3_SS -location parent ...
```

- 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 3-2. Multiple Isolation Cells**



#### Multiple Strategies in a Path Examples

Refer to [Figure 3-2](#) 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 ...
```

## Power States

Power Aware simulation uses information from a Power State Table (PST) for Power Aware analysis.

|   |           |
|---|-----------|
| <b>Power State Tables.....</b>  | <b>52</b> |
| <b>Cross-PST Analysis .....</b>   | <b>53</b> |
| <b>Power State Composition .....</b>                                    | <b>54</b> |
| <b>State Dependency Determination With add_power_state Options.....</b> | <b>56</b> |
| <b>X Propagation and Drivers.....</b>                                   | <b>57</b> |

## Power State Tables

Power Aware simulation uses information from a Power State Table (PST) for 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 are 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.

### 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 simulator converts the power states specified by the add\_power\_state command in the UPF file to the corresponding PST states.

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 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 a 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 that is absent in another PST in the same scope and with the same or equivalent supply, it is marked as **UNDETERMINED**. A warning is flagged and the state is ignored for PST Analysis.

## 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, the tool 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
- Supply port — Power state information (state and voltage level) is defined by the `supply_net_type` declaration of the HDL cell
- 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

The tool performs level shifter and isolation cell checks from information provided with `add_power_state` UPF commands and also reports the power domain state dependencies in the Architecture Report.

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). The tool uses this net state and voltage information to determine whether isolation or level-shifting is required or not.

---

**Note**

---



If the tool is statically unable to determine the validity of state combinations of two power domains/supply sets, the tool treats it as a valid state combination.

---

In the following example for static analysis, the tool assumes that PD1\_ON and PD2\_ON may co-exist at some point.

```
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.

---

**Note**

---



For better static analysis, 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).

To determine the state relation between two power domains/supply sets, the tool 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

Use the `-logic_expr` or `-supply_expr` option of the UPF `add_power_state` command to determine whether power states of different power domains can co-exist.

### Example 1: Dependency Specified With `add_power_state -logic_expr`

In this example, from `logic_expr` of `PD1_S1` the tool 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 {VDD2 == FULL_ON && GND2 == FULL_ON}}

add_power_state PD2 -state PD2_S2 {-supply_expr {VDD1 == FULL_ON &&
                                                  GND1 == FULL_ON}}
```

### Example 2: Dependency Specified With `add_power_state -logic_expr`

In this example, from `logic_expr` of `PD1_S1` and `PD2_S2` the tool 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 3: Dependency Specified With `add_power_state -logic_expr`

In this example, from `logic_expr` of `PD1_S1`, `PD3_S3` the tool 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 4: Dependency Specified With `add_power_state -logic_expr`

In this example, from `logic_expr` of `PD1_S1` and `PD2_S2` the tool 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: Dependency Specified With add\_power\_state -supply\_expr

In this example, from supply\_expr of PD1\_S1 and PD2\_S2 (VDD state) the tool 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 6: Dependency Specified With add\_power\_state -supply\_expr

In this example, from supply\_expr of PD1\_S1 and PD2\_S2 (VDD state), the tool 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 7: Dependency Specified With add\_power\_state -supply\_expr

In this example, from supply\_expr of PD1\_S1 and PD2\_S2 (VDD voltage range), the tool 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}}}}
```

## X Propagation and Drivers

Power Aware simulation generates numerous X values because of the CORRUPTION simstate. This introduces more chances to miss design issues by being too optimistic due to RTL x-optimism. To catch design issues related to X values earlier, run the vopt -xprop command as part of the three-step flow.

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
```

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 the `vopt -xprop` command is able to catch the issue immediately. This situation occurs 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. These assertions are essentially noise (false positives) and not a design issue. When the design comes out of the CORRUPT simstate, the X-propagation assertions are enabled automatically.
- 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.

The following is an example of using the entire `vopt` command, including the `-xprop` argument:

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

The `-xprop` argument provides fine control values:

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

## Related Topics

[vopt \[Questa SIM Command Reference Manual\]](#)

[Using the Three-Step Flow](#)

# Checker Module and the `bind_checker` Command

The `bind_checker` UPF command enables you to insert a checker module into a design without modifying the design code or introducing functional changes. You can verify your design by writing custom assertions and implementing coverage for UPF strategy elements.

However, writing a separate checker command for every individual design element makes the list of `bind_checker` commands unreasonably large. The tool 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 enables creating a placeholder for a parameter value. You can use generics to write a `bind_checker` command that covers 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 tool 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 input clock (for example, `clk`) of the retention flipflop.

By default, the tool does not consider the edge of the input clock. To specify the input clock edge, use following command:

```
vopt -pa_enable=generic_clk_edge
```

For a posedge triggered retention flipflop, UPF\_GENERIC\_CLOCK refers to *clk* while !UPF\_GENERIC\_CLOCK refers to *!clk*. For a negedge triggered retention flipflop, UPF\_GENERIC\_CLOCK refers to *!clk* while !UPF\_GENERIC\_CLOCK refers to *clk*.

- UPF\_GENERIC\_CLOCK\_EDGE — Captures the edge triggered input clock of the retention flipflop.

---

**Note**



Alternatively, use UPF\_GENERIC\_CLOCK with the vopt  
-pa\_enable=generic\_clk\_edge command to enable edge triggering.

---

- 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 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 checker_retention \
  -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 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]) );
```

## Reasons For Disabling the Simulation Semantics

The tool disables the simulation semantics of a design instance for various reasons, and displays a message whenever it disables the simulation semantics.

The tool disables the simulation semantics of a design instance because of any of the following reasons:

- You specify the UPF\_dont\_touch attribute to exclude any module, architecture, entity instance, and signals from Power Aware behavior:

```
set_design_attributes -attribute UPF_dont_touch TRUE -models <>
```

- You specify the UPF\_simstate\_behavior as DISABLE:

```
UPF_simstate_behavior <DISABLE>

set_design_attributes -attribute {UPF_simstate_behavior <DISABLE>}
```

- Model libraries are implicitly, explicitly, or automatically connected to supply nets.

The model library (.v) takes precedence and applies corruption semantics by itself. The tool drives the appropriate supply values to the PG-pin of the cell.

- Power Aware cells do not match to any UPF strategy.
- Design elements are implicitly, explicitly or automatically connected to a particular supply net or supply set.

## Assertion Control

Power Aware simulation enables you to control the assertions present in the test bench or design instance during power-down.

By default, the assertions (including currently active assertions are disabled) are disabled during power-down.

The tool provides the following options to control the assertions:

- **Case 1:** When your design instance or test bench has user-defined assert control tasks (such as \$asserton, \$assertoff, and \$assertkill), the tool introduces two levels of commands to control the assertions:
  - **First Level Commands** — `sim_assertion_control`, `vopt -pa_enable=assertduringpoweroff`, and `vopt -pa_enable=forceasrtoffpwrdown`
  - **Second Level Commands** — User-defined assert control tasks, such as `$asserton`, `$assertoff`, and `$assertkill`

If your design has both first level and second level commands, the first level commands take precedence and the tool controls the assertions based on the precedence of the first level commands. For details on the precedence of the first level commands, see [“Precedence of Assertion Control Commands”](#).

However, in the background, the tool evaluates the second level commands. When the first level commands turn inactive, the tool controls the assertions based on the latest evaluated second level command.

- **Case 2:** When your design instance or test bench does not have user-defined assert control tasks (such as `asserton`, `assertoff`, and `assertkill`), the tool controls the assertion based on the precedence of the assertion control commands. See [“Precedence of Assertion Control Commands”](#).

---

### Restriction



The tool does not support usage of the assertion enable `-force` or assertion enable `-release` command with the `sim_assertion_control` command, and it displays an error.

---

## Precedence of Assertion Control Commands

1. The [sim\\_assertion\\_control](#) command.
2. The `vopt -pa_enable=assertduringpoweroff` value.

It enables assertions present in the test bench or design instance during power-down.

3. The `vopt -pa_enable=forceasrtoffpwrdown` value.

It disables assertions present in the test bench or design instance during power-down.



# Value Conversion Tables

A value conversion table (VCT) is a UPF definition (IEEE Std 1801) 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.

Power Aware simulation provides several predefined VCTs described in Annex B of IEEE Std 1801-2015, which you use with the connect\_supply\_net -vct command.

Refer to the predefined VCTs in the following file:

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

**Questa-Specific Value Conversion Tables..... 65**

## Questa-Specific Value Conversion Tables

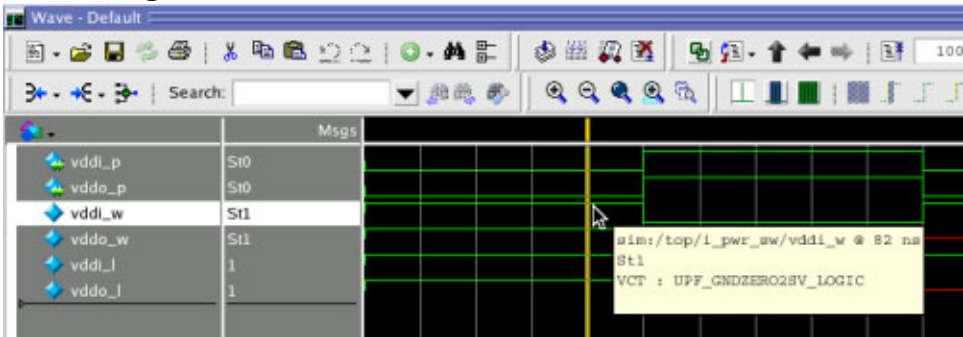
Power Aware simulation automatically inserts VCT for pins detected as power and ground based on the set\_domain\_supply\_net and supply\_set functions.

Power Aware simulation applies the power and ground-specific VCT on the supply net connection if that net is used as follows:

- Primary ground net
- Isolation ground net
- Retention ground net
- Related ground supply used in set\_related\_supply\_net
- Tied to a macro PG pin of type primary ground

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:


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



You can see reference to these VCTs in the Macro Cell Report and the Connection Report.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

### Note

 Power Aware simulation provides several predefined VCT definitions (see “[Value Conversion Tables](#)”) that you use with `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.

---

## Questa-Specific VCTs

- 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 an 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.

## Limitations

Power Aware simulation supports only the following HDL data types:

- System Verilog — logic, bit, wire, reg
- Verilog — wire, reg
- VHDL — bit, std\_logic, std\_ulogic, boolean (any subtypes of these are not supported)

## Related Topics

[Power Aware Reports](#)

## Path-Based Semantics for Power Aware Strategies

Power Aware simulation follows new semantics for inserting the isolation, level shifter, and repeater cell. The tool applies isolation, level shifter, or repeater strategies on a per path basis in the design.

The strategy that applies to a port on a path specifies the power intent for that path exclusively. If no strategy applies to a port on a path, then the power intent is to not have any cell (isolation, level shifter, or repeater) inferred by that port for that path.

The tool inserts an isolation, level shifter, and repeater cell closest to, and connected to, the target insertion port within the extent of the location domain and follows the listed semantics:

- For an isolation, level shifter, and repeater cell — If you specify `-sink` in the strategy, then the inserted cell affects receivers that are powered by a supply set that matches the specified supply set.
- For an isolation cell — If you specify `-diff_supply_only TRUE` in the strategy, then the inserted cell affect receivers that are powered by a supply set that does not match the driving supply set.

In case of multiple paths to different receiving supplies, you should be careful when specifying the location domain as an isolation, level shifter, or repeater cell inferred for one path may affect another path.

The tool displays a warning message (vopt-9927) if it cannot implement the isolation, level shifter, or repeater power intent without duplicating ports, and inserts the isolation, level shifter, or repeater cell in the specified location (self, parent, or fanout) and in the path to the target receivers.

---

### Note



By default, path-based semantics is enabled. Use the following command to disable path-based semantics:

```
vopt -pa_disable=netsplit
```

---

## Target Insertion Point

Target insertion port definition is as follows:

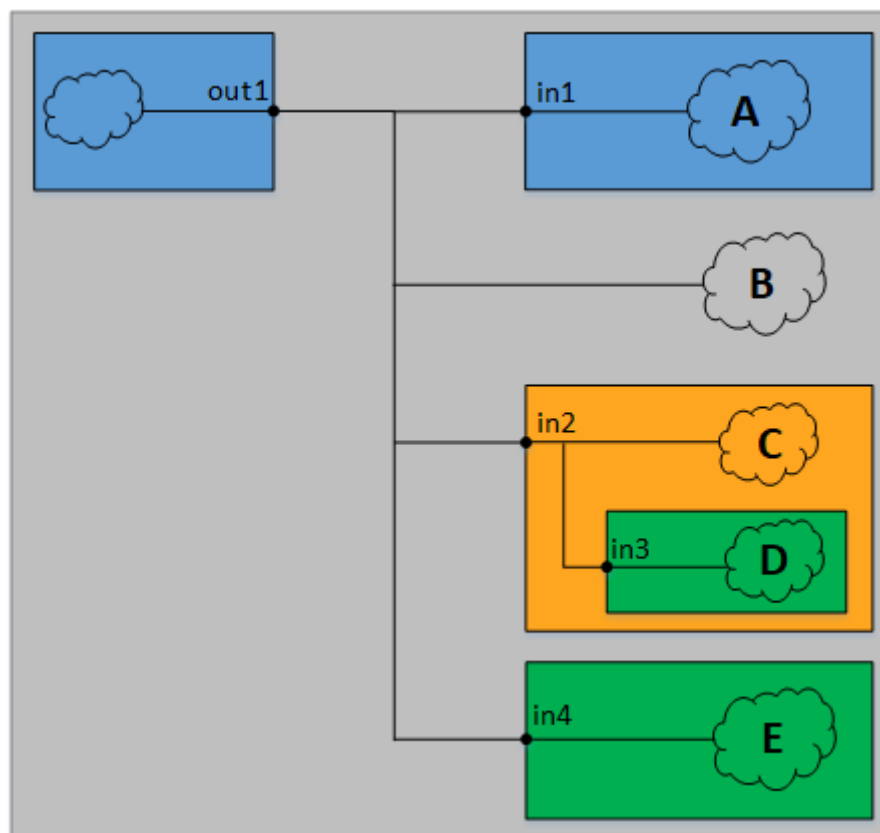
- For isolation and level shifter cell, the target insertion point depends on whether you specify the `-location fanout` option or not:
  - When you specify `-location fanout` — The target insertion port is the port on the location domain boundary that is closest to the receiving logic. If the receiving logic is in a macro cell instance, the target insertion port is the input port of that macro cell

instance, on the lower boundary of the location domain; otherwise, the target insertion port is the location domain port that is driven by the port to which the strategy applies.

- When you do not specify -location fanout — The target insertion port is the port of the location domain that is (for the self domain), or corresponds to (for the parent or child domain), the port to which the strategy applies.
- For a repeater cell, the target insertion port is the port to which the strategy applies.

## Example

Consider the following design example:



### Case 1: iso1 Strategy

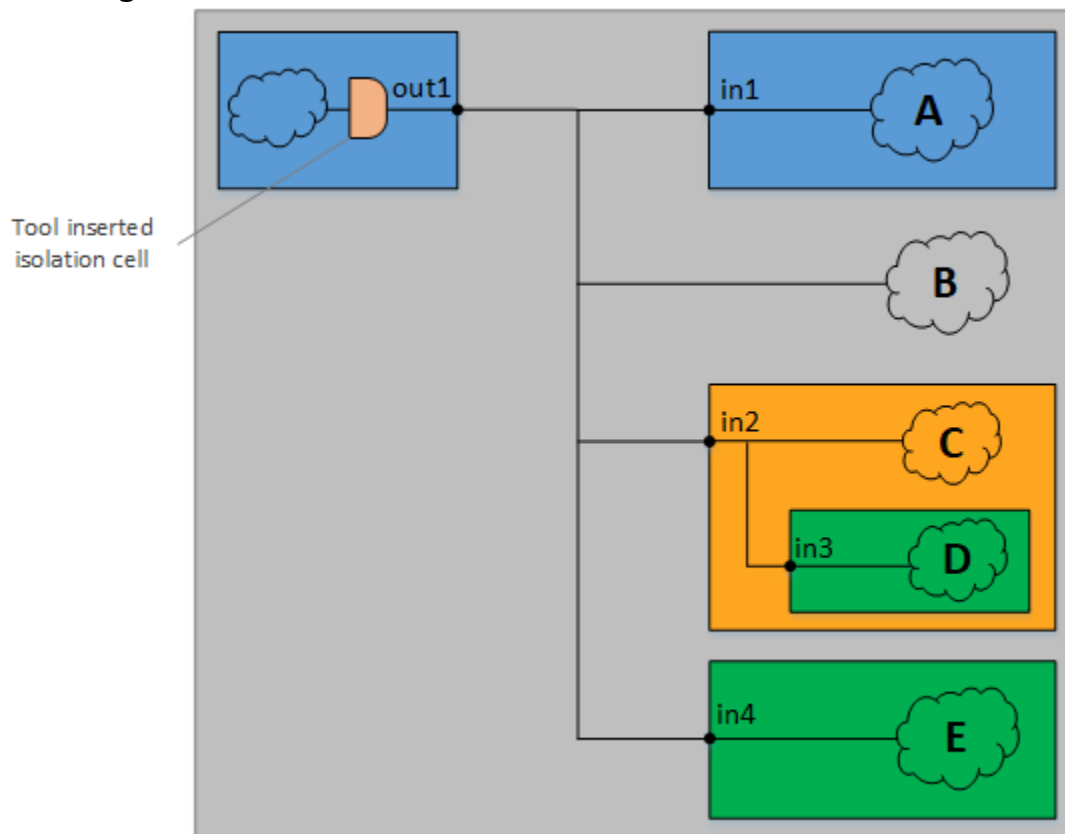
You specify the following isolation strategy:

```
set_isolation iso1 -domain blue -sink green
```

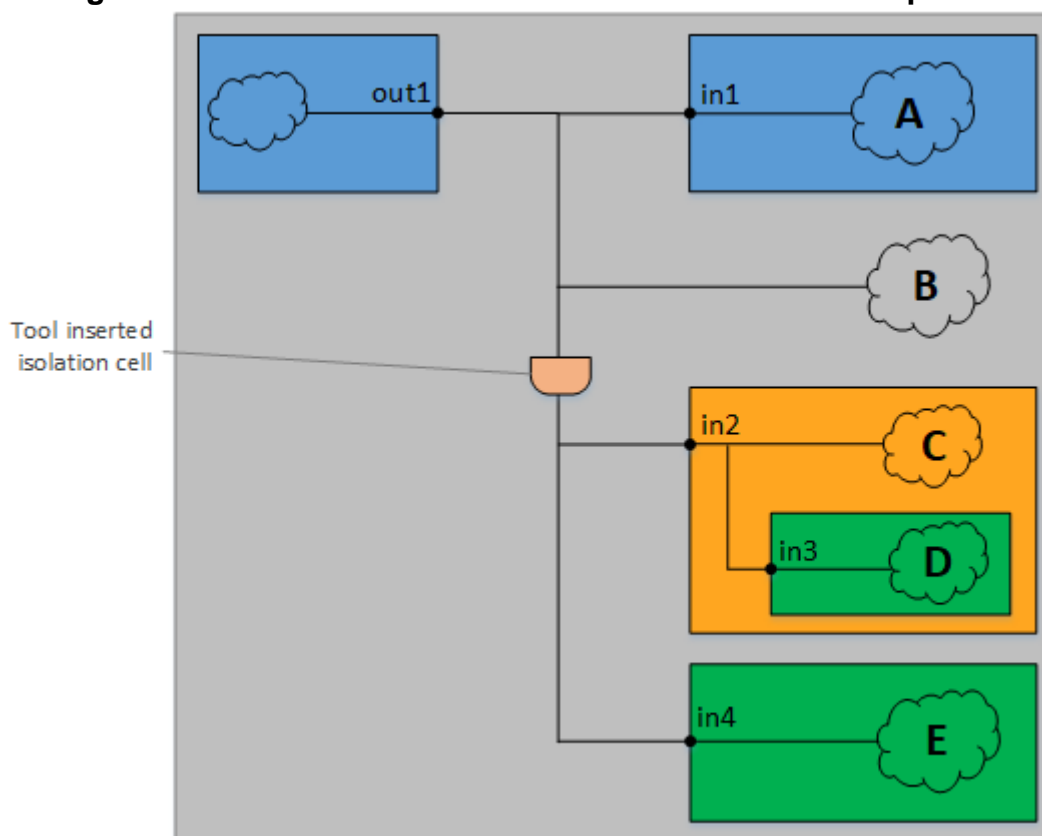
**Table 3-2. Tool Behavior for iso1 Strategy**

| -location | Semantics   |
|-----------|---|
| self      | The tool cannot implement the isolation strategy that affects only the target receiver (green domain). The tool gives a warning message (vopt-9927), and inserts the isolation cell in the self domain and in the path to the green domain. See “Figure 3-4”.   |
| parent    | The tool cannot implement the isolation strategy that affects only the target receiver (green domain). The tool gives a warning message (vopt-9927), and inserts the isolation cell in the parent domain and in the path to the green domain. See “Figure 3-5”. |
| fanout    | The tool implements the isolation strategy that affects only the target receivers (-sink). See “Figure 3-6”.  |

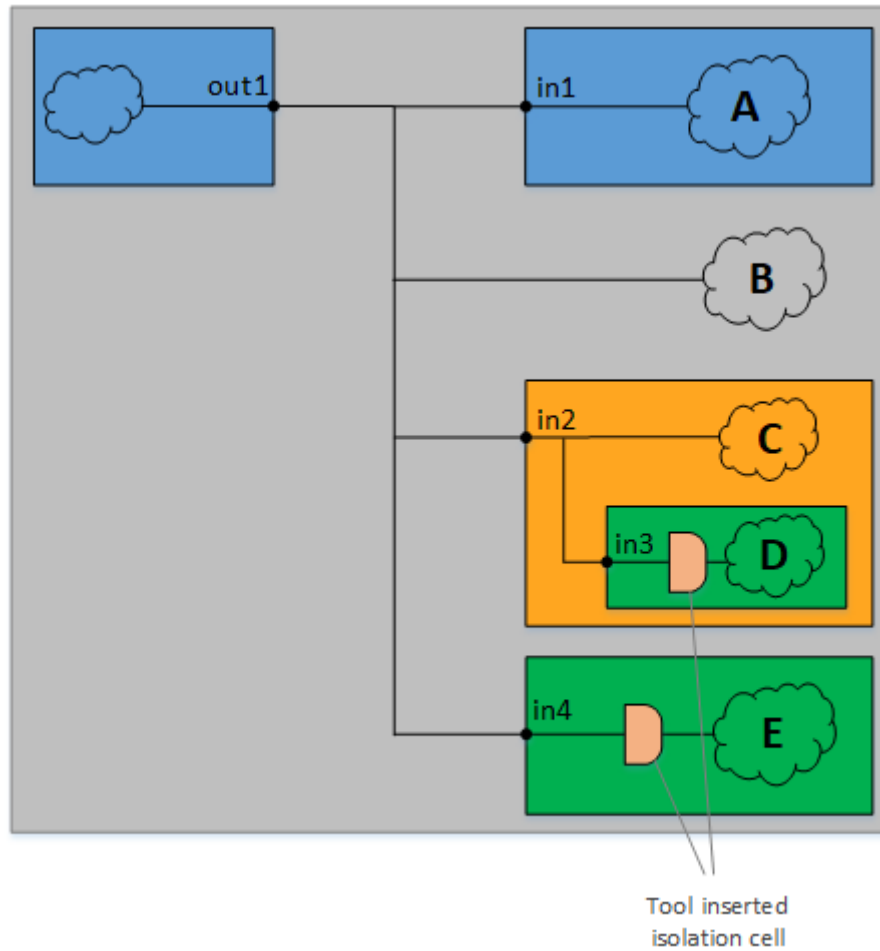
**Figure 3-4. Tool inserted isolation cell for iso1 -location self**



**Figure 3-5. Tool inserted isolation cell for iso1 -location parent**



**Figure 3-6. Tool inserted isolation cell for iso1 -location fanout**



### Case 2: iso2 Strategy

You specify the following isolation strategy:

```
set_isolation iso2 -domain blue -sink orange
```

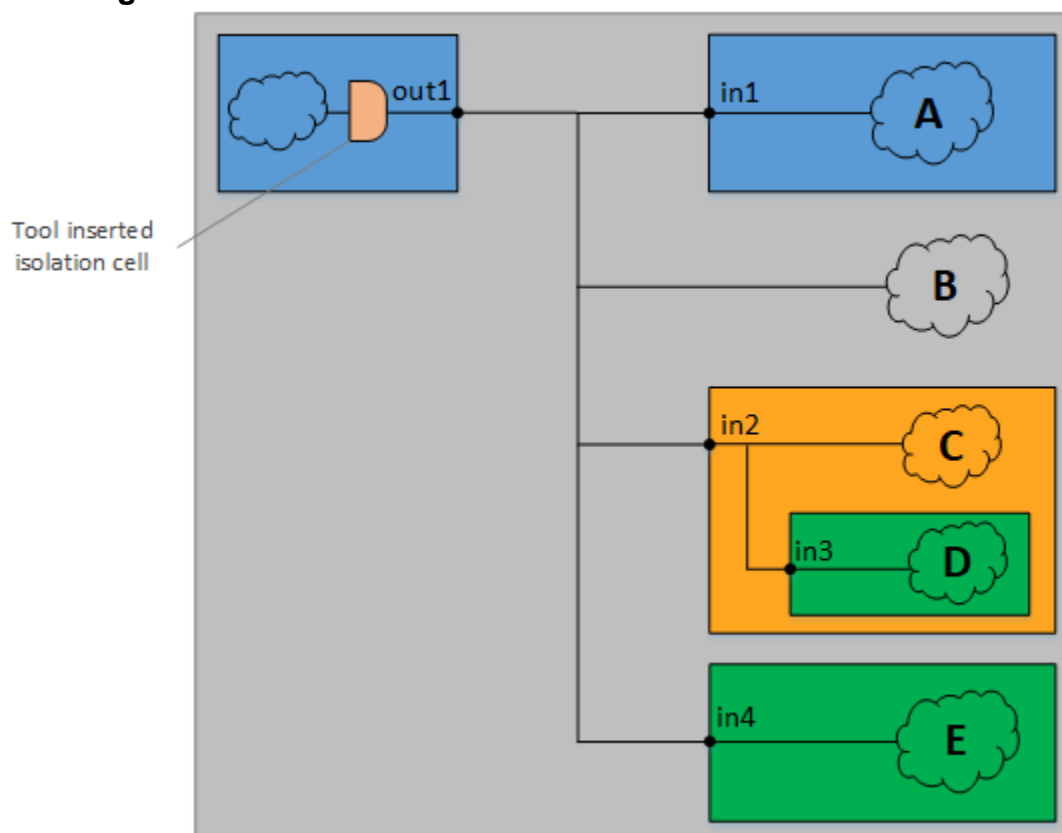
**Table 3-3. Tool Behavior for iso2 Strategy**

| -location | Semantics   |
|-----------|---|
| self      | The tool cannot implement the isolation strategy that affects only the target receiver (orange domain). The tool gives a warning message (vopt-9927), and inserts the isolation cell in the self domain and in the path to the orange domain. See “Figure 3-7”.   |
| parent    | The tool cannot implement the isolation strategy that affects only the target receiver (orange domain). The tool gives a warning message (vopt-9927), and inserts the isolation cell in the parent domain and in the path to the orange domain. See “Figure 3-8”. |

**Table 3-3. Tool Behavior for iso2 Strategy (cont.)**

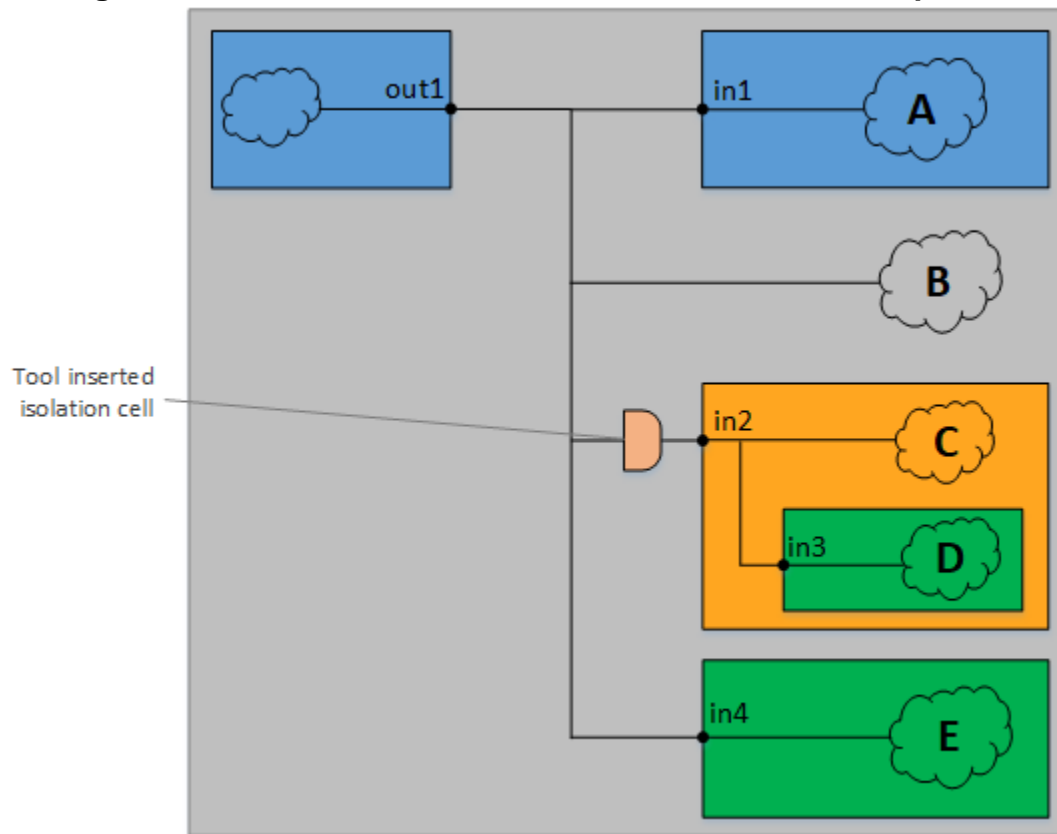
| -location | Semantics  |
|-----------|--|
| fanout    | The tool implements the isolation strategy that affects only the target receivers (-sink). See “ <a href="#">Figure 3-9</a> ”. |

**Figure 3-7. Tool inserted isolation cell for iso2 -location self**

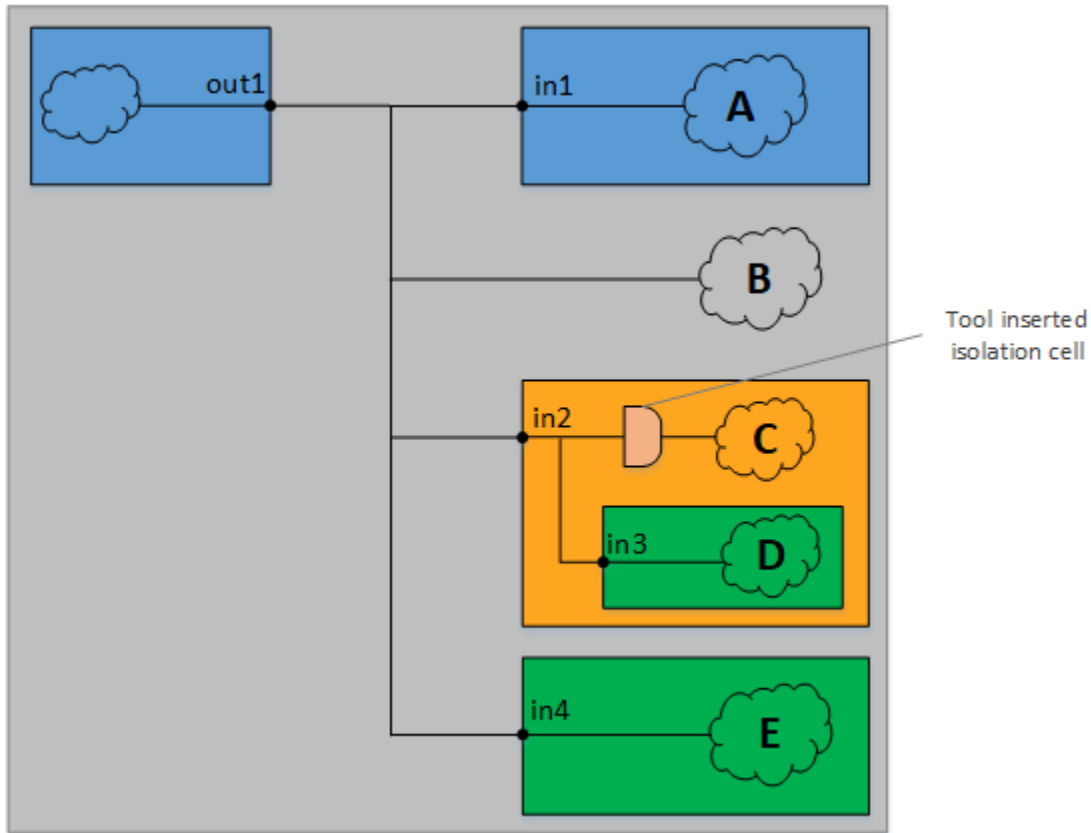




**Figure 3-8. Tool inserted isolation cell for iso2 -location parent**



**Figure 3-9. Tool inserted isolation cell for iso2 -location fanout**



### Case 3: iso3 Strategy

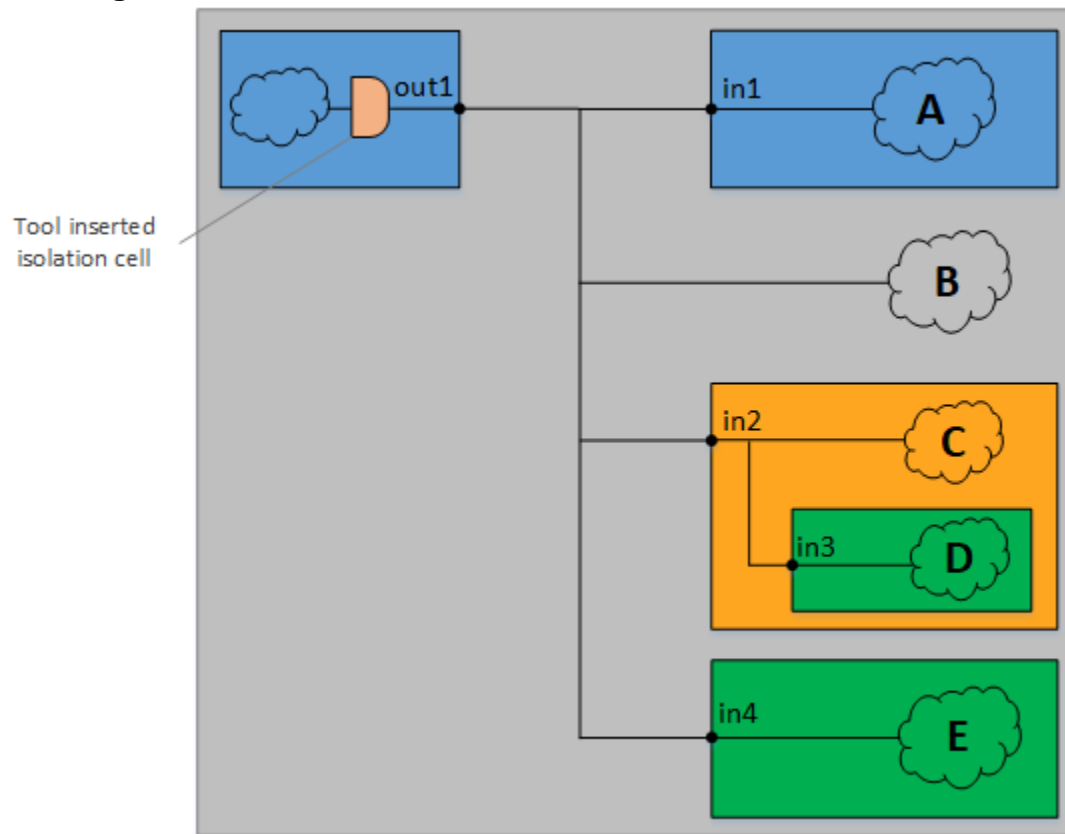
You specify the following isolation strategy:

```
set_isolation iso3 -domain blue -diff_supply_only TRUE
```

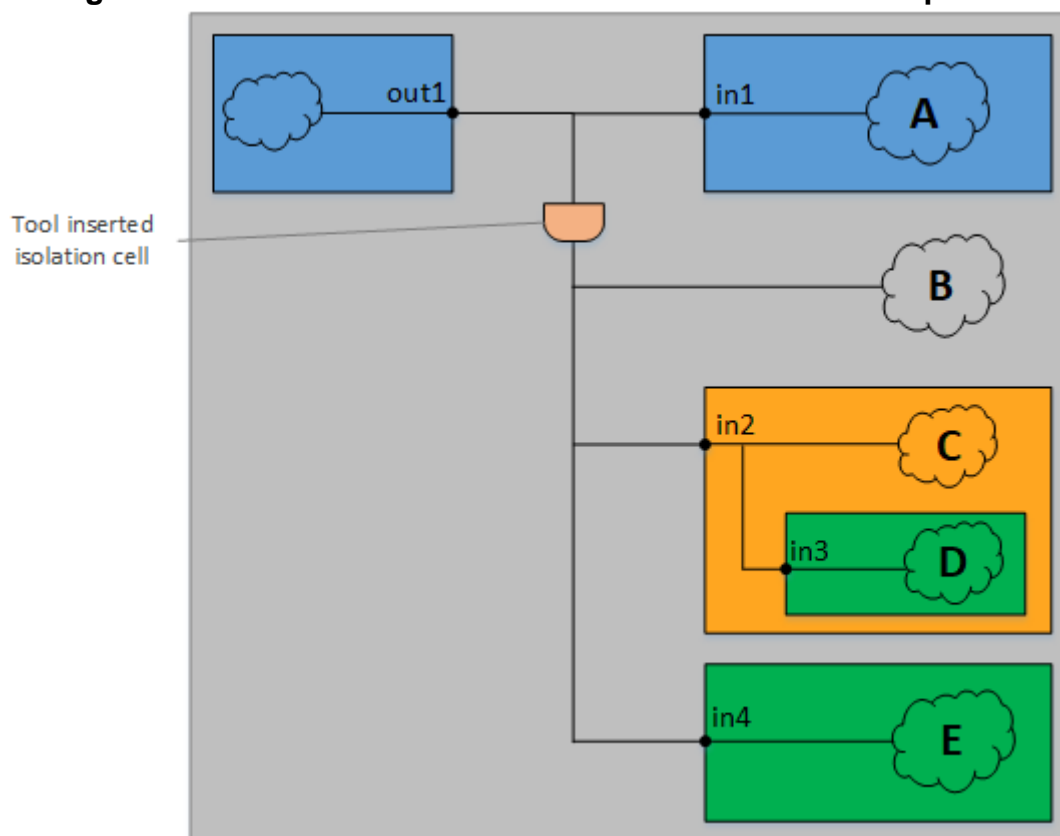
**Table 3-4. Tool Behavior for iso3 Strategy**

| -location | Semantics   |
|-----------|---|
| self      | The tool cannot implement the isolation strategy that affects only the target receivers (gray, green, and orange domains). The tool gives a warning message (vopt-9927), and inserts the isolation cell in the self domain and in the path to the gray, green, and orange domains. See “ <a href="#">Figure 3-10</a> ”. |
| parent    | The tool implements the isolation strategy that affects only the target receivers (-diff_supply_only TRUE). See “ <a href="#">Figure 3-11</a> ”.  |
| fanout    | The tool implements the isolation strategy that affects only the target receivers (-diff_supply_only TRUE). See “ <a href="#">Figure 3-12</a> ”.  |

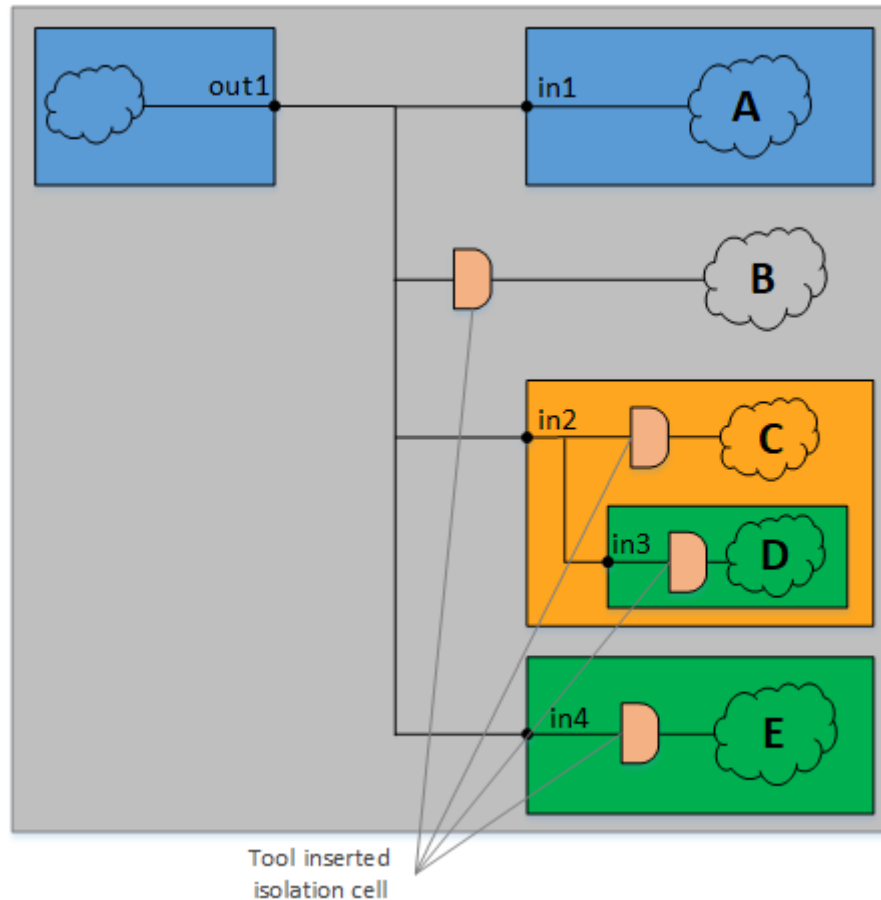
**Figure 3-10. Tool inserted isolation cell for iso3 -location self**



**Figure 3-11. Tool inserted isolation cell for iso3 -location parent**



**Figure 3-12. Tool inserted isolation cell for iso3 -location fanout**



## Precedence Resolution for Power Aware Strategies

Power Aware simulation follows the precedence resolution rules defined in the IEEE Std 1801-2018, section 5.7, but there are some differences.

### Precedence Resolution for Isolation, Level Shifter, and Repeater Strategy

The tool resolves the precedence of the UPF commands in two stages—the first stage is called granularity-based precedence resolution and the second stage is called option-based precedence resolution.

#### Granularity-Based Precedence Resolution

The tool follows the listed granularity-based precedence resolution, and in the given order:

1. Command that has part select or bit select of the port in the -elements option
2. Command that has whole port specified in the -elements option

3. Command that has instance name instead of port name in the -elements option

After applying the granularity-based precedence resolution, if there are multiple strategies that have the same level of highest precedence, then the tool applies option-based precedence resolution.

#### **Option-Based Precedence Resolution for Isolation Strategy**

The tool follows the listed option-based precedence resolution, and in the given order:

1. Command that has the -no\_isolation option specified
2. Command that has the -force\_isolation option specified
3. Command that has the -source, -sink and -diff\_supply\_only TRUE options specified
4. Command that has the -source and -sink options specified
5. Command that has the -diff\_supply\_only TRUE, and -sink or -source options specified
6. Command that has the -diff\_supply\_only TRUE, and -source or -sink options specified
7. Command that has the -diff\_supply\_only TRUE option specified
8. Command that has the -sink option specified for the output port, or the -source option specified for the input port
9. Command that has the -sink or -source option specified

If you do not specify any of the above options, then the commands have the same precedence. However, if the strategies are applicable to the same receiver, then the tool gives a suppressible error. If you suppress this error, then the tool does not apply any strategy. See [“Path-Based Semantics for Power Aware Strategies”](#).

#### **Option-Based Precedence Resolution for Level Shifter Strategy**

The tool follows the listed option-based precedence resolution, and in the given order:

1. Command that has the -no\_shift option specified
2. Command that has the -force\_shift specified
3. Command that has the -source and -sink specified
4. Command that has the -sink for the output port or -source for the input port specified
5. Command that has the -source or -sink specified

If you do not specify any of the above options, then the commands have the same precedence. However, if the strategies are applicable to the same receiver, then the tool gives a suppressible error. If you suppress this error, then the tool does not apply any strategy. See [“Path-Based Semantics for Power Aware Strategies”](#).

### **Option-Based Precedence Resolution for Repeater Strategy**

The tool follows the listed option-based precedence resolution, and in the given order:

1. Command that has the -source and -sink options specified
2. Command that has the -sink or -source option specified
3. Command that has the -source or -sink option specified

If you do not specify any of the above options, then the commands have the same precedence. However, if the strategies are applicable to the same receiver, then the tool gives a suppressible error. If you suppress this error, then the tool does not apply any strategy. See “[Path-Based Semantics for Power Aware Strategies](#)”.

### **Precedence Resolution for Retention Strategy**

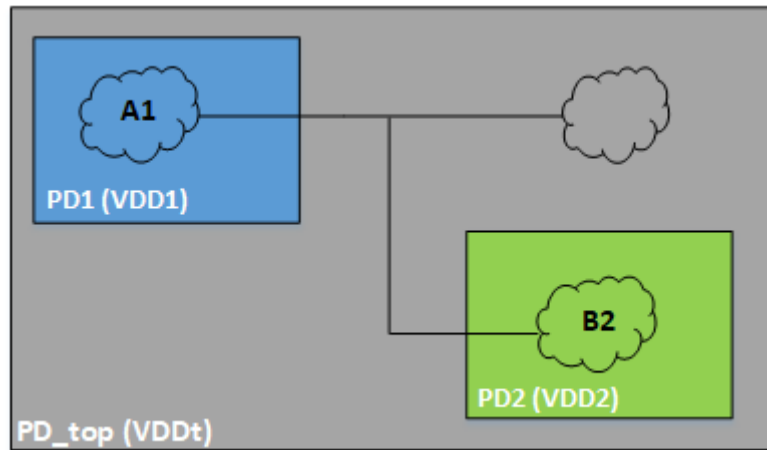
If multiple set\_retention commands apply to the same sequential or state element, then the following criteria determines the relative precedence of the commands and only the commands with the highest precedence actually applies to the sequential or state element:

1. Command that has part select or bit select of a signal in the -elements option
2. Command that has whole signal specified in the -elements option
3. Command that has process name specified in the -elements option
4. Command that has generate block name specified in the -elements option
5. Command that is specified on an interface signal

If you do not specify any of the above, then the commands have the same precedence, and the tool gives a suppressible error.

## Example

Consider the following example:



ISO1:

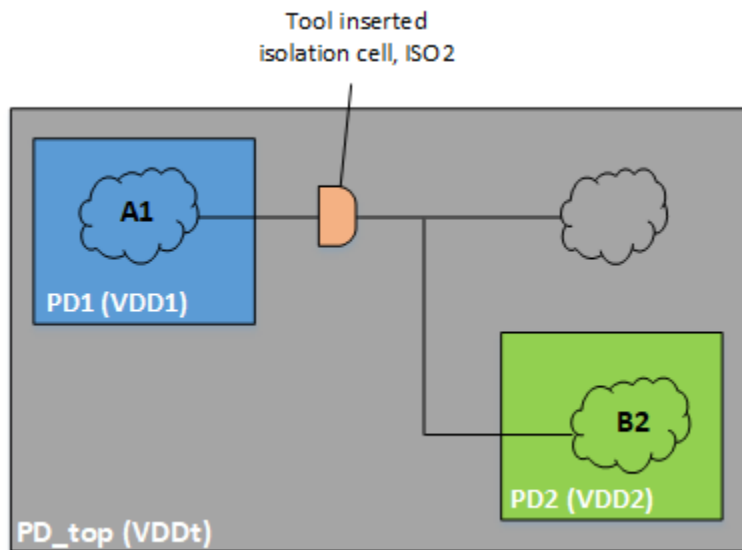
```
set_isolation -elements A1 -location parent
```

ISO2:

```
set_isolation -elements A1 -diff_supply_only TRUE -location parent
```



According to the precedence rules, ISO2 has higher precedence than ISO1. Hence, the tool drops the ISO1 strategy and gives a message, and inserts isolation cell as shown in the following figure:



## Specifying Files Using the Vopt Command

Use the `-pa_upf`, `-pa_preupfile`, and `-pa_tclfile` to specify files in the vopt command.

- **-pa\_upf** — Specifies a UPF file, which is used in the vopt run.
- **-pa\_preupfile** — Specifies a Tcl file that is processed before processing the UPF file. The Tcl file contains UPF and Tcl commands.
- **-pa\_tclfile** — Specifies a Tcl file that is processed after processing the UPF file. The Tcl file contains UPF and Tcl commands.

For example, in the following command, the tool first processes the *pre\_pa.upf* file, then the *pa.upf* file, and then the *post\_pa.upf* file:

```
vopt -pa_upf pa.upf -pa_preupfile pre_pa.upf -pa_tclfile post_pa.upf [other
vopt args]
```



# Chapter 4

## 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>84</b>  |
| Isolation Checks .....  | 85         |
| Level Shifter Checks .....  | 86         |
| Path Analysis Checks .....  | 87         |
| Retention Checks .....  | 87         |
| Back-to-Back Checks .....   | 87         |
| <b>Power Aware Checks Command Reference</b> .....                   | <b>89</b>  |
| Static RTL Checks .....   | 90         |
| Static GLS Checks .....   | 97         |
| Dynamic Checks .....  | 108        |
| Quick Reference of Static RTL and Dynamic Checks .....              | 115        |
| <b>Power Aware Checks Control</b> .....                             | <b>118</b> |
| Controlling Checks With the pa_checks Command .....                 | 119        |
| Controlling Checks With the pa_msg Command .....                    | 121        |
| Precedence Order of Arguments .....                                 | 123        |
| Pathname Convention in Arguments .....                              | 125        |
| <b>Power Aware Static Check Display</b> .....                       | <b>126</b> |
| Viewing Static Checks in the Results Analysis Window .....          | 126        |
| GUI Elements of the Results Analysis Window for Static Checks ..... | 127        |

## Power Aware Checks Overview

Power Aware checks ensure that your design 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.

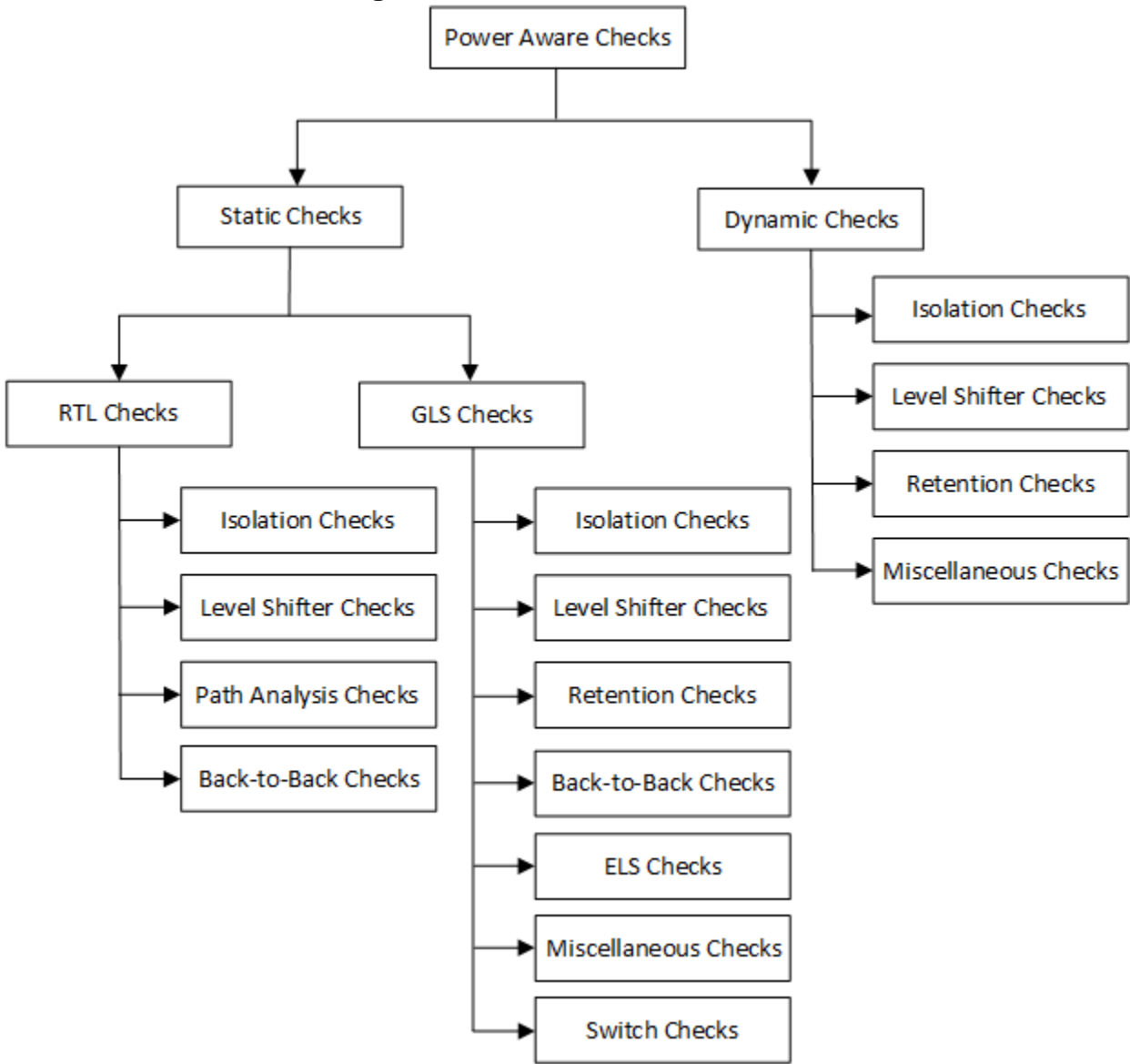
There are two categories of Power Aware checks:

- **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 4-1. Power Aware Checks



|                            |    |
|----------------------------|----|
| Isolation Checks .....     | 85 |
| Level Shifter Checks ..... | 86 |
| Path Analysis Checks.....  | 87 |
| Retention Checks .....     | 87 |
| Back-to-Back Checks.....   | 87 |

Isolation Checks

Isolation checks report any missing or inconsistent isolation cells in your design. Isolation is present in the design if the UPF file contains a set\_isolation command defining an isolation

strategy, or a `set_isolation -instance` strategy that specifies the need for isolation and points to the instantiated isolation cells.

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

Level shifter checks report any missing or inconsistent level shifter cells in your design. Level shifting is present in the design if the UPF file contains a `set_level_shifter` command defining a level shifting strategy, or a `set_level_shifter -instance` strategy that specifies the need for level shifting and points to the instantiated level shifting cells.

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 *lower* 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

Path analysis checks report not analyzed and good paths for isolation and level shifter requirements. By default, the tool does not report path analysis checks, because there are many not analyzed and good paths for a power domain crossing.

During path analysis checks, the tool reports the following paths:

- **Not analyzed 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](#)

[Power State Tables](#)

## Retention Checks

Retention checks report various inconsistent retention conditions in your design. Retention is present in the design if the UPF file contains a `set_retention` command defining a retention strategy, or a `set_retention -instance` strategy that specifies the need for retention and points to the instantiated retention cells.

If you have to retain a state element in a power domain, the retention cells have the following 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](#)

[Dynamic Retention Checks](#)

## Back-to-Back Checks

Back-to-back checks report back-to-back cells in your design. Replace the back-to-back cells with a single cell.

During back-to-back checks, the tool 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 Aware Checks Command Reference

Use the vopt arguments to enable static RTL, static GLS, and dynamic checks.

## Note


 If you do not enable a check during optimization (vopt), you cannot enable it during simulation (vsim). If you enable the check during optimization, you can disable or re-enable it during simulation.

**Table 4-1. Power Aware Checks Command Reference**

| Check                             | vopt Argument   | Description  |
|-----------------------------------|-----------------|--|
| <a href="#">Static RTL Checks</a> | -pa_checks=s    | Enables all static RTL isolation and level shifter checks.<br>Use the static RTL checks on your RTL design.  |
| <a href="#">Static GLS Checks</a> | -pa_glschecks=s | 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. See <a href="#">“Static GLS Back-to-Back Checks”</a> .<br>Use the static GLS checks on your gate-level design. |
| <a href="#">Dynamic Checks</a>    | -pa_checks=d    | Enables all dynamic checks. Alternatively, specifying -pa_checks without any argument also enables all dynamic checks.<br>Use the dynamic checks on your RTL and gate-level designs.   |

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. See [“Power Aware Checks Control”](#).

## Note

 To specify more than one Power Aware check, use a plus sign (+) to separate multiple values:

```
vopt -pa_checks=s+d
```

|   |            |
|---|------------|
| <b>Static RTL Checks</b> .....                                | <b>90</b>  |
| <b>Static GLS Checks</b> .....                                | <b>97</b>  |
| <b>Dynamic Checks</b> .....                                   | <b>108</b> |
| <b>Quick Reference of Static RTL and Dynamic Checks</b> ..... | <b>115</b> |

## Static RTL Checks

Static RTL checks validate your RTL design with the power intent defined in the UPF file. For static RTL checks, the tool analyzes the UPF file for power state tables (PSTs), power states added on power domains (by an `add_power_state` command), and supply sets. The purpose of this analysis is to detect the power domain relative ON/OFF condition and the relative operating voltages.

Use the Static RTL checks on your RTL designs.

To enable all static RTL checks (except static RTL path analysis checks), use the following command:

```
vopt -pa_checks=s
```

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 tool 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 tool performs all static RTL checks.

View the results of the static RTL checks in the following files:

- **vopt log file** — Summary of the static RTL check results
- **Static check report** (*report.static.txt*) — Details of the static RTL check results

**Table 4-2. Static RTL Checks**



| Check   | Description  |
|---|--|
| <a href="#">Static RTL Isolation Checks</a>     | Static RTL isolation checks report any missing or inconsistent isolation cells in your RTL design.         |
| <a href="#">Static RTL Level Shifter Checks</a> | Static RTL level shifter checks report any missing or inconsistent level shifter cells in your RTL design. |
| <a href="#">Static RTL Path Analysis Checks</a> | Static RTL path analysis checks report any not analyzed path or good paths in your RTL design.             |
| <a href="#">Static RTL Back-to-Back Checks</a>  | Static RTL back-to-back checks report any back-to-back cells in your RTL design.                           |

## Static RTL Isolation Checks

Static RTL isolation checks 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 the following table. All static RTL isolation check results are written to the report file, *report.static.txt*.

**Table 4-3. Static RTL Isolation Checks**

| vopt Argument     | Mnemonic                              | Description   |
|-------------------|---------------------------------------|---|
| -pa_checks=scrsti | ISO_CLAMP_DIFF_AS_RESET               | <p>Reports isolation cells whose isolation clamp value is different from the D flip flop reset value driving the isolation cell.</p> <p> <b>Restriction:</b> The tool does not support the check in the following conditions:</p> <ul style="list-style-type: none"><li>• Some bits of the register are not D flip flops.</li><li>• Different bits of the register are driven from different D flip flops.</li><li>• The reset value is a VHDL enum label.</li></ul> |
| -pa_checks=scrsti | ISO_CLAMP_SAME_AS_RESET               | <p>Reports isolation cells whose isolation clamp value is same as the D flip flop reset value driving the isolation cell.</p> <p> <b>Restriction:</b> The tool does not support the check in the following conditions:</p> <ul style="list-style-type: none"><li>• Some bits of the register are not D flip flops.</li><li>• Different bits of the register are driven from different D flip flops.</li><li>• The reset value is a VHDL enum label.</li></ul>      |
| -pa_checks=sii    | ISO_INCORRECT<br>ISO_INCORRECT_SUPPLY | <p>Reports incorrect isolation cell when an isolation cell is required for the power domain crossing, but you specify the set_isolation -no_isolation command in the UPF file.</p>  |

**Table 4-3. Static RTL Isolation Checks (cont.)**

| vopt Argument  | Mnemonic         | Description  |
|----------------|------------------|--|
| -pa_checks=smi | ISO_MISSING      | 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.   |
| -pa_checks=sni | ISO_NOT_ANALYZED | 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.                                      |
| -pa_checks=sdi | ISO_NOT_INSERTED | Reports not inserted isolation cell when you specify the isolation strategy with the set_isolation -no_isolation command in the UPF file. However, an isolation cell may or may not be required for the power domain crossing. |
| -pa_checks=sri | ISO_NOT_REQUIRED | 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.  |
| -pa_checks=svi | ISO_VALID        | 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.   |
| -pa_checks=si  |                  | Performs all static RTL isolation checks (scrsti, smi, sri, sii, svi, sni, and sdi).   |

## Related Topics

[Isolation Checks](#)

## Static RTL Level Shifter Checks

Static RTL level shifter checks 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 the following table. All static RTL level shifter check results are written to the report file, *report.static.txt*.

**Table 4-4. Static RTL Level Shifter Checks**

| vopt Argument  | Mnemonic        | Description  |
|----------------|-----------------|--|
| -pa_checks=sml | LS_MISSING      | 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 tool did not infer one.   |
| -pa_checks=srl | LS_REDUNDANT    | Reports redundant 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 tool infers one.   |
| -pa_checks=sil | LS_INCORRECT    | 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.<br><br>For example, the tool 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. |
| -pa_checks=svl | LS_VALID        | 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 tool infers one.   |
| -pa_checks=snl | LS_NOT_ANALYZED | 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.   |
| -pa_checks=sdl | LS_NOT_INSERTED | Reports not inserted level shifter cell when you specify the level shifter strategy with the set_level_shifter -no_shift command in the UPF file. However, a level shifter cell may or may not be required for the power domain crossing.  |
| -pa_checks=sl  |                 | Performs all static RTL level shifter checks (sml, srl, sil, svl, snl, and sdl).   |

## Related Topics

[Level Shifter Checks](#)

## Static RTL Path Analysis Checks

Static RTL path analysis checks 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 the following table.

To write the results of static RTL path analysis checks to the report file, *report.static.txt*, use the following command:

```
vopt -pa_enable=reportcrossings [other vopt args]
```

**Table 4-5. Static RTL Path Analysis Checks**

| vopt Argument   | Mnemonic              | Description  |
|-----------------|-----------------------|--|
| -pa_checks=snpl | LS_PATH_NOT_ANALYZED  | 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. |
| -pa_checks=scpl | LS_PATH_GOOD          | 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.                               |
| -pa_checks=snpi | ISO_PATH_NOT_ANALYZED | 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.         |
| -pa_checks=scpi | ISO_PATH_GOOD         | 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.   |

**Table 4-5. Static RTL Path Analysis Checks (cont.)**

| vopt Argument        | Mnemonic          | Description   |
|----------------------|-------------------|---|
| -pa_checks=snpl+snpi | PATH_NOT_ANALYZED | 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. |
| -pa_checks=scpl+scpi | PATH_GOOD         | 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.                |

## Related Topics

[Path Analysis Checks](#)

## Static RTL Back-to-Back Checks

Static RTL back-to-back checks 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 the following table. All static RTL back-to-back check results are written to the report file, *report.static.txt*.

**Table 4-6. Static RTL Back-to-Back Checks**

| vopt Argument    | Mnemonic           | Description  |
|------------------|--------------------|--|
| -pa_checks=s+b2b | ISO_B2B_CTRL_SAME  | Reports back-to-back isolation cells with the same control signal for a power domain crossing.   |
|                  | ISO_B2B_CTRL_DIFF  | Reports back-to-back isolation cells with different control signals for a power domain crossing. |
|                  | ISO_B2B_CTRL_UNK   | Reports back-to-back isolation cells with unknown control signals for a power domain crossing.   |
|                  | ISO_B2B_CLAMP_SAME | Reports back-to-back isolation cells with the same clamp value for a power domain crossing.      |
|                  | ISO_B2B_CLAMP_DIFF | Reports back-to-back isolation cells with different clamp values for a power domain crossing.    |

## Related Topics

[Back-to-Back Checks](#)



## 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 tool analyzes the UPF file for power state tables (PSTs), power states added on power domains (by an `add_power_state` command), and supply sets. The purpose of this analysis is to detect the power domain relative ON/OFF condition and the relative operating voltages.

The purpose of the Static GLS checks is to detect the power domain relative ON/OFF condition, their relative operating voltages, and to ensure that the power management cells in the gate level netlist are consistent with the power intent defined in UPF. Please refer to “[Power Management Cell](#)” section for details on how the tool identifies power management cells in gate-level designs.

---

### Note



Your gate-level design can be either gate-level netlists out of synthesis, or fully PG netlists out of place-and-route.

---

To enable all static GLS checks (except static GLS back-to-back cell checks), use the following command:

```
vopt -pa_glschecks=s
```

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 tool cannot determine power domain relative ON/OFF states statically. In this case, static GLS checks report strategies as *Not Analyzed*. If state dependencies between two connected power domains are present in the PST/`add_power_state`, then the tool performs all static GLS checks.

View the results of the static GLS checks in the following files:

- **vopt log file** — Summary of the static GLS check results
- **Static check report (*report.static.txt*)** — Details of the static GLS check results

**Table 4-7. Static GLS Checks**

| Check   | Description   |
|---|---|
| <a href="#">Static GLS Isolation Checks</a>     | Static GLS isolation checks report any missing or inconsistent isolation cells in your gate-level design.         |
| <a href="#">Static GLS Level Shifter Checks</a> | Static GLS level shifter checks report any missing or inconsistent level shifter cells in your gate-level design. |
| <a href="#">Static GLS Retention Checks</a>     | Static GLS retention checks report any inconsistent retention condition in your gate-level design.                |

**Table 4-7. Static GLS Checks (cont.)**

| Check   | Description   |
|---|---|
| <a href="#">Static GLS Back-to-Back Checks</a>  | Static GLS back-to-back checks report any back-to-back cells in your gate-level design.             |
| <a href="#">Static GLS ELS Checks</a>           | Static GLS ELS checks report any issues in the ELS cells present in your gate-level design.         |
| <a href="#">Static GLS Switch Checks</a>        | Static GLS switch checks report any missing or inconsistent switch cells in your gate-level design. |
| <a href="#">Static GLS Miscellaneous Checks</a> | Static GLS miscellaneous checks report various miscellaneous issues in your gate-level design.      |

## Static GLS Isolation Checks

Static GLS isolation checks report any missing or inconsistent isolation cells in your gate-level design.

When you enable static GLS checks (using `vopt -pa_glschecks=s`), the tool performs the static GLS isolation checks listed in the following table. All static GLS isolation check results are written to the report file, *report.static.txt*.

**Table 4-8. Static GLS Isolation Checks**

| Mnemonic                     | Description   |
|------------------------------|---|
| ISO_VALID_CELL_WITH_STRATEGY | 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.   |
| ISO_CELL_WITHOUT_STRATEGY    | 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.                         |
| ISO_VALID_STRATEGY_NO_CELL   | 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. |

**Table 4-8. Static GLS Isolation Checks (cont.)**

| <b>Mnemonic</b>                    | <b>Description</b>  |
|------------------------------------|---|
| ISO_INCORRECT_LOCATION             | 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. |
| ISO_MISSING_CELL                   | 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 tool infer one.  |
| ISO_NOT_REQUIRED                   | 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 tool infers one.  |
| ISO_NOT_REQUIRED_CELL_AND_STRATEGY | 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.  |
| ISO_INCORRECT_STRATEGY             | Reports incorrect isolation strategy when an isolation cell is required for the power domain crossing, but you specify the isolation strategy with set_isolation-no_isolation in the UPF file.  |
| ISO_CTRL_REACH_DATA                | Reports valid isolation cell with control reaching data when isolation enable signals for the UPF strategy reaches the data pin of the cell.  |
| ISO_CTRL_UNREACHABLE               | Reports valid isolation cell with unreachable control when isolation enable signals for the UPF strategy does not reach the enable pin of the cell.   |

**Table 4-8. Static GLS Isolation Checks (cont.)**

| Mnemonic                      | Description  |
|-------------------------------|--|
| ISO_UNANALYZED_CELL           | Reports unanalyzed isolation cell when either of the following condition is met: <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> |
| ISO_PARALLEL_CELLS            | Reports parallel isolation cell on diverging signal in the design.   |
| ISO_MISMATCHING_CLAMP         | Reports mismatching isolation clamps when the clamp-value of the isolation cell does not match with the corresponding UPF strategy.  |
| ISO_MISMATCHING_SENSE         | Reports a mismatch in the polarity of the isolation enable signal when the polarity that you specify in the UPF strategy does not match with the value you specify in the Liberty attribute.   |
| ISO_REDUNDANT_ISOLATED_IP_PIN | Reports redundant 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 is_isolated.  |
| ISO_CTRL_NOT_ALWYS_ON         | Reports isolation control signals that are not generated from the always-on region of the design.  |
| ISO_CTRL_UNCONNECTED          | Reports isolation control signal pins that are unconnected or floating in the design.  |
| ISO_CTRL_CONST_DRIVER         | Reports isolation control signal pins that are driven by constants in the design.  |

## Related Topics

[Isolation Checks](#)

## Static GLS Level Shifter Checks

Static GLS level shifter checks report any missing or inconsistent level shifter cells in your gate-level design.

When you enable static GLS checks (using `vopt -pa_glschecks=s`), the tool performs the static GLS level shifter checks listed in the following table. All static GLS level shifter check results are written to the report file, *report.static.txt*.

**Table 4-9. GLS Static Level Shifter Checks**

| Mnemonic                    | Description   |
|-----------------------------|---|
| LS_VALID_CELL_WITH_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.   |
| LS_CELL_WITHOUT_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.                           |
| LS_VALID_STRATEGY_NO_CELL   | 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. |
| LS_INCORRECT_LOCATION       | 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.       |

**Table 4-9. GLS Static Level Shifter Checks (cont.)**

| Mnemonic                          | Description   |
|-----------------------------------|---|
| LS_TYPE_MISMATCH                  | <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<br/>Inferred rule = high_to_low</p>                                   |
| LS_NOT_REQUIRED_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>            |
| LS_NOT_REQUIRED_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> |
| LS_MISSING_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>   |

**Table 4-9. GLS Static Level Shifter Checks (cont.)**

| <b>Mnemonic</b>                 | <b>Description</b>   |
|---------------------------------|--|
| LS_REDUNDANT_THRESHOLD          | Reports redundant 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.                          |
| LS_REDUNDANT_THRESHOLD_STRATEGY | Reports redundant 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.   |
| LS_REDUNDANT_THRESHOLD_CELL     | Reports redundant 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.   |
| LS_UNANALYZED_CELL              | Reports not analyzed level shifter cell when either of the following condition is met: <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> |
| LS_PARALLEL_CELLS               | Reports parallel level shifter cells on diverging signal in the design.  |

## Related Topics

[Level Shifter Checks](#)

## Static GLS Retention Checks

Static GLS retention checks report any inconsistent retention condition in your gate-level design.

When you enable static GLS checks (using `vopt -pa_glschecks=s`), the tool performs the static GLS retention checks listed in the following table. All static GLS retention check results are written to the report file, *report.static.txt*.

**Table 4-10. Static GLS Retention Checks**

| Mnemonic                      | Description   |
|-------------------------------|---|
| RET_NO_STRATEGY_FOR_CELL      | Reports valid retention cell with no retention strategy when the retention cell is not associated with any retention strategy.  |
| RET_NO_CELL_FOR_STRATEGY      | 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. |
| RET_NO_STRATEGY_ELEM_FOR_CELL | Reports retention cells that are not in the strategy element lists.   |
| RET_CTRL_NOT_ALWAYS_ON        | Reports retention control signals that are not generated from the always-on region in the design.   |
| RET_CTRL_UNCONNECTED          | Reports retention control signals that are unconnected or floating in the design.   |
| RET_CTRL_CONST_DRIVER         | Reports retention control signals that are driven by constants in the design.   |

### Related Topics

[Retention Checks](#)

## Static GLS Back-to-Back Checks

Static GLS back-to-back checks 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 the following table. All static GLS back-to-back check results are written to the report file, *report.static.txt*.



Table 4-11. Static GLS Back-to-Back Checks

| vopt Argument       | Mnemonic                 | Description  |
|---------------------|--------------------------|--|
| -pa_glschecks=s+els | ISO_LS_B2B_CELLS         | Reports back-to-back isolation-level shifter cells present in the design. They can be replaced with an ELS cell.       |
|                     | LS_ISO_B2B_CELLS         | Reports back-to-back level shifter-isolation cells present in the design. They can be replaced with an ELS cell.       |
| -pa_glschecks=s+b2b | LS_B2B_CELLS             | Reports back-to-back level shifter cells present in the design. They can be replaced with a single level shifter cell. |
|                     | ISO_B2B_CELLS_CTRL_SAME  | Reports back-to-back isolation cells with same or different control signal present in the design.                      |
|                     | ISO_B2B_CELLS_CTRL_UNK   | Reports back-to-back isolation cells with unknown control signals present in the design.                               |
|                     | ISO_B2B_CELLS_CLAMP_SAME | Reports back-to-back isolation cells with the same clamp value present in the design.                                  |
|                     | ISO_B2B_CELLS_CLAMP_DIFF | Reports back-to-back isolation cells with different clamp values present in the design.                                |

## Related Topics

[Back-to-Back Checks](#)

## Static GLS ELS Checks

Static GLS ELS checks report any issues in the ELS cells present in your gate-level design.

When you enable static GLS ELS checks (using vopt -pa\_glschecks=s), the tool performs the static GLS ELS checks listed in the following table. All static GLS ELS check results are written to the report file, *report.static.txt*.

### Restriction

 The tool performs the static GLS ELS checks on 2 PG pin cells only.

**Table 4-12. Static GLS ELS Checks**

| Mnemonic                      | Description  |
|-------------------------------|--|
| ELS_CELL_INCORRECT_LOCATION   | Reports ELS cells that are not placed in the correct domain. The correct location of the ELS cells are as following: <ul style="list-style-type: none"><li>• Sink side ELS cell — Sink domain</li><li>• Source side ELS cell — Source domain</li></ul> |
| COMBO_CELL_INCORRECT_LOCATION | Reports combo cells that are not placed in the correct domain. The correct location of the sink side combo cell is the sink domain.  |
| ELS_OUTPUT_SUPPLY_MISMATCH    | Reports ELS cells whose output supply does not match with the isolation supply set.  |
| COMBO_INPUT_SUPPLY_MISMATCH   | Reports combo cells whose input supply does not match with the isolation supply set.   |

## Static GLS Switch Checks

Static GLS switch checks report any missing or inconsistent switch cells in your gate-level design.

When you enable static GLS checks (using `vopt -pa_glschecks=s`), the tool performs the static GLS switch checks listed in the following table. All static GLS switch check results are written to the report file, *report.static.txt*.

**Table 4-13. Static GLS Switch Checks**

| Mnemonic                 | Description  |
|--------------------------|--|
| SWTCH_CTRL_NOT_ALWAYS_ON | Reports switch control signals that are not generated from the always-on region of the design. |
| SWTCH_CTRL_UNCONNECTED   | Reports switch control signals that are unconnected or floating in the design.                 |
| SWTCH_CTRL_CONST_DRIVER  | Reports switch control signals that are driven by constants in the design.                     |

## Static GLS Miscellaneous Checks

Static GLS miscellaneous checks report various miscellaneous issues in your gate-level design.

When you enable static GLS checks (using `vopt -pa_glschecks=s`), the tool performs the static GLS miscellaneous checks listed in the following table. The static GLS miscellaneous check results are written to the report file, *report.static.txt*.

**Table 4-14. Miscellaneous Checks**

| Mnemonic               | Description   |
|------------------------|---|
| CELL_MISSING_LIB_ATT   | Reports missing liberty attributes for the design cells.  |
| CELL_INCORRECT_MAPPING | <p>Reports incorrect mapping of a cell to the strategy when an isolation, a level shifter, an ELS, or a combo cell is present in the design and you incorrectly map the strategy using the <code>map_*</code> or <code>use_interface_cell</code> UPF command:</p> <ul style="list-style-type: none"><li>• An ELS or combo cell does not have <code>is_isolation_cell</code> and <code>is_level_shifter</code> attribute or equivalent attribute.</li><li>• A level shifter cell does not have <code>is_level_shifter</code> attribute or equivalent attribute.</li><li>• An isolation cell does not have <code>is_isolation_cell</code> attribute or equivalent attribute.</li></ul> <p>When the tool detects an incorrect mapping, it issues a warning message, and treats the cell as if it is of the appropriate type.</p> |

## Dynamic Checks

Dynamic checks validate your design with the power intent defined in the UPF file during the Power Aware simulation.

Use the dynamic checks on your RTL and gate-level designs.

To enable all dynamic checks, use the following command:

```
vopt -pa_checks=d
```

To display the full hierarchical path of any failing check in the Transcript window, use the following command:

```
vopt -pa_checksoption=assertionhierpath [other vopt args]
```

**Table 4-15. Dynamic Checks**

| Check  | Description   |
|--|---|
| <a href="#">Dynamic Isolation Checks</a>     | Dynamic isolation checks report any missing or inconsistent isolation cells in your RTL and gate-level designs.         |
| <a href="#">Dynamic Level Shifter Checks</a> | Dynamic level shifter checks report any missing or inconsistent level shifter cells in your RTL and gate-level designs. |
| <a href="#">Dynamic Retention Checks</a>     | Dynamic retention checks report any inconsistent retention condition in your RTL and gate-level designs.                |
| <a href="#">Dynamic Miscellaneous Checks</a> | Specify a value to the vopt -pa_checks command to enable dynamic miscellaneous checks.                                  |

## Dynamic Isolation Checks

Dynamic isolation checks report any missing or inconsistent isolation cells in your RTL and gate-level designs.

To enable dynamic isolation checks, specify a value to the vopt -pa\_checks command as shown in the following table. 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 4-16. Dynamic Isolation Checks**

| vopt Argument  | Mnemonic          | Description  |
|----------------|-------------------|--|
| -pa_checks=icp | QPA_ISO_CLAMP_CHK | 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. |

**Table 4-16. Dynamic Isolation Checks (cont.)**

| <b>vopt Argument</b> | <b>Mnemonic</b>           | <b>Description</b>   |
|----------------------|---------------------------|--|
| -pa_checks=idp       | QPA_ISO_DIS_PSO           | Flags violation if the isolation control signal is disabled when the source domain is OFF and sink domain is ON.   |
| -pa_checks=iep       | QPA_ISO_EN_PSO            | Flags violation if the isolation control signal is not enabled when the source domain is OFF and sink domain is ON.  |
| -pa_checks=idpcoa    | QPA_ISO_DIS_COA           | 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: <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>  |
| -pa_checks=iepcoa    | QPA_ISO_EN_COA            | 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: <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> |
| -pa_checks=ifc       | QPA_ISO_FUNC_C<br>HK      | Flags violation if the value at the output of an isolation cell is different from that at its input during the inactive isolation period.  |
| -pa_checks=ira       | QPA_ISO_REDUND<br>ANT_ACT | 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.  |

**Table 4-16. Dynamic Isolation Checks (cont.)**

| vopt Argument   | Mnemonic                    | Description   |
|-----------------|-----------------------------|---|
| -pa_checks=irc  | QPA_ISO_PORT_TOGGLE         | <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> <b>Note:</b> The coverage report refers to QPA_ISO_PORT_TOGGLE as QPA_ISO_TOGGLE_POSEDGE and QPA_ISO_TOGGLE_NEGEDGE</p> <ul style="list-style-type: none"> <li>The following error message in the Transcript window corresponds to QPA_ISO_TOGGLE_POSEDGE in the coverage report. <pre># ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 60 ns, Isolated port for isolation cell (strategy: iso_PD_mid1) on port '/ tb/TOP/mid1/out2_bot[3]' toggled when its control signal is <b>activated</b>.</pre> </li> <li>The following error message in the Transcript window corresponds to QPA_ISO_TOGGLE_NEGEDGE in the coverage report. <pre># ** Error: (vsim-8936) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port for isolation cell (strategy: iso_PD_mid1) on port '/ tb/TOP/mid1/out1_bot[0]' toggled when its control signal is <b>de-activated</b>.</pre> </li> </ul> |
| -pa_checks=ircn | QPA_ISO_PORT_TOGGLE_NEGEDGE | Flags violation if the value on the isolated port changes when its isolation control signal is changing state (high_to_low).  |
| -pa_checks=ircp | QPA_ISO_PORT_TOGGLE_POSEDGE | Flags violation if the value on the isolated port changes when its isolation control signal is changing state (low_to_high).  |
| -pa_checks=it   | QPA_ISO_PORT_ACTIVE         | Flags violation if the value on the isolated port changes during the active isolation period.   |
| -pa_checks=umi  | QPA_UPF_MISSING_ISO_CHK     | Flags violation if the isolation strategy is not specified for a power domain crossing whose source domain is ON and sink domain is OFF.  |
| -pa_checks=i    |                             | Enables the following dynamic isolation checks: iep, iepcoa, idp, idpcoa, irc, ira, icp, and ifc.   |

## Related Topics

[Isolation Checks](#)

## Dynamic Level Shifter Checks

Dynamic level shifter checks report any missing or inconsistent level shifter cells in your RTL and gate-level designs.

To enable dynamic level shifter checks, specify a value to the `vopt -pa_checks` command as shown in the following table. All dynamic level shifter check results are written to the Transcript window.

**Table 4-17. Dynamic Level Shifter Checks**

| vopt Argument  | Mnemonic                 | Description  |
|----------------|--------------------------|--|
| -pa_checks=uml | QPA_UPF_MISSING_LS_CHK   | 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.<br><br>Also, strategy specified with <code>set_level_shift -no_shift</code> in the UPF file are checked for any missing level shifter cells.   |
| -pa_checks=uil | QPA_UPF_INCORRECT_LS_CHK | 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.<br><br>For example, the tool reports an incorrect level shifter cell if the power domain crossing requires a <code>low_to_high</code> level shifter cell and you specify the level shifter strategy in the UPF file as <code>high_to_low</code> . |
| -pa_checks=ul  |                          | Enables all dynamic level shifter checks.  |

### Note



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 Retention Checks

Dynamic retention checks report any inconsistent retention condition in your RTL and gate-level designs.

To enable dynamic retention checks, specify a value to the `vopt -pa_checks` command as shown in the following table. All dynamic retention check results are written to the Transcript window.

**Table 4-18. Dynamic Retention Checks**

| vopt Argument    | Mnemonic          | Description   |
|------------------|-------------------|---|
| -pa_checks=rcs   | QPA_RET_CLK_STATE | 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.   |
| -pa_checks=rop   | QPA_RET_OFF_PSO   | <p><b>Master-Slave Configuration</b> — Flags violation if the retention condition is not asserted when the power is switched off.</p> <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>  |
| -pa_checks=rpo   | QPA_RET_PD_OFF    | <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><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> |
| -pa_checks=rsa   | QPA_RET_SEQ_ACT   | <p>Flags violation if the clock toggles during the retention period.</p> <p> <b>Note:</b> This check is for balloon-latch configuration only; it does not apply to master-slave (slave-alive) retention.</p>   |
| -pa_checks=rtcon | QPA_RET_ON_RTC    | Flags violation if the retention condition is off when the power is enabled.  |



**Table 4-18. Dynamic Retention Checks (cont.)**

| vopt Argument     | Mnemonic           | Description  |
|-------------------|--------------------|--|
| -pa_checks=rtcoff | QPA_RET_OFF_RTC    | Flags violation if the retention condition is off when the power is disabled.  |
| -pa_checks=rtctog | QPA_RET_RTC_TOGGLE | Flags violation if the retention condition toggles during power-down. Toggling of the retention condition during power-down corrupts the retained value.                         |
| -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> |
| -pa_checks=r      |                    | Enables all dynamic retention checks (rcs, rop, rpo, rsa, rtcon, rtcoff, and rtctog).  |

## Related Topics



[Retention Checks](#)

## Dynamic Miscellaneous Checks

Specify a value to the vopt -pa\_checks command to enable dynamic miscellaneous checks.

All dynamic miscellaneous check results are written to the Transcript window.


**Table 4-19. Dynamic Miscellaneous Checks**

| vopt Argument | Mnemonic        | Description   |
|---------------|-----------------|---|
| -pa_checks=t  | QPA_PD_BIAS_ACT | Flags violation when input to a power domain toggles when sink supply is in the BIAS state.<br> <b>Restriction:</b> The tool does not support the check for VHDL port of user-defined enum type. |
| -pa_checks=t  | QPA_PD_OFF_ACT  | Flags violation if the input to a power domain toggles when the power domain is turned off.<br> <b>Restriction:</b> The tool does not support the check for VHDL port of user-defined enum type. |

**Table 4-19. Dynamic Miscellaneous Checks (cont.)**

| vopt Argument  | Mnemonic                          | Description   |
|----------------|-----------------------------------|---|
| -pa_checks=cp  | QPA_CTRL_SIG_CRP<br>T             | <p>Flags violation when the power signal to any power domain gets corrupted.</p> <p>This check does not flag a violation when both the source and sink power domain supplies are off.</p> <p>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.</p>  |
| -pa_checks=p   | QPA_PD_STATUS_I<br>NFO            | Reports the power domains that are switched on or off.  |
| -pa_checks=pis | QPA_UPF_ILLEGAL_<br>STATE_REACHED | <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>   |
| -pa_checks=ugc | QPA_GLITCH_ASSE<br>RT_ERR         | 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 <a href="#">pa msg -glitch_window</a> command to specify the maximum allowed time window of the glitch.   |
| -pa_checks=npu | QPA_NRET_ASYNC<br>F               | <ul style="list-style-type: none"> <li>Default behavior:<br/>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>.</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: <ol style="list-style-type: none"> <li>Flags violation if the asynchronous reset is not asserted within T<sub>max</sub> time units after power-up. See “<a href="#">Supported Attributes</a>”.</li> <li>Flags violation if the asynchronous reset is not asserted for minimum T<sub>width</sub> time units. See “<a href="#">Supported Attributes</a>”.</li> </ol> </li> </ul> |

**Table 4-19. Dynamic Miscellaneous Checks (cont.)**

| vopt Argument  | Mnemonic       | Description   |
|----------------|----------------|---|
| -pa_checks=upc | QPA_UPF_PG_CHK | Flags violation if the isolation or retention supplies are switched off during the active isolation or retention period.<br><br> <b>Note:</b> The coverage report refers QPA_UPF_PG_CHK as QPA_SUPPLY_PWR_CHK. |

## 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.

### Note

 All static GLS checks (except static GLS back-to-back checks) are enabled using the following command:

```
vopt -pa_glschecks=s [other vopt args]
```

**Table 4-20. 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=sdi   |                   |
| 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 |
| Isolation enable protocol check for COA states  |                  | -pa_checks=iepcoa |

**Table 4-20. Argument Values for Static RTL and Dynamic Checks (cont.)**

| Checks  | Static RTL Value     | Dynamic Value     |
|---|----------------------|-------------------|
| 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      |                   |
| Not analyzed path for isolation and level shifter requirement | -pa_checks=snpl+snpi |                   |

**Table 4-20. Argument Values for Static RTL and Dynamic Checks (cont.)**

| Checks  | Static RTL Value     | Dynamic Value  |
|---|----------------------|----------------|
| 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 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 file, and use this file with the `vopt -pa_tclfile` command. The `pa_checks` commands in the Tcl 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 tool resolves the priority of commands based on the precedence of arguments of the `pa_checks` command. See “[Precedence Order of Arguments](#)”.

- **pa msg** — Specify the `pa msg` commands in a Tcl file, and use this file at the VSIM prompt. The `pa msg` commands in the Tcl 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 tool resolves the priority of commands based on the precedence of arguments of the `pa msg` command.

|   |            |
|---|------------|
| <b>Controlling Checks With the <code>pa_checks</code> Command</b> | <b>119</b> |
| <b>Controlling Checks With the <code>pa msg</code> Command</b>    | <b>121</b> |
| <b>Precedence Order of Arguments</b>                              | <b>123</b> |
| <b>Pathname Convention in Arguments</b>                           | <b>125</b> |

## Controlling Checks With the `pa_checks` Command

Enable granular control of static RTL and dynamic checks by using a Tcl file, containing the `pa_checks` commands, with the `vopt` command.

### Procedure

1. Create a Tcl 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 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 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 tool in a UPF file or a Tcl 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_mid1_1) on port '/tb/TOP/mid1/
out1_bot[4]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_1) on port '/tb/TOP/mid1/
out1_bot[2]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_2) on port '/tb/TOP/mid1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_1) on port '/tb/TOP/mid1/
out1_bot[4]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_1) on port '/tb/TOP/mid1/
out1_bot[2]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_mid1# **
Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns,
Isolated port for isolation cell (strategy: iso_PD_mid1_2) on port '/tb/
TOP/mid1/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/mid1/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 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 {mid1/
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_mid1_2) on port '/tb/TOP/mid1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_2) on port '/tb/TOP/mid1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 110 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_2) on port '/tb/TOP/mid1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_mid1
```

## Related Topics

[pa\\_checks](#)

[save\\_checks\\_config](#)

# Controlling Checks With the `pa msg` Command

Enable granular control of dynamic checks by invoking a Tcl 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 file that contains one or more `pa msg` commands.
2. Invoke the Tcl file at the VSIM prompt.

This applies the `pa msg` commands to the Power Aware simulation.

For example, the following command specifies a Tcl 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_mid1_1) on port '/tb/TOP/mid1/
out1_bot[4]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_1) on port '/tb/TOP/mid1/
out1_bot[2]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 50 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_2) on port '/tb/TOP/mid1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_1) on port '/tb/TOP/mid1/
out1_bot[4]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_1) on port '/tb/TOP/mid1/
out1_bot[2]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 64, Power Domain:PD_mid1# **
Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns,
Isolated port for isolation cell (strategy: iso_PD_mid1_2) on port '/tb/
TOP/mid1/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/mid1/out2_bot
```

To decrease the reported error messages to those of interest, invoke a Tcl 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 {mid1/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_mid1_2) on port '/tb/TOP/mid1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 80 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_2) on port '/tb/TOP/mid1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_mid1
# ** Error: (vsim-8910) QPA_ISO_PORT_TOGGLE: Time: 110 ns, Isolated port
for isolation cell (strategy: iso_PD_mid1_2) on port '/tb/TOP/mid1/
out2_bot[4:5]' toggled when its control signal is activated.
# File: ./src/model/test.upf, Line: 79, Power Domain:PD_mid1
```

## Precedence Order of Arguments

If a Tcl file contains multiple `pa_checks` or `pa msg` commands that apply to the same check, then the tool 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 tool 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.

## Examples

Assume you include the following two `pa_checks` commands in the same Tcl file:

### Case 1

```
pa_checks -enable -checkIds {npu} -chksctrl_expr {/tb/top_tb/set == 1'b1  
|| /tb/ret_pwr == 1'b1} -elements {top_tb/inst6/out1}  
  
pa_checks -enable -checkIds {npu} -chksctrl_expr {/tb/top_tb/set == 1'b1  
|| /tb/ret_pwr == 1'b0} -models srff_vl2
```

Where the `top_tb/inst6/out1` element belongs to the `srff_vl2` model.

Since `-elements` takes precedence over `-models`, the tool performs the first command, and for the second command, the tool ignores the `top_tb/inst6/out1` element and applies the non-retention register check (`npu`) on other elements in the `srff_vl2` model.

### Case 2

```
pa msg -enable -checkIds {umi} -domains {PD1}  
  
pa msg -disable -checkIds {umi} -elements {top/inst1}
```

Where the `top/inst1` element belongs to the `PD1` power domain.

Since `-elements` takes precedence over `-domains`, the tool disables the dynamic missing isolation cell check (`umi`) for the `top/inst1` element.

### Case 3

```
pa msg -enable -checkIds {umi} -sources {PD1}  
  
pa msg -disable -checkIds {umi} -elements {top/inst1} -transitive TRUE
```

Since `-elements` takes precedence over `-sources`, the tool disables the dynamic missing isolation check (`umi`) for the `top/inst1` element.

Hence, if there is a missing isolation cell from the `PD1-to-PD2` crossing on the element `{top/inst1}`, then the tool does not report it, because it matches the specified `-element` argument (in the second command)—even if it lies inside the source scope `PD1`, for which the first command enable the checks. For crossings that do not start with `PD1`, the tool applies the second command according to the precedence order.

## Related Topics

[pa\\_checks](#)

## Pathname Convention in Arguments

Use either the absolute or the relative pathname to specify the design or UPF objects in the arguments of the `pa_checks` and `pa_msg` commands.

- **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 /dut/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](#)

[set\\_scope](#)

## 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 check results are not available for ModelSim SE.

**Viewing Static Checks in the Results Analysis Window** ..... 126

**GUI Elements of the Results Analysis Window for Static Checks** ..... 127

## 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 following command:

```
vopt -pa_checks=s
```

The tool generates the *report.static.tdb* file 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, which contains your static check results.

3. Select a predefined 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 predefined 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. See “[Dataflow Window](#)” in the *User’s Manual*. This is available only if the design is loaded for simulation.

## Related Topics

[Static RTL Checks](#)

# GUI Elements of the Results Analysis Window for Static Checks

The Results Analysis window provides features that are specific to loading a *report.static.tdb* file.

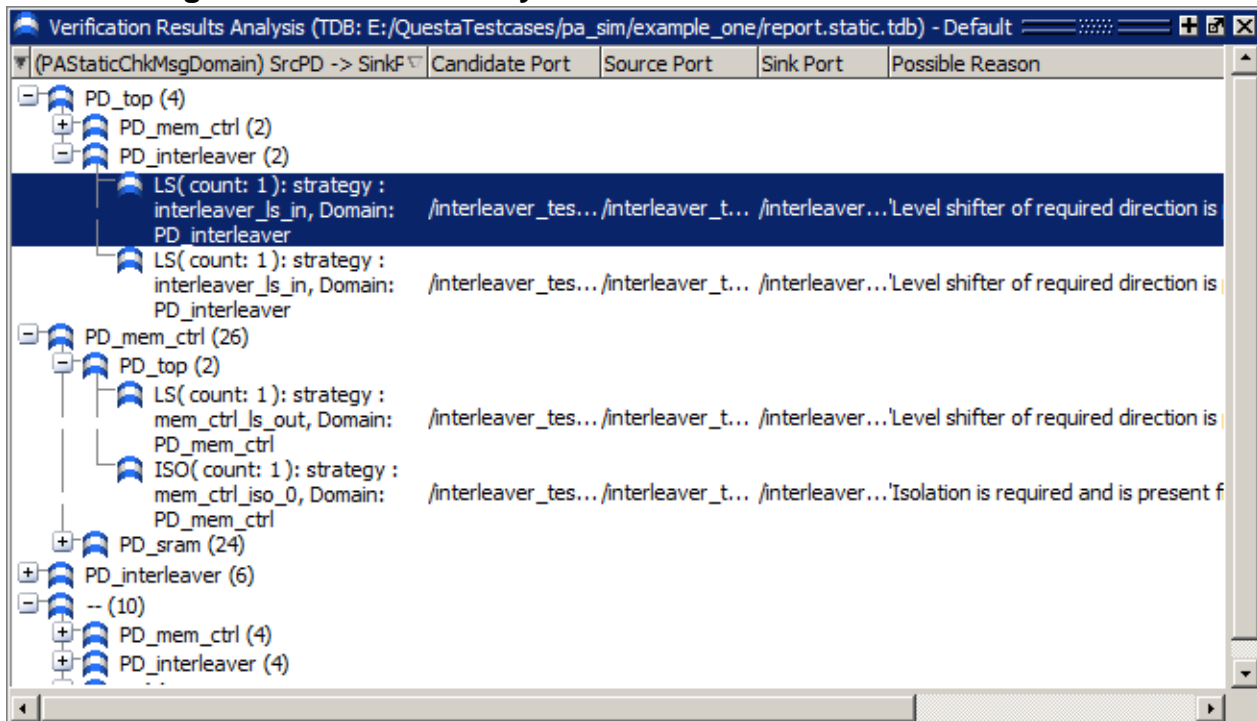
### Restriction



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 information.

**Figure 4-2. Results Analysis Window With Static Check Results**



## Power Aware Specific Columns

The following columns are specific to the *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.

## Predefined Analysis Questions

After loading the TDB file, you can choose between two predefined configurations of the window by selecting an Analysis Question:

- Show Power Aware Static Check Messages - Domain Wise

The results are sorted in order of the source power domain, sink power domain, and then message.

- Show Power Aware Static Check Messages - Strategy Wise



The results are sorted in order of the cell type (such as isolation or level shifter), violation type (such as valid or not analyzed), source power domain, and sink power domain.



# Chapter 5

## Power Aware Reports and Messages

---




Power Aware simulation generates various reports and messages that you use to analyze and validate the power intent of your design. The tool generates the reports during optimization (with the vopt -pa\_genrpt command), and the messages during runtime.

|                                   |            |
|-----------------------------------|------------|
| <b>Power Aware Reports .....</b>  | <b>132</b> |
| <b>Power Aware Messages .....</b> | <b>136</b> |




## Power Aware Reports

During optimization, the tool generates various Power Aware reports (at the *pa\_reports/* directory in the current working directory) depending upon the values that you provide to the *vopt -pa\_genrpt* argument.



**Table 5-1. Power Aware Reports**

| Report   | Description  | vopt Argument   |
|--|--|---|
| Architecture Report<br>( <i>report.pa.txt</i> )            | Contains information related to the Power Aware architecture that results from the power intent defined in the UPF file, when applied to the design.           | <i>-pa_genrpt=pa [+b]</i><br> <b>Note:</b> Specify the argument value “b” with “pa” to include bitwise expanded information in the report.   |
| Ack Port Driver Report<br>( <i>report.ack.driver.txt</i> ) | Contains information related to the acknowledge (ack) port drivers of the power switch cells.  | None<br> <b>Note:</b> The report is generated 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 tool does not report the ack port. In this case, the ack port is driven by the logic specified in the IEEE Std 1801. |
| Connection Report<br>( <i>report.connection.txt</i> )      | Contains information related to the connections between your UPF, HDL, and Liberty files.  | <i>-pa_genrpt=conn</i>  |
| Design Element Report<br>( <i>report.de.txt</i> )          | Contains information related to the elements present in your design, and the corresponding Power Aware information.  | <i>-pa_genrpt=de [+b]</i><br> <b>Note:</b> Specify the argument value “b” with “pa” to include bitwise expanded information in the report.   |
| Macro Cell Report<br>( <i>report.cell.txt</i> )            | Contains information related to the macro cells present in your Power Aware design and the connectivity of the liberty cell related to the macro cell, if any. | <i>-pa_genrpt=cell</i>  |

**Table 5-1. Power Aware Reports (cont.)**

| Report  | Description  | vopt Argument   |
|---|--|---|
| Missing Liberty Cell Report<br>( <i>report.missingliberty.txt</i> )       | Contains a list of cells that do not have any Liberty cell definition.   | -pa_libertyfiles=<liberty_filename>  <br>[-pa_loadlibertydb=<database_pathname>]  |
| Multi-Rail Cell Report<br>( <i>report.multiRail.cells.txt</i> )           | Contains a list of multi-rail cells that do not have any explicit UPF connections.   | None<br> <b>Note:</b> The report is generated only when the design has at least one multi-rail cell that does not have any explicit UPF connection.  |
| Non-Retention Synchronous Flop Report<br>( <i>report.nretsyncff.txt</i> ) | Contains a list of hierarchical paths of flip-flops that are non-retention in nature and do not have an asynchronous control signal.   | -pa_checks=npu<br> <b>Note:</b> The report is generated only when the design has at least one flip-flop that is non-retention in nature, and does not have an asynchronous control signal.                             |
| Power Cell Report<br>( <i>report.powercells.txt</i> )                     | 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.                      | [-pa_enable=conninsidemacro]  <br>[-pa_enable=autoconns]<br> <b>Note:</b> The report is generated only when there is at least one unconnected cell present inside a macro cell or an unconnected level shifter cell. |
| PST Analysis Report<br>( <i>report.pst.txt</i> )                          | Contains information related to the PSTs in your design, including the composition details.<br><br>The report contains tags (strings within brackets), which also appear in the static check report. | -pa_genrpt=pst  |
| Source Sink Path Report<br>( <i>report.srcsink.txt</i> )                  | Contains information related to the source sink-paths that are analyzed for static checks.<br><br>See “Path-ID Link”.  | -pa_genrpt=srcsink  |

**Table 5-1. Power Aware Reports (cont.)**

| Report  | Description  | vopt Argument   |
|---|--|---|
| Static Check Report<br>( <i>report.static.txt</i> )                             | <p>Contains information related to the static checks performed on your design.</p> <p>A database file (<i>report.static.tdb</i>) is also generated, which enables you to use the Results Analysis window to view the static check results.</p> <p>See “Path-ID Link”.</p> <p> <b>Restriction:</b> The database file is not available for ModelSim SE.</p> | -pa_checks=s  |
| Supply Network Initialization Report<br>( <i>report.supplynetworkinit.txt</i> ) | <p>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 tool.</p>   | -pa_genrpt=supplynetworkinit  |
| Unassociated Cell Report<br>( <i>report.unassociated.cells.txt</i> )            | <p>Contains a list of power management cells that are not associated with any UPF strategy.</p>  | <p>None</p> <p> <b>Note:</b> The report is generated only when there is at least one power management cell that is not associated with any UPF strategy.</p> |

#### Path-ID Link

The static check report, *report.static.txt*, contains a Path-ID link, `PATH_<number>`:

```
1.1. Source port: /top_vl/q1 [LowConn] to Sink port: /top_vl/q1 [HighConn], \
Total 3 Not analyzed isolation cells [Total Crossings: 3, Shared Crossings:
1.1.1. Inferred type: ISO_NOT_ANALYZED, count: 3
Candidate Port: /top_vl/q1, Strategy: PD_0_ISO, Domain: /top_vl/PD_0
Possible reason: 'For crossing (/top_vl/PD_0) => (--) : either source po
d in any power domain'
Analysis link: [PD2 to PD1]
Path-ID link: [PATH_5]
```

The source sink report, *report.srcsink.txt*, contains the details of the `PATH_<number>`:

```
[PATH_5]
-----

[Src Supply ( /top_vl/PD_0 )]
[UPF Cell : /top_vl/q1 [0:2]]
[ Cand Port ( /top_vl/q1 ),]
[ isolation ( PD_0_ISO ),]
[Port : /top_vl/q1 [0:2]]
[ ( Output )]
[Sink Supply ( -- )]
```

## Generating Power Aware Reports

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 “[Table 5-1](#)”.
    - If you do not specify a value to the `-pa_genrpt` argument, all reports, except the Source Sink Path Report, are generated.

#### Tip

**i** To generate the Source Sink Path Report, use the following command:

```
vopt -pa_genrpt=srcsink
```

- 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 Messages

During runtime, the tool automatically generates Power Aware messages at the Transcript window.

The 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

For example:

```
# ** 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.
```



# 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>137</b> |
| <b>Power Aware Coverage Flow .....</b>                             | <b>140</b> |
| Generating Coverage Reports Using Power Aware Simulation .....     | 141        |
| Generating Coverage Reports Using Non-Power Aware Simulation ..... | 143        |
| Analyzing Coverage Using the Power Aware Test Plan .....           | 145        |
| <b>Coverage Support of UPF Objects .....</b>                       | <b>149</b> |
| <b>Excluding Objects From Power Aware Coverage .....</b>           | <b>152</b> |

## Power Aware Coverage Overview

Configure Power Aware simulation or non-Power Aware simulation to analyze your Power Aware and non-Power Aware coverage.

### Coverage Using Power Aware Simulation

Configure your Power Aware simulation to generate the following:

- **Power Aware Coverage Report** — Contains [Power Aware Coverage Details](#)—dynamic check coverage and power states and transition coverage.
- **Combined Coverage Report** — Contains Power Aware and non-Power Aware coverage details.
  - Power Aware coverage details — Contains dynamic check coverage and power states and transition coverage details.
  - Non-Power Aware coverage details — Contains functional and structural (toggle only) coverage details. However, you can enable complete functional and structural (statement, expression, FSM, branch, and toggle) coverage details using the following command:
- **Power Aware Test Plan** — Contains Power Aware coverage details in an XML format. You can also view the test plan in the Verification Management Tracker window of the simulator.

```
vopt -pa_enable=codecoverage
```

### Note



To add non-Power Aware coverage details in the test plan, use the +cover or -pa\_enable=codecoverage argument in the vopt command. See “[Analyzing Coverage Using the Power Aware Test Plan](#)”.

---

## Coverage Using Non-Power Aware Simulation

Configure your non-Power Aware simulation to generate the following:

- **Power Aware Coverage Report** — Contains [Power Aware Coverage Details](#)— dynamic check coverage and power states and transition coverage.
- **Combined Coverage Report** — Contains Power Aware and non-Power Aware coverage details.
  - Power Aware coverage details — Contains dynamic check coverage and power states and transition coverage details.
  - Non-Power Aware coverage details — Contains complete functional and structural (statement, expression, FSM, branch, and toggle) coverage details.

## Power Aware Coverage Details

Power Aware coverage contains the following details:

- **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 stimulus 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](#)”.

### Tip



For more information on coverage, refer to the *Questa SIM GUI Reference Manual*.

---

## Reporting Transitions in the Coverage Reports

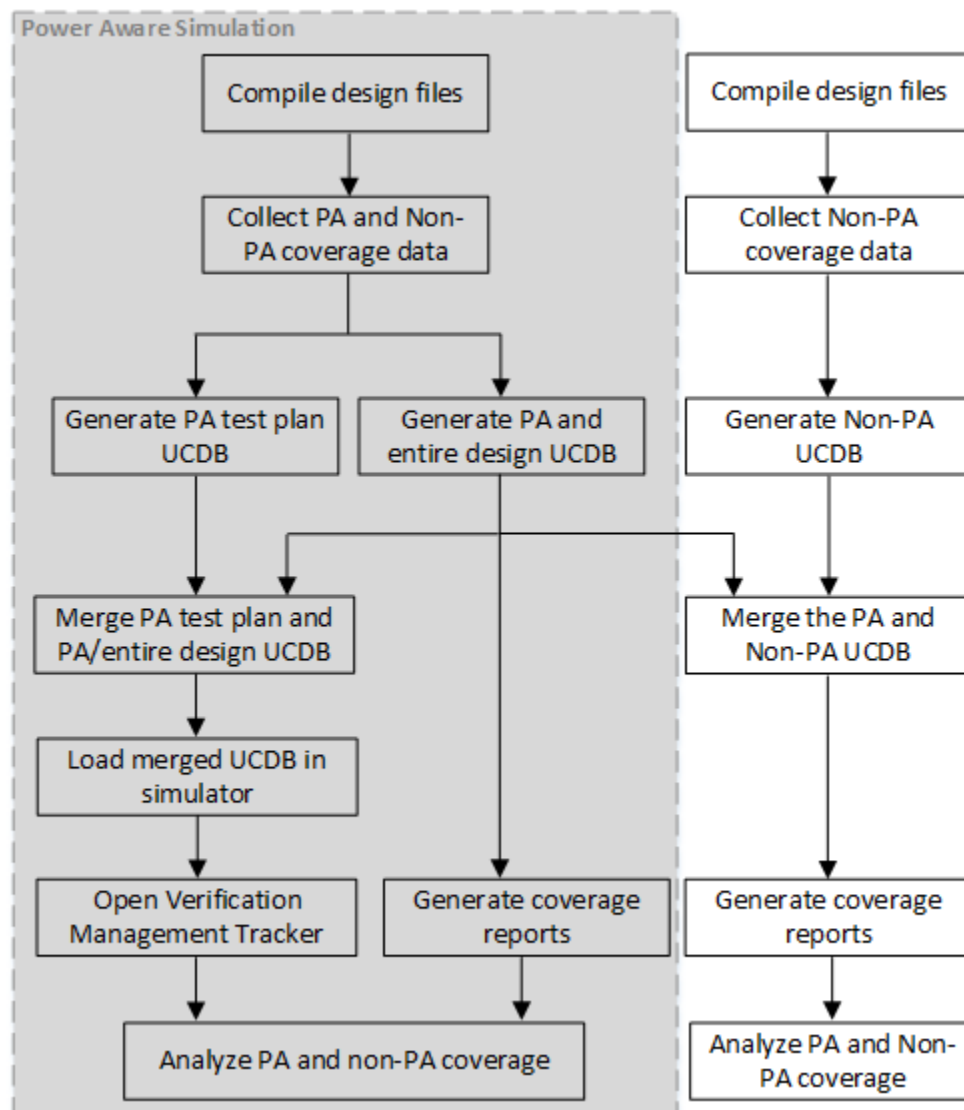
The tool follows the listed semantics for reporting transitions in the Power Aware and combined coverage reports:

- If the UPF file does not contain the [describe\\_state\\_transition](#) or [add\\_state\\_transition](#) command, the tool 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 tool 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.

## Power Aware Coverage Flow

Generate coverage reports and a test plan to analyze your Power Aware and non-Power Aware coverage.



`<install_dir>/examples/pa_sim/PA_coverage/`

For more information on the example, view the *README* file provided in the directory.

|   |            |
|---|------------|
| <b>Generating Coverage Reports Using Power Aware Simulation .....</b>     | <b>141</b> |
| <b>Generating Coverage Reports Using Non-Power Aware Simulation .....</b> | <b>143</b> |
| <b>Analyzing Coverage Using the Power Aware Test Plan .....</b>           | <b>145</b> |

# Generating Coverage Reports Using Power Aware Simulation

Configure the standard Power Aware simulation to generate Power Aware and combined coverage report. Generate these reports in either command line or from the GUI.

## Procedure

1. Compile using your existing command:

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

2. Optimize using your existing vopt command with the following arguments:

```
vopt -pa_coverage +cover [-pa_enable=codecoverage] [other vopt args]
```

- **-pa\_coverage** — Collects Power Aware coverage data.
  - 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, are enabled.


For example, the following command collects Power Aware coverage data of dynamic checks:

```
vopt -pa_coverage=checks -pa_checks [other vopt args]
```

To enable a specific dynamic check, specify values to the -pa\_checks argument of the vopt command. See [“Dynamic Checks”](#).

---

### Tip

 For finer control on the coverage data, see [““add\\_state\\_transition” on page 243”](#), [““describe\\_state\\_transition” on page 262”](#), and [““describe\\_state\\_cross\\_coverage” on page 313”](#).

---

- 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 [other vopt args]
```

- To enable complete cross coverage, use the following command:


```
vopt -pa_coverage=powerstate+crossdontcare [other vopt args]
```

See [“Cross Coverage of Power States”](#).

- **-pa\_enable=codecoverage** — (Optional) Collects non-Power Aware coverage data, which contains functional and structural (statement, expression, FSM, branch, and toggle) coverage details.

---

**Note**

 To use the `-pa_enable=codecoverage` argument, you need to enable coverage using the `+cover` argument.

---

3. Simulate using your existing `vsim` command:

```
vsim -pa -coverage [other vsim args]
```

4. Run your simulation in the simulator:

```
run [other run args]
```

5. Generate the entire design UCDB, for example, *all.ucdb*, which contains Power Aware and non-Power Aware information:

```
coverage save all.ucdb
```

6. Generate a Power Aware UCDB, for example, *pa.ucdb*:

```
coverage save -pa pa.ucdb
```

7. (Optional) Post-simulation load the Power Aware UCDB or entire design UCDB in the simulator:

```
vsim -viewcov pa.ucdb  
vsim -viewcov all.ucdb
```

---

**Note**

 You do not need to load the UCDB during live simulation.

---

8. (Optional) To exclude a power state or transition from coverage reports, use the following command.

State exclusion:

```
coverage exclude -scope /alu_tester/dut -pstate PD_TOP.simMode  
NORMAL
```

Transition exclusion:


```
coverage exclude -scope /alu_tester/dut -ptrans PD_TOP.primary  
{PD_TB_always_on -> *}
```

9. Generate the Power Aware coverage report:

```
coverage report -pa [-verbose] [-html]
```

---

**Tip**

 To generate coverage reports on the command line, use the `vcover` report command instead of the `coverage report` command.

---

### Note

 (GUI only) Choose **Tools > Coverage Report** to generate the reports.

---

10. Generate the combined coverage report, which contains both Power Aware and non-Power Aware coverage object:

```
coverage report -pacombined [-verbose] [-html]
```

11. (Optional) To list the Power Aware objects that are excluded from the coverage statistics, use the following command:

```
coverage report -excluded -pa
```

## Related Topics

[vopt](#)

[coverage exclude](#)

[coverage report](#)

[vcover report](#)

[Excluding Objects From Power Aware Coverage](#)

# Generating Coverage Reports Using Non-Power Aware Simulation

Configure non-Power Aware simulation with Power Aware simulation to generate Power Aware and combined coverage report. Generate these reports in either command line interface or from the GUI.

## Prerequisites

- Complete non-Power Aware simulation. See “[Usage Flow for Code Coverage Collection](#)” in the *Questa SIM User’s Manual*.
- Generate a non-Power Aware UCDB, for example, *nonpa.ucdb*. See “[Saving Code Coverage in the UCDB](#)” in the *Questa SIM User’s Manual*.
- Complete Power Aware simulation, and generate a Power Aware UCDB, for example, *pa.ucdb*. See step 1 to 6 in “[Generating Coverage Reports Using Power Aware Simulation](#)”.

## Procedure

1. Merge the Power Aware UCDB and the non-Power Aware UCDB to generate the merged UCDB at the command line:

```
vcover merge merged.ucdb pa.ucdb nonpa.ucdb
```

2. Load the merged UCDB in the simulator:

```
vsim -viewcov merged.ucdb
```

3. (Optional) To exclude a power state or transition from coverage reports, use the following command.

State exclusion:

```
coverage exclude -scope /top/pd_alu -pstate PD_SYS1 SLEEP
```

Transition exclusion:

```
coverage exclude -scope /top/pd_alu -ptrans PD_SYS2 {* -> slp}
```

---

#### Tip



To generate coverage reports on the command line, use the `vcover` report command instead of the `coverage report` command.

---

4. Generate the Power Aware coverage report:

```
coverage report -pa [-verbose] [-html]
```

---

#### Note



(GUI only) Alternatively, choose **Tools > Coverage Report** to generate the reports.

---

5. Generate the combined coverage report:

```
coverage report -pacombined [-details] [-verbose]  
[-assert] [-file <filename>] [-xml -file <filename>] [-html]
```

6. (Optional) To list the Power Aware objects that are excluded from the coverage statistics, use the following command:

```
coverage report -excluded -pa
```

## Related Topics

[vopt](#)

[coverage exclude](#)

[coverage report](#)

[vcover report](#)

[Excluding Objects From Power Aware Coverage](#)



# 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 simulator to analyze Power Aware coverage details. Also, you can add non-Power Aware coverage details in the test plan.

## Restriction



ModelSim SE does not support the test plan generation.

## Procedure

1. Compile using your existing command:

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

2. Optimize using your existing vopt command with the following arguments:

```
vopt -pa_coverage -pa_enable=autotestplan [+cover]
[-pa_enable=codecoverage] [other vopt args]
```

- **-pa\_coverage** — Collects Power Aware coverage data.
  - 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, are enabled.

For example, the following command collects Power Aware coverage data of dynamic checks:

```
vopt -pa_coverage=checks [other vopt args]
```

To enable a specific dynamic check, specify values to the -pa\_checks argument of the vopt command. See [“Dynamic Checks”](#).

## Tip



For finer control on the coverage data, see [“add\\_state\\_transition”](#), [“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 [other vopt args]
```

- To enable complete cross coverage, specify the following argument with your vopt command. See [“Cross Coverage of Power States”](#).

```
vopt -pa_coverage=powerstate+crossdontcare [other vopt args]
```

- **-pa\_enable=autotestplan** — 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+pdhiertestplan
```

- **-pa\_enable=codecoverage** — (Optional) Collects non-Power Aware coverage data, which contains functional and structural (statement, expression, FSM, branch, and toggle) coverage details.

---

**Note**

---



To use the `-pa_enable=codecoverage` argument, you need to enable coverage using the `+cover` argument.

---

3. Simulate using your existing `vsim` command:

```
vsim <optimized_output> -pa -coverage [other vsim args]
```

4. Run your simulation in the simulator:

```
run [other run args]
```

5. Generate the entire design UCDB, for example, *all.ucdb*, which contains Power Aware and non-Power Aware information:

```
coverage save all.ucdb
```

6. (Optional) Generate the Power Aware design UCDB, for example, *pa.ucdb*, which contains Power Aware information:

```
coverage save -pa pa.ucdb
```

7. Generate the Power Aware test plan UCDB, *QuestaPowerAwareTestplan.ucdb*:

```
pa autotestplan [-format] [-filename]
```

The tool generates test plan at the *pa\_reports* directory.

8. Merge the Power Aware test plan UCDB, and the Power Aware UCDB or entire design UCDB to generate the merged UCDB at the command line:

```
vcover merge merged.ucdb pa_reports/QuestaPowerAwareTestplan.ucdb  
all.ucdb
```

Alternatively, merge the Power Aware test plan UCDB and the Power Aware UCDB to generate the merged UCDB at the command line:

```
vcover merge merged.ucdb pa_reports/QuestaPowerAwareTestplan.ucdb  
pa.ucdb
```

9. Load the merged UCDB, *merged.ucdb*, in the simulator:

```
vsim -viewcov merged.ucdb
```

10. Choose **View > Verification Management > Tracker** to open the Verification Management Tracker window and analyze Power Aware coverage.

#### Tip

**i** You can modify the Power Aware test plan XML file, use the `xml2ucdb` command to generate the modified test plan UCDB, and follow the steps from 8 to 10 to view the modified test plan in the Verification Management Tracker window.

## Results

The power domains in the test plan are listed in two ways: Flat View and Hierarchical View.

- **Flat View** — By default, the tool generates the flat view of the test plan, where all power domains are listed in a flat hierarchy.

**Figure 6-1. Power Aware Test Plan (Flat View) in the Verification Management Tracker Window**

| Testplan Section / Coverage Link | Type     | Coverage | % of Goal | Status | Description         |
|----------------------------------|----------|----------|-----------|--------|---------------------|
| testplan                         | Testplan | 36.99%   | 36.99%    |        |                     |
| alu_tester                       | Testplan | 36.99%   | 36.99%    |        | Design hierarchi... |
| dut                              | Testplan | 36.99%   | 36.99%    |        | Design hierarchi... |
| ALU_sw                           | Testplan | 48.61%   | 48.61%    |        | Power Switch        |
| IN_ISO_sw                        | Testplan | 9.37%    | 9.37%     |        | Power Switch        |
| IN_sw                            | Testplan | 35.41%   | 35.41%    |        | Power Switch        |
| OUT_RET_sw                       | Testplan | 9.37%    | 9.37%     |        | Power Switch        |
| OUT_sw                           | Testplan | 35.41%   | 35.41%    |        | Power Switch        |

**Figure 6-2. Power Aware Test Plan (Flat View) in an XML Format**

| #       | Section                | Description                      | Link                                | Type        | Weight | Goal |
|---------|------------------------|----------------------------------|-------------------------------------|-------------|--------|------|
| 1       | alu tester             | Design hierarchical scope        |                                     |             | 1      | 100  |
| 1.1     | dut                    | Design hierarchical scope        |                                     |             | 1      | 100  |
| 1.1.1   | OUT RET sw             | Power Switch                     |                                     |             | 1      | 100  |
| 1.1.1.1 | Power State Coverage   | To cover Power States            | OUT RET sw STATE COVERAGE           | Valu tester | COVERG | 100  |
| 1.1.1.2 | Power State Transition | To cover Power State Transitions | OUT RET sw TRANSITION COVERAGE      | Valu        | COVERG | 100  |
| 1.1.1.3 | Power Switch Control   | Check to catch corruption on     | Valu tester/dut/QPA_CTRL_CRPT_CHK_2 | ASSERTI     | 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.

**Figure 6-3. Power Aware Test Plan (Hierarchical View) in the Verification Management Tracker Window**

| Testplan Section / Coverage Link | Type     | Coverage | % of Goal | Status | Description        |
|----------------------------------|----------|----------|-----------|--------|--------------------|
| testplan                         | Testplan | 33.2%    | 33.2%     |        |                    |
| Power Domain Hier Tree Coverage  | Testplan | 33.2%    | 33.2%     |        | Power Domain ...   |
| PD_TOP                           | Testplan | 33.2%    | 33.2%     |        | Power domain       |
| Sub Power Domains                | Testplan | 52.43%.. | 52.43%..  |        | Sub-Power Dom...   |
| PD_OUT                           | Testplan | 64.93%.. | 64.93%..  |        | Power domain       |
| PD_IN                            | Testplan | 35.15%.. | 35.15%..  |        | Power domain       |
| PD_ALU                           | Testplan | 57.22%.. | 57.22%..  |        | Power domain       |
| simMode                          | Testplan | 100%     | 100%      |        | Simulation Mode    |
| primary                          | Testplan | 71.66%.. | 71.66%..  |        | Supply set         |
| Power State Transition C...      | Testplan | 43.33%.. | 43.33%..  |        | To cover Power ... |

**Figure 6-4. 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.Valu tester/d | COVERGROUP | 1      | 100  |
| 1.0.1.2 | SimMode State Transition | To cover Simulation State Transitions | simMode TRANSITION COVERAGE.Valu     | 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.Valu tester/d | COVERGROUP | 1      | 100  |
| 1.0.2.2 | Power State Transition   | To cover Power State Transitions      | primary TRANSITION COVERAGE.Valu te  | COVERGROUP | 1      | 100  |

**pa\_coverageinfo** — The tool 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 tool creates a state named *Undefined*, which becomes active. This undefined state is added by the tool 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/LibreOffice Calc Extension](#)

[Questa Testplan Creation Using Excel Add-In](#)

[pa autotestplan](#)

## Coverage Support of UPF Objects

Power Aware simulation 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, retention strategies, control signals, and simstates. The tool also supports cross coverage of power states, which refers to the system-level coverage that is represented using the cross of various power states.

**Table 6-1. Coverage Support of UPF Objects**

| UPF Object         | -pa_coverage Value | Description   |
|--------------------|--------------------|---|
| Isolation strategy | iso                | Enables coverage of the following: <ul style="list-style-type: none"><li>• Predefined states and transitions of the simstates of the isolation supply set.</li><li>• Predefined states and transitions of the isolation enable signal:<ul style="list-style-type: none"><li>• Predefined states — ACTIVE_LEVEL, INACTIVE, ACTIVE_X and ACTIVE_Z</li><li>• Predefined transitions — HIGH_TO_LOW, and LOW_TO_HIGH</li></ul></li></ul> |
| Port and net       | portpstate         | Enables coverage of user-defined states and transitions of ports and nets.  |
|                    | implicitportnet    | Enables coverage of the implicitly created state (by the tool) that is used in the -supply_expr option of the add_power_state command.  |
| Power domain       | powerstate         | Enables coverage of user-defined states and transitions of power domains. See <a href="#">“Cross Coverage of Power States”</a> .  |
|                    | simMode            | Enables coverage of states and transitions of simstates of the primary supply set.  |
| Power state group  | powerstate         | Enables coverage of user-defined states and transitions of power state groups.  |

**Table 6-1. Coverage Support of UPF Objects (cont.)**

| UPF Object          | -pa_coverage Value | Description   |
|---------------------|--------------------|---|
| Power switch        | switch             | Enables coverage of the following: <ul style="list-style-type: none"> <li>• Predefined states and transitions of the switch</li> <li>• Predefined states and transitions of the control port and acknowledge (ack) port of the switch: <ul style="list-style-type: none"> <li>• Predefined states — ACTIVE_LEVEL, INACTIVE, ACTIVE_X and ACTIVE_Z</li> <li>• Predefined transitions — HIGH_TO_LOW, and LOW_TO_HIGH</li> </ul> </li> </ul> |
| PST                 | portpststate       | Enables coverage of user-defined states and transitions of PSTs   |
| Retention strategy  | ret                | Enables coverage of the following: <ul style="list-style-type: none"> <li>• Predefined states and transitions of the simstates of the retention supply set</li> <li>• Predefined states and transitions of the save and restore events: <ul style="list-style-type: none"> <li>• Predefined states — ACTIVE_LEVEL, and INACTIVE</li> <li>• Predefined transitions — HIGH_TO_LOW, and LOW_TO_HIGH</li> </ul> </li> </ul>                   |
| Supply set          | powerstate         | Enables coverage of user-defined and predefined states and transitions of supply sets.  |
| Supply set simstate | powerstate         | Enables coverage of predefined states and transitions of supply sets simstates.   |

## Cross Coverage of Power States

The tool 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 tool considers only power states of power domains for cross coverage.
- The tool 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_crosscoveragedepth <integer>` argument with your `vopt` command.
- To disable cross coverage, specify the `-pa_coverageoff=crosscov` argument with your `vopt` command.

---

**Tip**

 To view the results of cross coverage for a specified list of power domains in a design, use the [describe\\_state\\_cross\\_coverage](#) 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 tool 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 tool 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
```

**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 tool 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+crossdntcare
```

### Related Topics

[Power Aware Coverage Overview](#)

## Excluding Objects From Power Aware Coverage

Use the coverage exclude command to exclude any object within a particular hierarchical path from coverage statistics and coverage reports.

### Examples

- To exclude a power state, SLEEP, of a power domain, PD\_SYS1, found within the instance /top/pd\_alu, use the following command:

```
coverage exclude -scope /top/pd_alu -pstate PD_SYS1 SLEEP
```

- To exclude a power transition of a power domain, PD\_SYS2, found within the instance top/pd\_alu, use the following command:

```
coverage exclude -scope /top/pd_alu -ptrans PD_SYS2 { * -> slp }
```

- To exclude a power state of the primary supply set of a power domain, PD\_SYS2, found within the instance /top/pd\_alu, use the following command:

```
coverage exclude -scope /top/pd_alu -pstate PD_SYS2.primary  
DEFAULT_CORRUPT
```

- To exclude a simMode coverage state of the primary supply of a power domain, PD\_SYS2, found within the instance /top/dut, use the following command:

```
coverage exclude -scope /top/dut -pstate PD_SYS2.simMode CORRUPT
```

- To exclude a simMode coverage state of the isolation strategy supply of a power domain, PD\_SYS2, found within the instance /top/dut, use the following command:

```
coverage exclude -scope /top/dut -pstate PD_SYS2.PD_SYS2_iso.simMode  
CORRUPT
```

- To exclude an isolation control signal state of a power domain, PD\_SYS2, found within the instance /top/dut, use the following command:

```
coverage exclude -scope /top/dut -pstate PD_SYS2.PD_SYS2_iso.ISO_SIG  
ACTIVE_X
```



- To exclude a power switch control signal state of a power domain, PD\_SYS2, found within the instance /top/dut, use the following command:

```
coverage exclude -scope /top/dut -pstate psw_1.pwr_ctrl ACTIVE_X
```



# Chapter 7

## Power Aware Information Model

---

Power Aware information model is a data model to capture the power-management information, which is the result of application of UPF, Liberty, and HDL commands on your design.

The information is captured in a standard form, and you access the information model objects and its properties with the following:

- [Tcl Interface](#) — Uses the UPF 3.1 Tcl APIs in a Tcl script or UPF file.
- [HDL Interface](#) — Uses the UPF 3.1 HDL package functions in a test bench or simulation model.

---

### Note



Power Aware simulation uses the UPF 3.1 version of the UPF packages and supports the same set of UPF package definitions.

---

|  |            |
|--|------------|
| <b>Tcl Interface</b> .....                               | <b>156</b> |
| Running the Tcl APIs .....                               | 156        |
| Supported Tcl APIs .....                                 | 157        |
| <b>HDL Interface</b> .....                               | <b>160</b> |
| Supported HDL Package Functions .....                    | 160        |
| Supported Native HDL Representation .....                | 162        |
| Power Aware Coverage Using HDL Package Functions .....   | 163        |
| Power Aware Assertions Using HDL Package Functions ..... | 165        |

## Tcl Interface

The Tcl interface enables you to access the Power Aware information model objects and its properties in a Tcl script or UPF file using the UPF 3.1 Tcl APIs.

The UPF 3.1 Tcl APIs are defined in the IEEE Std 1801-2018, Section 11.1.2. These APIs are also referred as UPF query commands. You can use the Tcl APIs with any previous version of the UPF standard (such as UPF 3.0 or UPF 2.1). Although the output format and properties of the commands are dependent on the UPF 3.1 standard.

|                                   |            |
|-----------------------------------|------------|
| <b>Running the Tcl APIs .....</b> | <b>156</b> |
| <b>Supported Tcl APIs .....</b>   | <b>157</b> |

## Running the Tcl APIs

Run the UPF 3.1 Tcl APIs 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 Three-Step Flow](#)” topic.

### Procedure

1. Activate the interactive mode:

```
vopt -pa_interactive <design_top_module>
```

2. Create the Power Aware information model database, *design.bin.padb*, in the current working directory:

```
vopt -pa_upf <upf_file> -o <optimized_output> -pa_dumpimdb  
<other_arguments>
```

3. Run the Tcl APIs to query from the database.


For example:

```
PA> upf_query_object_properties /alu/dut/PD_TOP  
PA> upf_query_object_type /alu/dut/PD_RAM
```

## Supported Tcl APIs

Power Aware simulation supports the syntax and semantics of the UPF 3.1 Tcl APIs, also called as UPF query commands, as defined by IEEE Std 1801-2018, Section 11.1.2.

### Note

 To query a supply or logic net whose name is the same as that of a supply or a logic port, use the suffix @upfSupplyNetT or @upfLogicNetT string, respectively:

```
upf_query_object_properties /tb/logic_netport@upfLogicNetT
upf_query_object_type /tb/logic_net1@upfSupplyNetT
```

**Table 7-1. 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 Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |
| UPF 1.0     | Yes     |

## 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 Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |
| UPF 1.0     | Yes     |

## 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 Version | Support | Comment  |
|-------------|---------|--|
| UPF 3.1     | Yes     | Supported option (not a part of IEEE Std 1801): -verbose<br>See “ <a href="#">Usage Notes</a> ”. |
| UPF 3.0     | Yes     |  |
| UPF 2.1     | Yes     |  |
| UPF 2.0     | Yes     |  |
| UPF 1.0     | Yes     |  |

## Usage Notes

The -verbose option 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)`

---

### Note



The option is not a part of the IEEE Std 1801.

---

## upf\_query\_object\_type

Returns the type of an object in the Power Aware database.

## Support for UPF Standard

| UPF Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |
| UPF 1.0     | Yes     |

## HDL Interface

The HDL interface enables you to access the Power Aware information model objects and its properties directly in a test bench or simulation model using the UPF 3.1 HDL package functions.


|   |            |
|---|------------|
| <b>Supported HDL Package Functions.....</b>                     | <b>160</b> |
| <b>Supported Native HDL Representation.....</b>                 | <b>162</b> |
| <b>Power Aware Coverage Using HDL Package Functions .....</b>   | <b>163</b> |
| <b>Power Aware Assertions Using HDL Package Functions .....</b> | <b>165</b> |

## Supported HDL Package Functions


The simulator supports the syntax and semantics of various HDL package functions defined by IEEE Std 1801-2018, Section 11.2.3.

### Supported UPF 3.1 HDL Package Functions For SystemVerilog

#### Note

 To use the UPF 3.1 HDL package functions, create the Power Aware information model database using the following command. However, if you use a function that is present in UPF 2.0 package functions, you do not need to create the database.

```
vopt -pa_dumpimdb <other arguments>
```

| 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.<br> <b>Note:</b> The option is not a part of the IEEE Std 1801. |
|                       | upf_get_value_int        |   |
|                       | upf_get_value_str        |   |



| Function Type          | Function Name               | Comment   |
|------------------------|-----------------------------|---|
| Immediate write access | set_supply_state            |   |
|                        | supply_on                   |   |
|                        | supply_off                  |   |
|                        | supply_partial_on           |   |
| Continuous access      | upf_create_object_mirror    | Write access is not supported.                  |
| HDL access             | upf_get_handle_by_name      |   |
|                        | upf_iter_get_next           |   |
|                        | upf_object_in_class         | See “ <a href="#">Unsupported Classes</a> ”.    |
|                        | upf_query_object_properties | See “ <a href="#">Unsupported Properties</a> ”. |
| Utility                | upf_query_object_pathname   |   |
|                        | upf_query_object_type       |   |

**Table 7-2. Unsupported Classes**

|                     |
|---------------------|
| upfCompositeDomainT |
|---------------------|

**Table 7-3. Unsupported Properties**

|                        |                           |
|------------------------|---------------------------|
| UPF_IS_USE_EQUIVALENCE | UPF_PD_STATE_TRANSITIONS  |
| UPF_LOGIC_REFS         | UPF_REF_KIND              |
| UPF_NAME_PREFIX        | UPF_REF_OBJECT            |
| UPF_NAME_SUFFIX        | UPF_RETENTION_PARAMETERS  |
| UPF_NEXT_EXTENT        | UPF_ROOT_DRIVER           |
| UPF_NORMALIZED_BITS    | UPF_SLICE_BITS            |
| UPF_LOGIC_REFS         | UPF_SMALLEST_ATOMIC_SLICE |
| UPF_NAME_PREFIX        | UPF_SS_TRANSITIONS        |
| UPF_NAME_SUFFIX        | UPF_SUBDOMAINS            |
| UPF_NEXT_EXTENT        | UPF_UPPER_BOUNDARY        |
| UPF_NORMALIZED_BITS    |                           |

## Supported UPF 2.0 HDL Package Functions For SystemVerilog and VHDL

### Note



To use the UPF 2.0 HDL package functions, you do not need to create the Power Aware information model database.

---

| Function Name       |
|---------------------|
| supply_on           |
| supply_off          |
| supply_partial_on   |
| get_supply_value    |
| get_supply_voltage  |
| get_supply_on_state |
| get_supply_state    |

## Supported Native HDL Representation

The simulator supports various native HDL representation defined by IEEE Std 1801.

### Note



To use the native HDL representation, create the Power Aware information model database using the following command:

```
vopt -pa_dumpimdb <other arguments>
```

---

| Type name         | SystemVerilog  |
|-------------------|--|
| upfBooleanObjT    | struct packed {<br>upfHandleT handle;<br>upfBooleanT current_value;<br>} upfBooleanObjT                                      |
| upfPdSsObjT       | struct packed {<br>upfHandleT handle;<br>upfPowerStateObjT current_state;<br>upfSimstateE current_simstate;<br>} upfPdSsObjT |
| upfPowerStateObjT | struct packed {<br>upfHandleT handle;<br>upfBooleanT is_active;<br>} upfPowerStateObjT                                       |

| Type name     | SystemVerilog   |
|---------------|---|
| upfSupplyObjT | <pre>struct packed {     upfHandleT handle;     upfSupplyTypeT current_value; } upfSupplyObjT</pre> |

## Power Aware Coverage Using HDL Package Functions

Use the UPF 3.1 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 test bench.

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 test bench. 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 test bench. 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 the Power Aware information model database of your design.

```
vopt -pa_dumpimdb <other arguments>
```

5. Load the Power Aware information model database of your design in the simulator:

```
vsim <other_arguments>
```

6. 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 SytemVerilog 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
# ...
```

## Power Aware Assertions Using HDL Package Functions

Use the UPF 3.1 HDL package functions to write Power Aware assertions in your test bench.

### Procedure

1. Declare the native HDL objects.

For example:

```
upfPowerStateObjT CAMERA_PS_ON;
upfPowerStateObjT VIDEO_PS_ON;
```

2. Get the dynamic property of the UPF object by using the continuous access HDL package functions.

For example:

```
upf_create_object_mirror ("/tb/chip_top/PD_CAMERA.ON",
"CAMERA_PS_ON");
upf_create_object_mirror ("/tb/ss1.SS_ON", "VIDEO_PS_ON");
```

3. Call the assertion module in your initial block.

For example:

```
assertionPowerState assert1(CAMERA_PS_ON.is_active,
VIDEO_PS_ON.is_active);
```

4. Create the assertion module.

For example:

```
module assertionPowerState (input state_ON_CAMERA, state_ON_VIDEO);
    reg cov_clk = 0;
    always @(state_ON_CAMERA, state_ON_VIDEO)
        cov_clk = 1'b1;
    always @(cov_clk)
        cov_clk = 1'b0;

    always@(posedge cov_clk) begin
        assert (!(state_ON_CAMERA == 1 && state_ON_VIDEO == 1))
        else
            $error($time,"----- Camera and video are both on at
the same time. Power Bug!!! -----");
        end
    endmodule
```

# Chapter 8

## Power Aware Simulation Display in Questa TKGUI

---

Use windows in the Questa SIM graphical user interface (GUI) to obtain a visual display of your Power Aware design and simulation.

|  |            |
|--|------------|
| <b>UPF Object Display</b> .....                      | <b>168</b> |
| UPF Objects .....                                    | 169        |
| Displaying UPF Objects in the GUI .....              | 169        |
| <b>Power Aware Source Window</b> .....               | <b>171</b> |
| UPF Color Scheme in the Source Window .....          | 172        |
| UPF-Specific Context Menu in the Source Window ..... | 173        |
| Show Drivers Control Bar .....                       | 174        |
| <b>Power Aware Schematic Display</b> .....           | <b>175</b> |
| Top-Down Debugging (From the Test Bench) .....       | 175        |
| Bottom-Up Debugging (From the DUT) .....             | 176        |
| <b>Schematic Window Features for Debugging</b> ..... | <b>177</b> |
| <b>Power Aware Waveform Display</b> .....            | <b>180</b> |
| Power Aware Waveform Concepts .....                  | 180        |
| Using Power Aware Highlighting .....                 | 181        |
| <b>Power State and Transition Display</b> .....      | <b>182</b> |
| Power State and Transition Concepts .....            | 182        |
| Displaying Power Aware State Machines .....          | 183        |
| Power Aware State Machine List Window .....          | 185        |
| Power Aware State Machine Viewer Window .....        | 187        |

# UPF Object Display

---

The Objects, Structure, and Wave windows provide visibility into UPF objects.

**UPF Objects** ..... 169


**Displaying UPF Objects in the GUI**..... 169



## UPF Objects

The Objects, Source, Structure, and Wave windows includes special Power Aware-specific information.

### Objects

| Object  | Window                              | Description   |
|---|-------------------------------------|---|
|  | Structure, Object, and Wave windows | Identifies UPF objects.                                 |
| PD Name column  | Structure window                    | Contains power domain information.                      |
| Design Unit column  | Structure window                    | Contains labeling of isolation and level shifter cells. |
| PAInfo column   | Objects window                      | Contains information about UPF objects.                 |
| Color codes   | Structure, Object, and Wave windows | Color coding based on different power domain.           |
| Special UPF context menu  | Source and Wave windows             | Contains UPF details.                                   |
| Filter out UPF object   | Structure window                    | Filters out 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 8-1](#)).

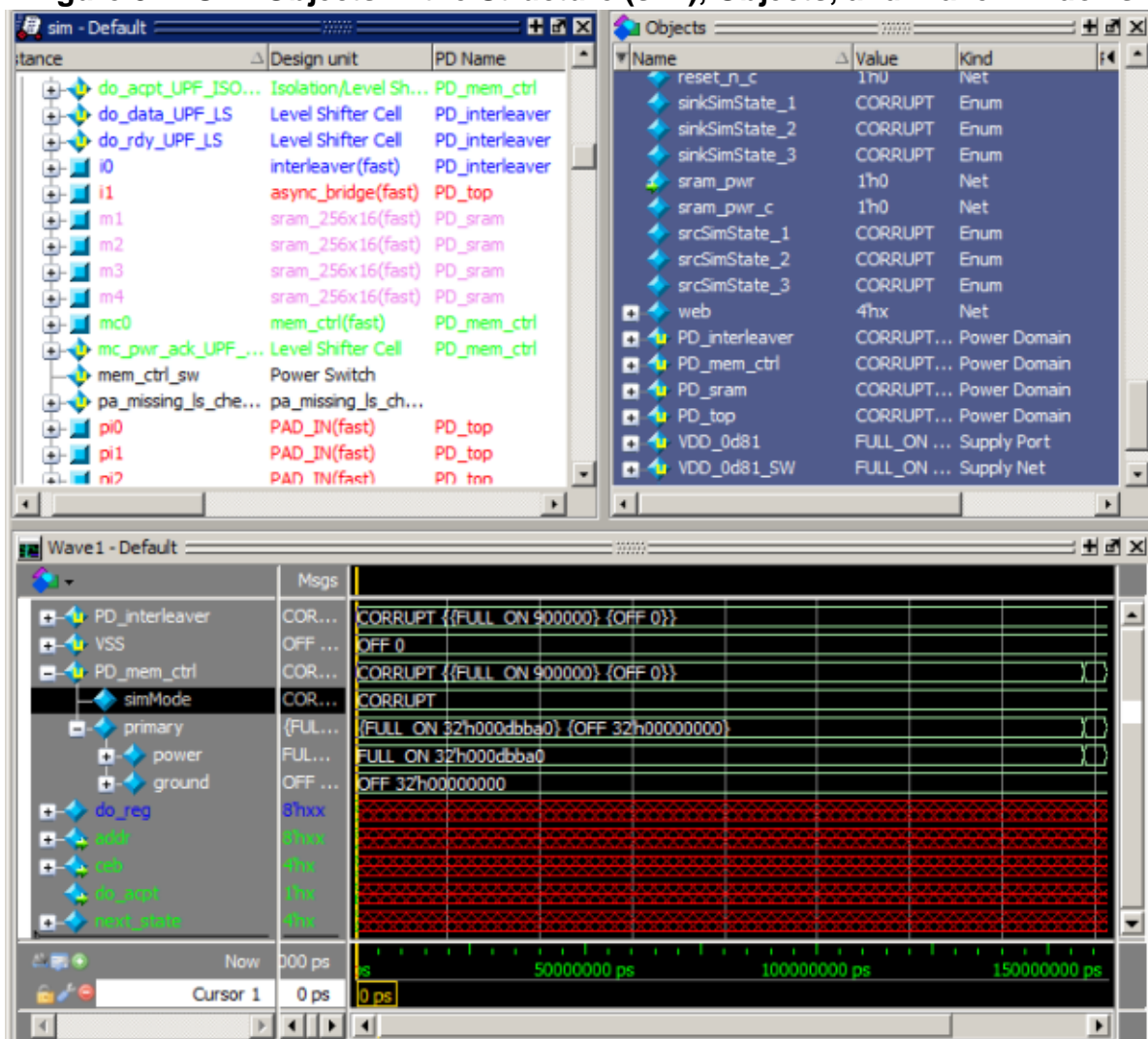
- Click the UPF object to view additional port and net information in the Objects window.
- Right-click the UPF object and choose **Add Wave** to add all of the related ports and nets in the Wave window.

The default value of supply ports and nets in these windows are:

- Voltage (Decimal)
- State (Enum, UNDETERMINED, FULL\_ON, OFF, PARTIAL\_ON).

## Results

**Figure 8-1. 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.

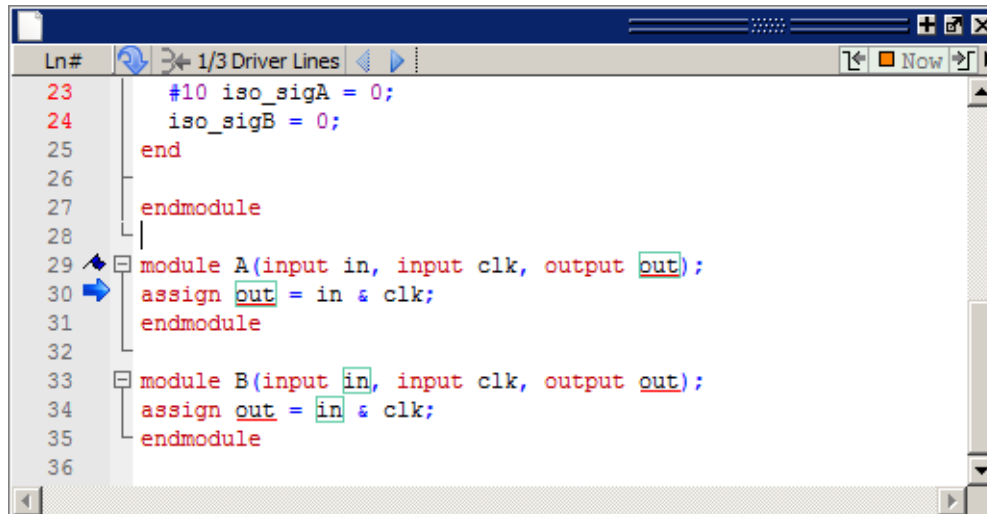
All of the features in this section are enabled when you add the `-pa_enable=debug` argument to your `vopt` command line.

|  |            |
|--|------------|
| <b>UPF Color Scheme in the Source Window .....</b>         | <b>172</b> |
| <b>UPF-Specific Context Menu in the Source Window.....</b> | <b>173</b> |
| <b>Show Drivers Control Bar .....</b>                      | <b>174</b> |

## 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.



### 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.

## 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.

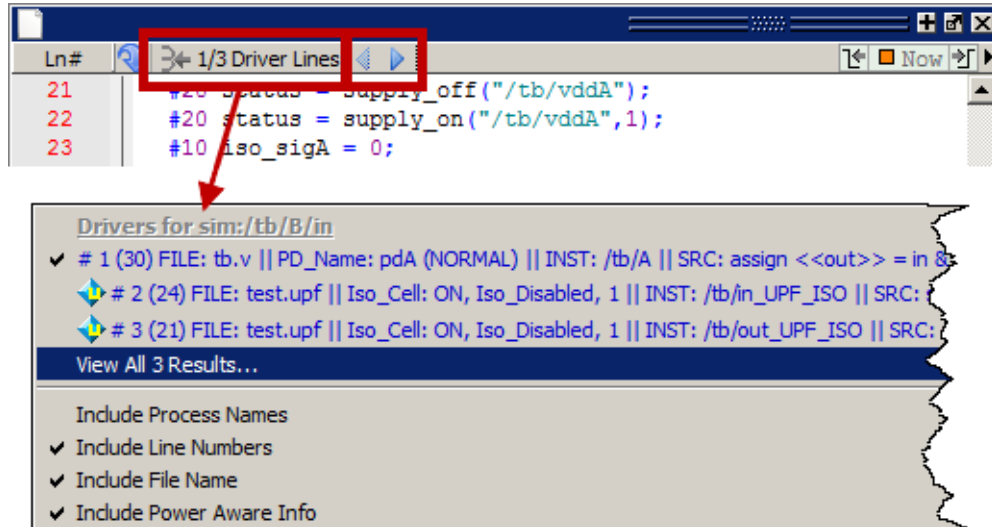
### Objects

| 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.<br><br>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 columned 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.

# 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>175</b> |
| <b>Bottom-Up Debugging (From the DUT) .....</b>       | <b>176</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 Two-Step 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 DUT)

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 Two-Step Flow](#)). For example:

```
vlog design.v
vcom design.vhdl
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.vhdl
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 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'.
```



# 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.

## Objects

| Object             | Description   |
|--------------------|---|
| Power Domain       | <p>All design elements are colored and highlighted according to their respective power domains (see “<a href="#">Figure 8-2</a>”)</p> <ul style="list-style-type: none"> <li>• All design elements, such as mux, flip-flops, and gates are colored according to the power domain specification.</li> <li>• The granularity of power domain display is at the instance level.</li> <li>• Any simulation-only power domains that are specified on a process or signal are not highlighted.</li> <li>• Colorization supports up to 16 different power domains.</li> <li>• If the power domain is in the OFF state, the schematic is filled with a gray color (see “<a href="#">Figure 8-3</a>”).</li> <li>• Isolation cells are filled in with a green color.</li> <li>• Level Shifter cells are filled in with a purple color.</li> </ul> |
| Excluded Domains   | <p>You can define a power domain to be excluded.</p> <ul style="list-style-type: none"> <li>• Power Aware exclude support (see “<a href="#">upf_dont_touch</a>”).</li> <li>• 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)</li> </ul> <p>Both are supported with instance-level granularity; signal- and process-level exclusions are not displayed.</p>  |
| UPF Source Viewing | <p>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:</p> <ul style="list-style-type: none"> <li>• Right-click and select <b>View Selection &gt; Power Domain &gt; Display UPF source code</b> (see “<a href="#">Figure 8-4</a>”).</li> <li>• 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 “<a href="#">Figure 8-5</a>”).</li> </ul>   |

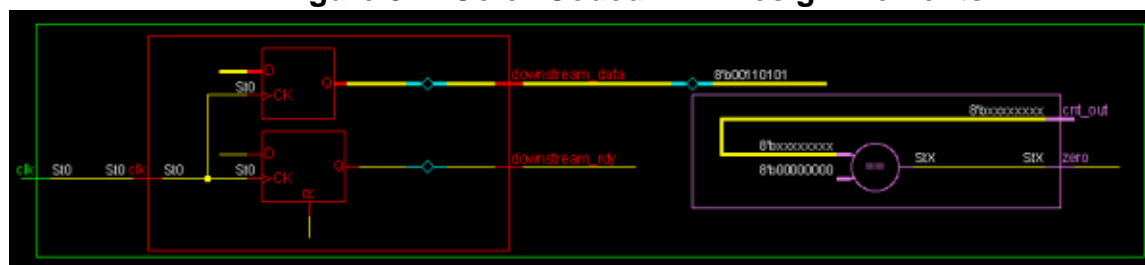
## Usage Notes

- 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
```

- Schematic Displays for Power Aware Debugging

**Figure 8-2. Color-Coded HDL Design Elements**



**Figure 8-3. Gray Area Indicates a Power Domain That is Off**

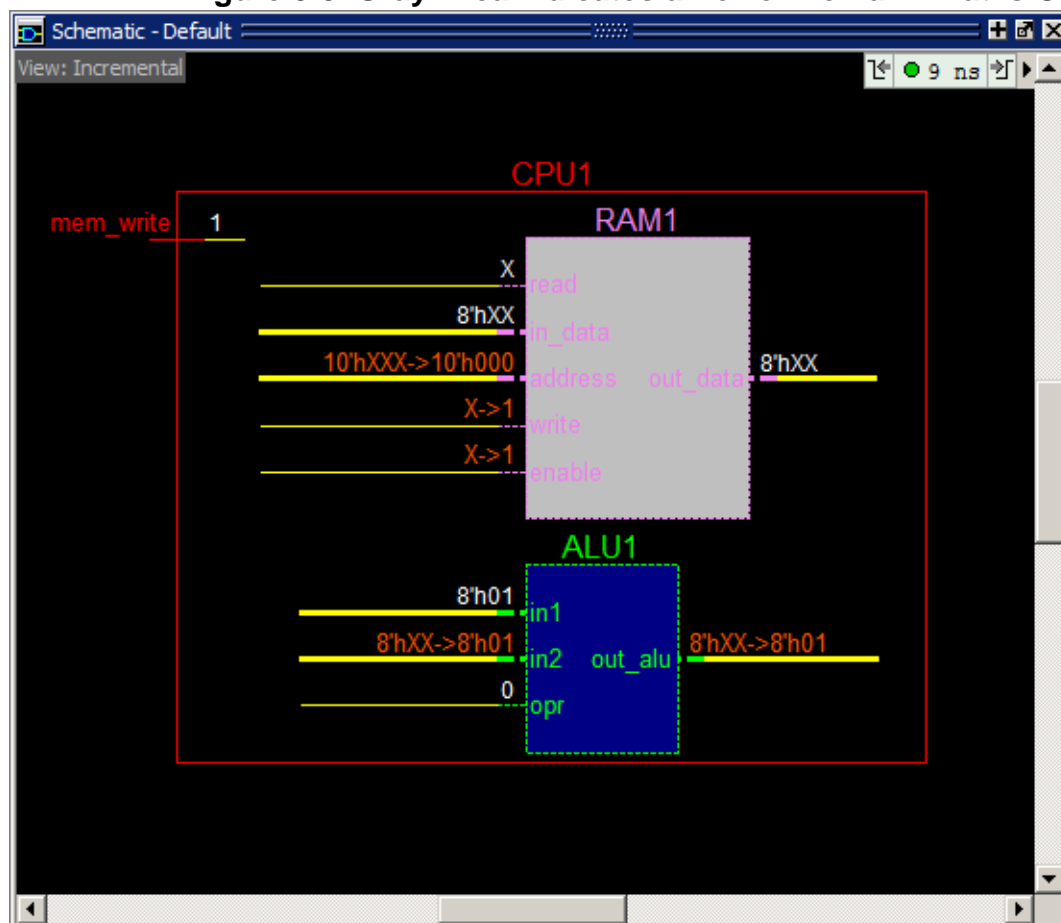


Figure 8-4. UPF Source File: Right-Click and Choose Power Domain

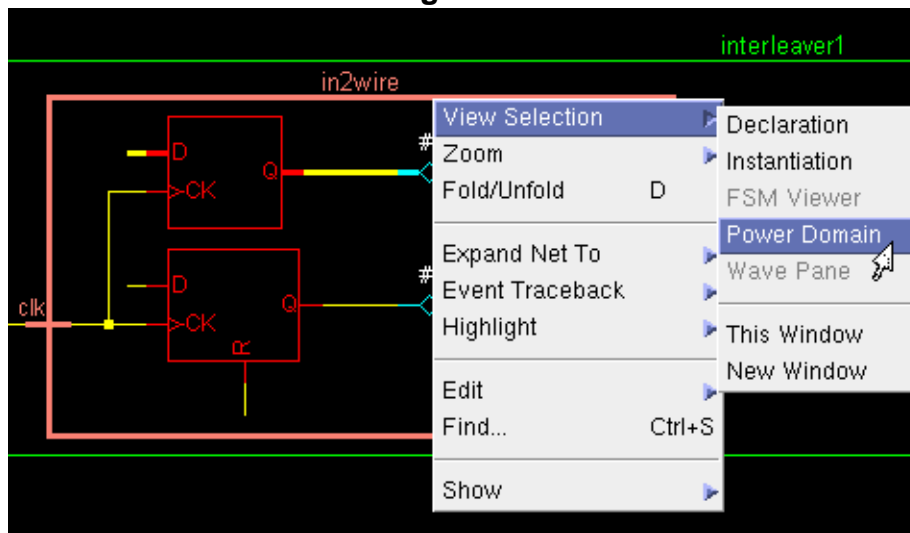
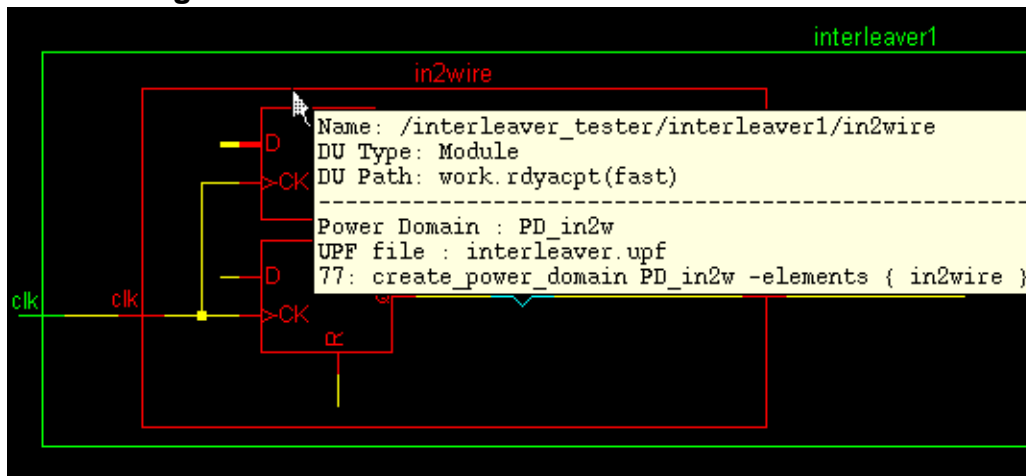


Figure 8-5. UPF Source File: Hover the Mouse and View Tool Tip



## Power Aware Waveform Display

You can use the Wave window to view bias mode, corruption, and isolation information for your design.

|  |            |
|--|------------|
| <b>Power Aware Waveform Concepts .....</b> | <b>180</b> |
| <b>Using Power Aware Highlighting.....</b> | <b>181</b> |

## 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 8-6 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

---

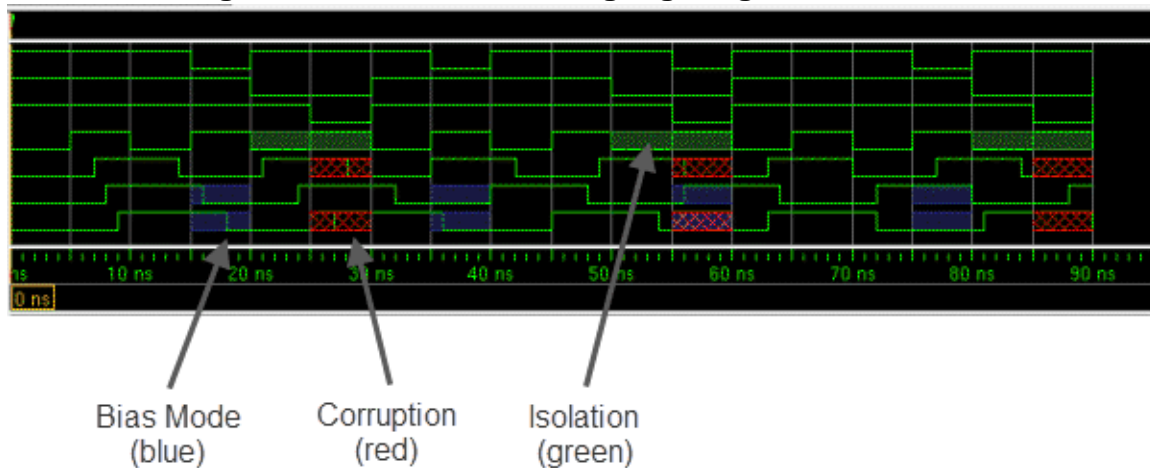
**Tip**

 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 8-6. Power Aware Highlighting in the Wave Window**



## Using Power Aware Highlighting

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:

```
vopt -pa_enable=highlight [other vopt arguments]
```

2. Enable highlighting during simulation:

```
vsim -pa_highlight [other vsim arguments]
```

This argument enables the generation of the WLF data used by the Wave window to display the highlighting.

3. Enable highlighting in the GUI:

**Wave > Wave Preferences > [Display tab] > PA waveform highlighting**

You only need to perform this action on your first invocation, or if your GUI settings are reset.


4. View Power Aware activity in post-simulation debug by loading the WLF file:

```
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>182</b> |
| <b>Displaying Power Aware State Machines .....</b>   | <b>183</b> |
| <b>Power Aware State Machine List Window .....</b>   | <b>185</b> |
| <b>Power Aware State Machine Viewer Window .....</b> | <b>187</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/nets 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 and -pa\_enable=fsmbasedcov argument.
- vsim command includes -coverage and -fsmdebug argument.

---

#### Note

 To view FSMs in the GUI, you can either include the -pa\_enable=fsmbasedcov argument in the vopt command, or the -fsmdebug argument in the vsim command.

---

### 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 Viewer Window](#).

3. View bubble diagrams of individual state machines in the [Power Aware State Machine Viewer Window](#).



## Power Aware State Machine List Window

To access:

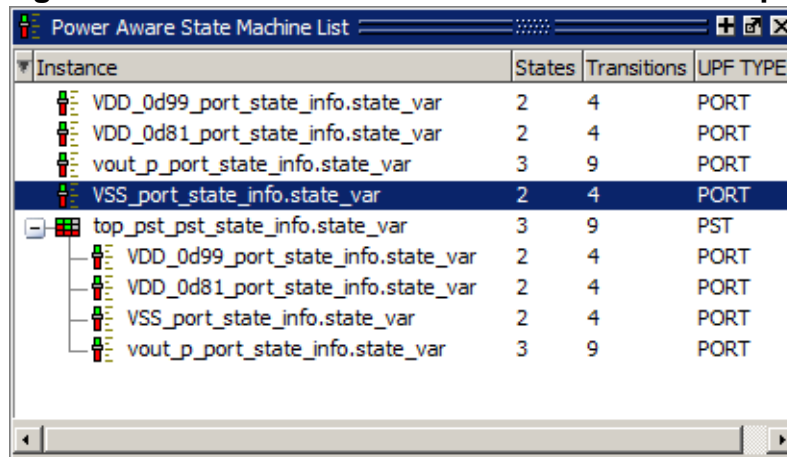
- View > PA State Machine List
- Command: view powerstatelist

Use the window to view a list of Power Aware state machines in the design with the hierarchical path of their creation after beginning a Power Aware simulation.

### Description

For example, if a PST has been created in *dut/top*, then this PST is visible in scope *dut/top/pa\_coverageinfo*.

**Figure 8-7. Power Aware State Machine List Example**



| 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     |

### Objects

**Table 8-1. Power Aware 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.   |
| 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> |

**Table 8-2. Power Aware State Machine List Window Columns**

| Column      | Description   |
|-------------|---|
| Instance    | Instance name of the state machine.<br><br>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:<br>PORT, NET, PST, POWER DOMAIN, SUPPLY SET   |

**Table 8-3. Power Aware State Machine List Window Popup Menu**

| Option             | 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.                    |

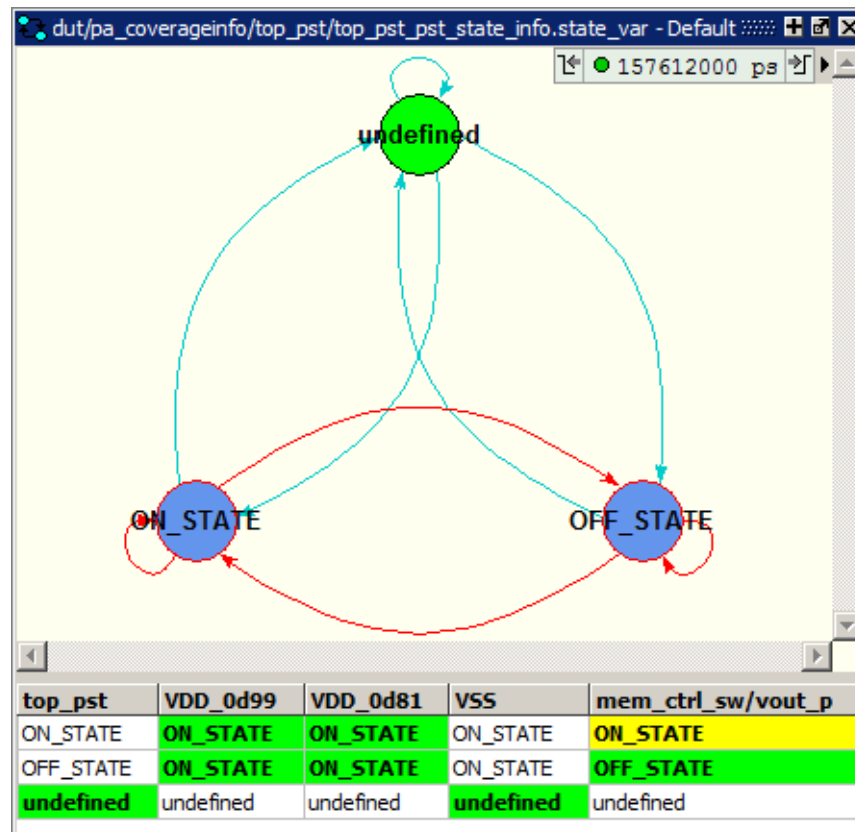
## Power Aware State Machine Viewer Window

To access: Double click the Power Aware state machine you want to analyze from the Power Aware State Machine List Window.

Use the window to view a state diagram of any of your Power Aware state machines.

### Description

**Figure 8-8. Power Aware State Machine Viewer Window Example**



### Objects

**Table 8-4. Power Aware State Machine Viewer Window GUI Elements**

| Object    | Description                         |
|-----------|-------------------------------------|
| Title bar | The UPF type and hierarchical path. |
| Tab label | Leaf name of the state machine.     |

**Table 8-4. Power Aware State Machine Viewer Window GUI Elements (cont.)**

| Object        | Description   |
|---------------|---|
| State Diagram | <p>All states of a state machine and their transitions.</p> <ul style="list-style-type: none"> <li>• Bubbles — Information associated with logic expression supply expression simstate legality. Information appears in properties box associated with each state bubble.</li> <li>• 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).</li> <li>• Label — State name</li> <li>• Transitions — Directional arrows showing transitions between the states.</li> <li>• Time Mode (upper right-hand corner) — The time of the current state transition, which decides yellow and green. It also affects States table.</li> <li>• Goto Previous and Next State buttons — Enables synchronization between other state machine windows and the wave window and the objects window.</li> <li>• Synchronization — By default, this window is synchronized with the Wave window, Objects window, and other Power Aware State Machine View windows.</li> </ul> <p>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.</p> <p>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.</p> |

**Table 8-4. Power Aware State Machine Viewer Window GUI Elements (cont.)**

| Object       | Description   |
|--------------|---|
| States table | <p>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.</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• Rows — Shows values for the time selected in the Time Mode widget, where green indicates the current value and yellow indicates the previous value.</li> </ul> |

**Table 8-5. 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.  |
| 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                  | <p>Displays the Power Aware State Machine Properties dialog box, which contains the following information about the selection.</p> <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> |

**Table 8-6. FSM View Menu, Specific to Power Aware State Machines**

|                            |   |
|----------------------------|---|
| 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.<br>The condition format is based on the GUI_expression_format <a href="#">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.  |
| 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>   |

# Chapter 9

## Power Aware Mixed RTL and Gate-Level Simulation

---

Power Aware simulation applies the power intent on a gate-level, or a mixed RTL and gate-level design automatically.

Power Aware simulation follows the listed semantics for applying the power intent:

1. [Hard Macro Model](#)

- Detects hard macros in the design
- Applies connection semantics
- Applies corruption semantics

2. [Power Management Cell](#)

- Detects power management cells in the design and matches them with the corresponding UPF strategies

---

**Tip**

Run [Static GLS Checks](#) to validate your gate-level design with the power intent defined in the UPF file.

---

- Inserts power management cells when a power management cell is defined in the UPF strategy, but the cell is missing in the design
- Applies connection semantics
- Applies corruption semantics

3. [Gate-Level Cell](#)

- Detects gate-level cells in the design
- Applies corruption semantics

4. [Sequential UDP](#)

- Detects sequential UDPs
- Applies corruption and retention semantics based on the corresponding UPF strategies

Also, Power Aware simulation uses the information present in the Liberty model to create a proper connection of the supplies and control signals that are defined in the UPF file. See “[Liberty Model](#)”.

The installation directory provides an example code on mixed RTL and gate-level design at the following location:

`<install_dir>/examples/pa_sim/PA_mix_RTL_GLS/`

|  |            |
|--|------------|
| <b>Hard Macro Model .....</b>                        | <b>193</b> |
| Hard Macro Model Overview .....                      | 194        |
| Liberty Model of a Hard Macro .....                  | 197        |
| Specifying Power Intent of a Hard Macro .....        | 200        |
| <b>Power Management Cell .....</b>                   | <b>203</b> |
| <b>Gate-Level Cell .....</b>                         | <b>214</b> |
| <b>Sequential UDP .....</b>                          | <b>215</b> |
| <b>Liberty Model .....</b>                           | <b>217</b> |
| Liberty Model Overview .....                         | 217        |
| Specifying Liberty to a Power Aware Simulation ..... | 218        |
| Liberty Attribute Library Management .....           | 219        |



# Hard Macro Model

---

A macro model 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 macro model 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. You can process hard 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 following location:

```
<install_dir>/examples/pa_sim/PA_liberty_hard_models/
```

**Hard Macro Model Overview..... 194**

**Liberty Model of a Hard Macro..... 197**

**Specifying Power Intent of a Hard Macro..... 200**

## Hard Macro Model Overview

The tool detects certain instances in the design as hard macro instances, and applies connection and corruption semantics.

### Hard Macro Cell Detection

- The tool detects an HDL simulation model as a hard macro model if the simulation model has the following attribute associated with it by a UPF set\_design\_attributes specification:

```
set_design_attributes -models ISOLSCCELL -attribute  
UPF_is_macro_cell TRUE
```

- The tool detects an HDL simulation model as a hard macro model if the simulation model has the following attribute associated with it by an HDL attribute specification:

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

- The tool detects a Liberty model as a hard macro model if the Liberty model contains the following attribute:

```
cell (mod_vl) {  
    is_macro_cell : true;  
    ...
```

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

```
vopt -pa_enable=libertycellcrpt
```

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.

### Connection Semantics

- The primary power pin of the cell is connected to the primary supply of the power domain.
- If the cell has bias pins, then the bias pin of the cell is connected to the primary supply of the power domain.

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

```
vopt -pa_enable=powerunconnnets
```

To connect the unconnected pg pins of the cells present inside a hard macro cell to the always-on power supply created by the tool, use the following argument:

```
vopt -pa_enable=conninsidemacro
```

## Corruption Semantics

- **Non-Power Aware HDL Model** — If there is a corresponding Liberty cell for the model, then the tool applies Liberty-based corruption semantics:
  - Input ports are corrupted using their related supply.
  - Output ports are corrupted using the power-down function.

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

```
vopt -pa_enable=libertypamodelopt
```

With this argument, the tool applies the following Liberty-based corruption semantics:

- For combinatorial cells, only the outputs are corrupted.
- For sequential cells, both outputs and inputs are corrupted.
- **Power Aware HDL Model** — The tool disables the simulation semantics.

|   |            |
|---|------------|
| <b>Non-Power Aware HDL Model .....</b>      | <b>195</b> |
| <b>Power Aware HDL Model.....</b>           | <b>196</b> |
| <b>Extended Power Aware HDL Model .....</b> | <b>197</b> |

## Non-Power Aware HDL Model

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 Model

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 because of those connections.

---

### Tip

 To apply Liberty-based corruption to all macro cells of a Power Aware HDL model, use the following attribute. See “[Supported Attributes](#)”.

```
set_design_attributes -attribute {qpa_override_pbp_corruption TRUE}
```

---

## Extended Power Aware HDL Model

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. By default, the simulator supports UPF to HDL net connection.

To use this mode of operation for a given hard macro, follow the same procedure as described in the “[Power Aware HDL Model](#)” section.

## Liberty Model of a Hard Macro

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 of 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 Model](#)”.

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_disable=libertyinpcorrupt
```

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 ENABLE 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 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 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.

The 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_enable=pgoutsynccheck
```

## 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 simulator issues an error message.

The 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 simulator defaults to using the HDL simulation model without any additional information from the Liberty model.

## Specifying Power Intent of a Hard Macro

A macro model 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 macro model 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.

Specify the power intent for a hard macro by any of the following:

- [set\\_port\\_attributes](#) UPF command
- [RTL Attributes](#)
- [Liberty Attributes](#)

### RTL Attributes

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 Attributes

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 9-1 shows an example of a block diagram of hard macro power domains, using the following top-level UPF definition:

```
set_scope top

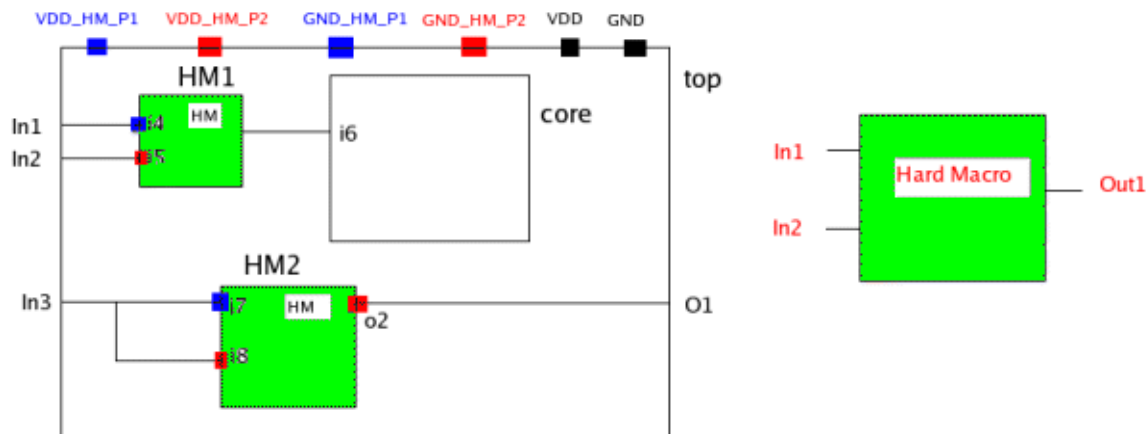
create_power_domain PD_top -include_scope

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 9-1. Design Consisting of Hard Macros**



The following sections show how to define the hard macro power intent for each method.

### set\_port\_attributes UPF Command

The constraints are specified on the pins:

```
set_port_attributes -ports { HM1/i4 } -related_power_port VDD_HM_P1
                  -related_ground_port 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 i4;  
  
(* UPF_related_power_pin = "VDD_HM_p2", UPF_related_ground_pin =  
  "GND_HM_p2" *) input i5;
```

For HM2

```
(* UPF_related_power_pin = "VDD_HM_p1", UPF_related_ground_pin =  
  "GND_HM_p1" *) input i7;  
  
(* UPF_related_power_pin = "VDD_HM_p2", UPF_related_ground_pin =  
  "GND_HM_p2" *) input i8;  
  
(* UPF_related_power_pin = "VDD_HM_p2", UPF_related_ground_pin =  
  "GND_HM_p2" *) output o2;
```

**Liberty 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;  
    }  
  }  
}
```

## Power Management Cell

Power Aware simulation identifies power management cells (isolation, level shifter, or retention) in the design, and matches them with a UPF strategy. It inserts power management cells when the cell is defined in the UPF strategy, but is missing in the design; it also applies connection and corruption semantics.

The topic has the following sections:

- [Identifying a Power Management Cell](#)
- [Matching a Power Management Cell](#)
- [Connection Semantics](#)
- [Corruption Semantics](#)

## Identifying a Power Management Cell

Power Aware simulation identifies a power management cell using the Liberty or HDL attributes.

**Table 9-1. Identifying Criteria**

| Identifying Criteria      | Example  |
|---------------------------|--|
| <b>Liberty attributes</b> |  |
| is_isolation_cell         | <pre>cell (ISO_LO) {<br/>  is_isolation_cell : true;<br/>  ...</pre>             |
| is_level_shifter          | <pre>cell (prim_ls_buf) {<br/>  is_level_shifter : true;<br/>  ...</pre>         |
| retention_cell            | <pre>cell (RETCELL) {<br/>  retention_cell : retdiff;<br/>  ...</pre>            |
| <b>HDL attributes</b>     |  |
| is_isolation_cell         | <pre>(* is_isolation_cell = 1 *)<br/>module ISOCELL ( I, ISO, Z) ;<br/>...</pre> |
| is_level_shifter          | <pre>(* is_level_shifter = 1 *)module LSCELL ( I,<br/>Z) ;<br/>...</pre>         |

## Matching a Power Management Cell

Power Aware simulation matches the power management cells to the corresponding UPF strategy in the UPF file using the below listed ways.

**Table 9-2. Matching Criteria**

| Matching Criteria  | Example  |
|--|--|
| <b>The -lib_cells option with the following commands</b> |  |
| map_isolation_cell                                       | <pre>map_isolation_cell ISO1 -domain A<br/>-lib_cells {ISO_CELL}<br/>...</pre> |

**Table 9-2. Matching Criteria (cont.)**

| Matching Criteria   | Example   |
|---|---|
| map_level_shifter_cell  | map_level_shifter_cell LS1 -domain A<br>-lib_cells {LS_LH}<br>...                                     |
| map_power_switch  | map_power_switch {sw} -domain PD<br>-lib_cells {SWITCH1}<br>...                                       |
| map_retention_cell  | map_retention_cell pd_ret -domain PD<br>-lib_cells {RET_CELL}<br>...                                  |
| use_interface_cell  | use_interface_cell myinterface -strategy<br>{ls_PD_mid2} -domain {PD_mid2} -lib_cells {LS_LH}<br>...  |
| <b>The name_format command with the following options</b>   |   |
| -level_shift_prefix   | name_format<br>-level_shift_prefix "MY_LS_"<br>...  |
| -level_shift_suffix   | name_format<br>-level_shift_suffix "_UPF_LS"<br>...   |
| -isolation_prefix   | name_format<br>-isolation_prefix "MY_ISO_"<br>...   |
| -isolation_suffix   | name_format<br>-isolation_suffix "_UPF_ISO"<br>...  |
| <b>Synopsys pragmas with isolation_cell_enable_pin, level_shifter_enable_pin, or retention_enable_pin set to TRUE</b> |   |
| isolation_upf   | ISOCELL ISO_LS_out2<br>(.ISO(en),.I(out2),.Z(iso_ls_out2));<br>//synopsys isolation_upf ISO3+A<br>... |
| level_shifter_upf   | LSCELL LS_out2<br>(.LS(en),.I(out2),.Z(ls_out2));<br>//synopsys level_shifter_upf LS3+A<br>...        |
| retention_upf   | RETCELL RET_1<br>( clk,in1[0],restore,out1[0],save );<br>//synopsys retention_upf RET1+PD_IN1<br>...  |
| <b>The -instance option with the following commands</b>   |   |
| set_isolation   | set_isolation -instance <instance_name>   |
| set_level_shifter   | set_level_shifter -instance <instance_name>   |

**Table 9-2. Matching Criteria (cont.)**

| Matching Criteria | Example                                 |
|-------------------|---|
| set_retention     | set_retention -instance <instance_name> |

## Connection Semantics

The tool connects the supply pins (such as primary power, backup power, and bias pins) of the power management cell according to the cell type.

## Retention Cell

The tool follows the listed connection semantics:

- If the tool identifies a retention cell that does not match with a UPF strategy, then the tool disables the simulation semantics of the retention cell.
- If the tool identifies a retention cell that matches with a UPF strategy, then it performs the following connections if explicit connections are missing:
  - Cell primary power pin is connected to the primary supply of the power domain.
  - Cell backup power pin is connected to the retention supply specified in the retention strategy.

If the retention strategy specifies the `set_retention -use_retention_as_primary` command, then the primary power and backup power is connected to the retention supply specified in the retention strategy.

## Always-On Cell

If the tool identifies an always-on cell, then it performs the following connections if explicit connections are missing:

- Cell primary power pin is connected to the primary supply of the power domain.
- Cell backup power pin is connected to the tool generated always-on supply source.

## Switch Cell

The tool follows the listed connection semantics:

- If the tool identifies a switch cell that does not match with a UPF strategy, then the tool disables the simulation semantics of the switch cell.
- If the tool identifies a switch cell that matches with a UPF strategy, then it performs automatic connections if explicit connections are missing. If explicit connections are missing, and the tool is not able to perform automatic connections, then the tool disables the simulation semantics.

## Isolation Cell

The tool follows the listed connection semantics:

- If the tool identifies an isolation cell that does not match with a UPF strategy, then the tool disables the simulation semantics of the isolation cell.
- If the tool identifies an isolation cell that matches with a UPF strategy, it performs the following connections depending on the location of the cell, if explicit connections are missing:
  - When the cell location is self:
    - Cell primary power pin is connected to the primary supply of the power domain.
    - Cell backup power pin is connected to the isolation supply of the isolation strategy.
    - If the isolation cell has bias pins, it performs the following connections:
      - If the bias pin is related to the primary supply, or to both primary and backup supply, or it is not related to either primary or backup supply in Liberty, then the bias pin is connected to the primary supply of the source power domain.
      - If the bias pin is related to the backup supply in Liberty, then the bias pin is connected to the isolation supply set of the isolation strategy.
  - When the cell location is parent:
    - Cell primary power pin is connected to the isolation supply set of isolation strategy.
    - Cell backup or secondary power is connected to the primary supply of the power domain.
    - If the isolation cell has bias pins, it performs the following connections:
      - If the bias pin is related to the primary supply, or to both primary and backup supply, or it is not related to either primary or backup supply in Liberty, then the bias pin is connected to the isolation supply set of the isolation strategy.
      - If the bias pin is related to the backup supply in Liberty, then the bias pin is connected to the primary supply of the source power domain.

### Special Isolation Cell

The tool identifies a special isolation cell when the input pin, output pin, and enable pin all have the backup power supply as their respective related\_supply in the Liberty file.

This special isolation cell has two power supply pins:

- **Primary power pin** — The pin with pg\_type attribute primary\_power or optionally additional presence of std\_cell\_main\_rail : true attribute on the pin.

- **Secondary or backup power pin** — The pin with pg\_type attribute backup power, or pg\_type primary\_power without the std\_cell\_main\_rail attribute (then the primary power pin shall have scmr :true)

The tool performs the following connections when the special isolation cell matches with an isolation strategy:

- The primary supply is connected to the primary supply of the domain in which the cell is present.
- The backup supply is connected to the isolation supply that you specify in the isolation strategy.
- The bias pins, which are related bias pins of the primary power, is connected to the appropriate function of the primary supply of the domain in which the cell is present.

## Level Shifter Cell

If the tool identifies a level shifter cell, then it performs the connections as specified in the following, and in the listed precedence:

1. The [connect\\_supply\\_net](#) command.
2. The [set\\_level\\_shifter](#) command with supply sets
3. If the level shifter cell matches with a level shifter strategy with no or incomplete supply set details, it performs the following connections:
  - The input supply pin is connected to the primary supply of the source power domain.
  - The output supply pin is connected to the primary supply of the sink power domain.
  - If the level shifter cell has bias pins, it performs the following connections.
    - If the bias pin is related to the input supply pin, then the bias pin is connected to the primary supply of the source power domain.
    - If the bias pin is related to the output supply pin, then the bias pin is connected to the primary supply of the sink power domain.
4. The vopt -pa\_enable=autoconns argument.
5. If a level shifter cell does not match with any UPF strategy, then the tool does not connect the power supply of the level shifter cell.

## Enable Level Shifter (ELS) Cell

Enable level shifter cells are integrated level shifter and isolation cells.



The tool identifies 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

Once the tool identifies an ELS cell, which may be a 3 or 2 PG pin ELS cell or a 2 PG pin combo cell, it identifies the PG pins for connections:

- **3 PG Pin ELS Cell (With 1 Ground Pin)**

- Input supply pin — As the related pin of isolation or level shifter cell data pin.
- Output supply pin — As the related pin of the logic output pin of the cell.
- Internal supply pin — By the presence of `std_cell_main_rail` : true attribute. If the `std_cell_main_rail` attribute is missing, the tool flags a suppressible error.

- **3 PG Pin ELS Cell (With 2 Ground Pins)**

The Liberty file of a 3 PG pin ELS cell has 2 ground ports:

- The first ground port has the `primary_ground` attribute.
- The second ground port has the `backup_ground` attribute.

Also, all data pins have `backup_ground` specified in their respective related supplies attributes.

The tool identifies the input, output, and internal supply pins similar to the 3 PG pin ELS cell with 1 ground pin. Also, the tool identifies the ELS cell depending on the type of biasing:

- Source side biased ELS cell — The input supply pin and the internal supply pin have the same `related_bias_pin`.
- Sink side biased ELS cell — The output supply pin and the internal supply pin have the same `related_bias_pin`.

- **2 PG Pin ELS Cell**

- Input supply pin — As the related pin of isolation or level shifter cell data pin.
- Output supply pin — As the related pin of the logic output pin and the enable pin of the cell.

- **2 PG Pin Combo Cell**

- Input supply pin — As the related pin of isolation or level shifter cell data pin and the enable pin of the cell.
- Output supply pin — As the related pin of the logic output pin of the cell.

### Connection Semantics

The tool performs the connections as specified in the following, and in the listed precedence:


1. The [connect\\_supply\\_net](#) command.
2. The [set\\_level\\_shifter](#) command with supply sets.
3. If the ELS cell matches with a level shifter strategy with no or incomplete supply set details, it performs the following connections:
  - **3 PG Pin ELS Cell (With 1 Ground Pin)**
    - The input supply pin is connected to the primary supply of the source power domain.
    - The output supply pin is connected to the primary supply of the sink power domain. In case the sink power supply is not equivalent to the isolation supply that you specify in the isolation strategy, the tool connects the output supply pin to the isolation supply.
    - The internal supply pin is connected to the primary supply of the domain in which the cell is present.
    - If the ELS cell has bias pins, it performs the following connections:
      - If bias pin is related to the internal PG pin, or shared between the internal PG pin and source side PG pin, or shared between the internal PG pin and sink side PG pin, then the bias pin is connected to the appropriate function of the primary supply of the domain in which the cell is present.
      - If the bias pin is related to the input PG pin only, then the bias pin is connected to the appropriate function of the primary supply of the source power domain.
      - If the bias pin is related to the output PG pin only, then the bias pin is connected to the appropriate function of the primary supply of the sink power domain.
  - **3 PG Pin ELS Cell (With 2 Ground Pins)**

The tool performs the input, output, internal, and bias pins connections similar to the 3 PG pin ELS cell with 1 ground pin.

For the backup ground, the tool performs the following connections:

- Source side ELS cell — The ground pin with the `backup_ground` attribute is connected to ground of the source supply.
- Sink side ELS cell — The ground pin with the `backup_ground` attribute is connected to ground of the sink supply.
- **2 PG Pin ELS Cell**


### **Note**

 The supply pin with the `std_cell_main_rail : true` attribute is connected to the primary supply of the domain in which the cell is present.

---

- Identifies whether it is a source or sink side ELS cell:
  - Sink side ELS cell — By the presence of the `std_cell_main_rail : true` attribute on the output supply.
  - Source side ELS cell — By the presence of the `std_cell_main_rail : true` attribute on the input supply.
- Performs the following connections:
  - Sink side ELS cell:
    - The output supply pin (with the `std_cell_main_rail` attribute : `true` attribute) is connected to the primary supply of the power domain in which the cell is present.
    - The input supply pin is connected to the primary supply of the source power domain.
  - Source side ELS cell:
    - The input supply pin (with the `std_cell_main_rail` attribute : `true`) is connected to the primary supply of the power domain in which the cell is present.
    - The output supply pin is connected to primary supply of the sink power domain. In case the sink power supply is not equivalent to the isolation supply that you specify in the isolation strategy, the tool connects the output supply pin to the isolation supply.
- **2 PG Pin Combo Cell**

### **Note**

 The supply pin with the `std_cell_main_rail : true` attribute is connected to the primary supply of the domain in which the cell is present

---

- Identifies whether it is a source or sink side combo cell:
  - Sink side combo cell — By the presence of the `std_cell_main_rail : true` attribute on the output supply.
  - Source side combo cell — By the presence of the `std_cell_main_rail : true` attribute on the input supply.
- Performs the following connections:
  - Sink side combo cell

- The output supply pin (with the `std_cell_main_rail` attribute : true) is connected to the primary supply of the power domain in which the cell is present.
  - The input supply pin is connected to the isolation supply of the matched isolation strategy.
    - Source side combo cell — The tool displays an error.
4. The `vopt -pa_enable=autoconnls` argument.
  5. If the ELS cell matches with an isolation strategy, the tool follows the isolation cell connection rules. See “[Isolation Cell](#)”.
  6. If the ELS cell does not match with any UPF strategy, then the tool does not connect the power supply of the ELS cell.

## Corruption Semantics

If a Liberty model is present for a power management cell, the tool applies Liberty-based corruption semantics, else it applies driver-based corruption semantics based on the supply specified in the UPF strategy.

### Always-On Cell

Always-on cells are corrupted only if the backup power goes off.

To disable Liberty-based corruption semantics for always-on cells, use the following argument:

```
vopt -pa_disable=aoncellcrpt
```

### Switch Cell

To disable Liberty-based corruption semantics for switch cells, use the following argument:

```
vopt -pa_disable=swcellcrpt
```

## 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 as following:

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

Default corruption values for VHDL are as following:

- 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.

### Disabling Automatic Insertion

Use the following vopt arguments to selectively disable automatic insertion of power management cells:

- `-pa_disable=insertiso` — Disables the insertion of isolation cells
- `-pa_disable=insertls` — Disables the insertion of level\_shifter cells
- `-pa_disable=insertret` — Disables the insertion of retention cells

In pure gate-level designs, isolation, level shifting, and retention cells are already present, and therefore you do not need to insert these cells. Use the following vopt argument to disable automatic insertion of isolation, level shifter, and retention cells:

- `-pa_gls` — Disables the insertion of all power management cells

## Reports

The tool reports the cells detected as an instance of a UPF strategy in the Architecture Report, *report.pa.txt*:

```
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 or level shifter cell, Power Aware simulation semantics are disabled and the cell is treated as always-on cell. The tool displays the following message:

```
** 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, isolation or level shifter cell, and displays the following message:

```
** 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
```

## Gate-Level Cell

Power Aware simulation detects gate-level cells, and applies corruption semantics.

### Gate-Level Cell Detection

- The tool detects a module as a gate-level cell if the module contains the ``celldefine` attribute or the module contains the *specify* block.
- You can treat any HDL cell (where the module has no ``celldefine` attribute or a *specify* block, or it is a VHDL cell) as a gate-level cell (a candidate for output only corruption) by setting the UPF attribute `UPF_is_leaf_cell` as 'true' on the cell. You can apply the attribute on both Verilog and VHDL cells.

- You can treat any gate-level cell as a HDL cell, specifically a candidate for driver-based corruption, by setting the UPF attribute `UPF_is_leaf_cell` as 'false' on the cell. You can apply the attribute on both Verilog and VHDL cells.

## Corruption Semantics

For detected gate-level cells, the tool applies corruption on the output ports and sequential logic of the cell. For RTL cells, the tool applies the standard processing of driver-based corruption.

# Sequential UDP


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

## UDP Corruption and Retention Modes

Use the following modes to mimic the corruption and retention behavior in sequential UDPs.

- 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 when the following conditions are met:
  - Save and restore are active and `save_condition` and `restore_condition` are true.
  - 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 is 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 the `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. To avoid application of retention semantics to UDPs that already include retention logic, exclude the UDP using the following command:

```
set_retention -instance <inst_name>
```

- 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.



## Liberty Model

Power Aware simulation uses the information present in the Liberty model to create a proper connection of the supplies and control signals that are defined in the UPF file. Also, Power Aware simulation uses the Liberty model to assist in automatic recognition of power management cells that are present in the design.

|  |            |
|--|------------|
| <b>Liberty Model Overview .....</b>                        | <b>217</b> |
| <b>Specifying Liberty to a Power Aware Simulation.....</b> | <b>218</b> |
| <b>Liberty Attribute Library Management .....</b>          | <b>219</b> |

## Liberty Model Overview

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

The information contained in a Liberty model may be of use in RTL, gate-level, and mixed RTL and gate-level 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 the Liberty model may be of use in applying power intent to the gate-level portion of the design.


Liberty model details are present in either of the following:

- Liberty Source File — File that contains one or more cells and typically has the file suffix *.lib*.
- Liberty Attribute Library — File output by the vopt command using the `-pa_dump libertydb` argument that combines the information from one or more Liberty source files. Liberty attribute library are useful when the same library is used many times.

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 contains versions of each cell. If your design references the cell name with multiple instances, vopt picks one at random and issues a warning message about multiple cells of the same name.

---

### Note

 To learn more about Liberty, read the Liberty User Guides and Reference Manual Suite at the following location. You need to select **Liberty**, and then register and accept the open source license agreement to access the suite.

<http://www.opensourceliberty.org>

---

## Specifying Liberty to a Power Aware Simulation

Specify the Liberty source files or the Liberty attribute library during the processing of the UPF file.

### Procedure

1. To specify the Liberty source files, use the following command:

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

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

2. To specify the Liberty attribute library, use the following command:

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

## Liberty Attribute Library Management

Use the vopt command to create, update, and refresh Liberty attribute libraries. The Liberty attribute libraries enable you to specify Liberty cells during the processing of the UPF file.

|   |            |
|---|------------|
| <b>Creating a Liberty Attribute Library .....</b>   | <b>219</b> |
| <b>Updating a Liberty Attribute Library .....</b>   | <b>219</b> |
| <b>Refreshing a Liberty Attribute Library .....</b> | <b>220</b> |

## Creating a Liberty Attribute Library

Use the vopt command to parse the Liberty source files and create an attribute library containing information extracted from the Liberty source files.

### Procedure

1. Create a Liberty attribute library:

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

- -pa\_libertyfiles — Specifies the Liberty source files to read. You can specify multiple files by separating file names with a comma.
- -pa\_dumplibertydb — Specifies the Liberty attribute library for future use.

2. Alternatively, you can add multiple file names to an external file and use the -f argument with the vopt command. This can be useful if you need to specify many Liberty source 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 the following:

```
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. You update an existing library when a Liberty source file is edited to correct an error and needs to be reprocessed.

### Procedure

1. To update an already existing Liberty source file, for example, *a.lib*, and update an already existing Liberty attribute library, for example, *lib\_datafile*, use the following command:

```
vopt -pa_libertyfiles=a.lib -pa_dumplibertydb=libdb/lib_datafile  
-pa_libertyenable=update
```

The content of the Liberty source file, *a.lib*, overwrites any cells created during the addition of the file during the initial creation or the previous update.

2. To add a new Liberty source file, for example, *c.lib*, and update an already existing Liberty attribute library, for example, *lib\_datafile*, use the following command:

```
vopt -pa_libertyfiles=c.lib -pa_dumplibertydb=/libdbs/lib_datafile  
-pa_libertyenable=update
```

The new cells from *c.lib* are added to those added during the initial creation or a previous update.

## Refreshing a Liberty Attribute Library

If you created a Liberty attribute library with a previous version of the tool, you can refresh the attribute library to work with the recent version.

### Procedure

Refresh the Liberty attribute library *lib\_db* to the current version of the simulator:

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

# Chapter 10

## Power Aware Simulation Debug

---

Debug your Power Aware Simulation with Power Aware apps, Questa TKGUI, or Visualizer Debug Environment.

|   |            |
|---|------------|
| <b>Debugging With Questa TKGUI .....</b>                    | <b>222</b> |
| Creating and Loading a Debug Database in Questa TKGUI ..... | 222        |
| Power Aware Debug Support in Questa TKGUI .....             | 222        |
| <b>Debugging With Visualizer Debug Environment .....</b>    | <b>225</b> |
| Creating and Loading a Debug Database in Visualizer .....   | 225        |
| Power Aware Debug Support in Visualizer .....               | 227        |
| <b>Debugging With Power Aware Apps .....</b>                | <b>230</b> |
| Power Aware Debugging Apps Overview .....                   | 230        |
| Running the Power Aware Apps .....                          | 230        |
| Finding the Source of a Corrupt Signal .....                | 231        |
| Reporting Selected Power Aware Information .....            | 233        |

## Debugging With Questa TKGUI

Power Aware simulation enables you to create a database of information for analyzing the results in the Questa SIM TKGUI after completion of a live simulation.

**Creating and Loading a Debug Database in Questa TKGUI . . . . . 222**

**Power Aware Debug Support in Questa TKGUI . . . . . 222**

## Creating and Loading a Debug Database in Questa TKGUI

The debug database helps you to debug a Power Aware simulation that is run in a regression setup or where you are not able to debug a live simulation.

### Procedure

1. Compile your design files:

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

2. Optimize your top-level design and enable creation of the post-simulation debug database:

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

---

#### Note



In a two-step flow, use the -pa\_enable=debug argument in the -voptargs argument to the vsim command.

---

3. Simulate the Power Aware version of your design and create the post-simulation debug database:


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


4. Load the WLF file and debug database to perform debug activities on the results:

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

## Power Aware Debug Support in Questa TKGUI

The tool provides debugging features that are available during a live simulation and post simulation mode.

| Debug Feature  | Live-Sim Mode | Post-Sim Mode   | Topic  |
|--|---------------|---|--|
| Highlights signal in the Wave window: <ul style="list-style-type: none"> <li>• Coloring/hashing</li> <li>• Popup information</li> </ul>  | Supported     | Supported   | <a href="#">Power Aware Waveform Display</a>                   |
| Displays UPF object in the Structure, Object, and Wave windows.  | Supported     | Supported   | <a href="#">UPF Object Display</a>                             |
| Displays Power Aware information in the Structure, Object, and Wave windows.   | Supported     | Supported   | <a href="#">Power Aware Simulation Display in Questa TKGUI</a> |
| Displays isolation and level-shifter cells in the Dataflow and Schematic windows.  | Supported     | Supported   | <a href="#">Power Aware Schematic Display</a>                  |
| Displays information on Power Aware static check debug in the Results Analysis window.<br> <b>Note:</b> Not available for ModelSim® SE™ (Questa SIM only). | Supported     | Supported<br>Works without running the simulation, but does require access to the Results Analysis window   | <a href="#">Power Aware Static Check Display</a>               |
| Displays 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> | <a href="#">Power Aware Checks</a>                             |

| Debug Feature   | Live-Sim Mode | Post-Sim Mode                 | Topic  |
|---|---------------|-------------------------------|--|
| Displays Power Aware test plan.<br> <b>Restriction:</b> Not Available for ModelSim® SE™ (Questa SIM only). | Supported     | Supported<br>Use viewcov mode | <a href="#">Analyzing Coverage Using the Power Aware Test Plan</a> |
| Displays information on power, port and PST state.  | Supported     | Not Supported                 | <a href="#">Power State and Transition Display</a>                 |
| Displays information on Power Aware source code debug.  | Supported     | Supported                     | <a href="#">Power Aware Source Window</a>                          |
| Displays information on Power Aware coverage.   | Supported     | Supported<br>Use viewcov mode | <a href="#">Power Aware Coverage</a>                               |



# Debugging With Visualizer Debug Environment

Power Aware simulation enables you to create a database of information for analyzing the results in the Visualizer Debug Environment after completion of a live simulation.

|  |            |
|--|------------|
| <b>Creating and Loading a Debug Database in Visualizer .....</b> | <b>225</b> |
| <b>Power Aware Debug Support in Visualizer .....</b>             | <b>227</b> |

## Creating and Loading a Debug Database in Visualizer

The debug database helps you in post-simulation debug. Create the appropriate design and waveform database files that are required by Visualizer for post-simulation debug.

### Procedure

1. Create your work library:

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

2. Compile your design files:

| If you want to...                        | Use the following:  |
|--|---------------------|
| Compile Verilog or SystemVerilog designs | vlog <design_files> |
| Compile VHDL designs                     | vcom <design_files> |

3. Optimize your top-level design:

```
vopt <design_top> -pa_upf <upf_file> -designfile <designfile> -debug
-o <optimized_output> [-pa_checks=s+d]
```

The vopt command creates the following database files:

- *design.bin* — Design database file containing the HDL design information.
- *design.bin.padb* — Power Aware design database file containing the Power Aware (UPF) design information.

4. (Optional) To view the details of the static and dynamic checks in the PA windows, add the following arguments to the vopt command:

```
vopt <design_top> [-pa_checks=s+d]
```

- -pa\_checks=s — Creates the static check debug data in the design database files. View the information related to the static checks in the PA Crossings window.

- `-pa_checks=d` — Creates the dynamic check debug data in the design database files. View the information related to the dynamic checks in the PA SimChecks window.
5. Simulate the Power Aware version of your design, and create the Visualizer waveform database file:

```
vsim -c -pa <optimized_output> -qwavedb=<options>
```

6. Add the following options to the `-qwavedb` argument for post-simulation debug:

```
-qwavedb=+signal+msgmode+displaymsgmode
```

The `-qwavedb` argument creates the Visualizer waveform database file, *qwave.db*, in the current working directory.

7. Load your design in Visualizer with the design database file, waveform database file, and Power Aware options:




```
visualizer -designfile <designfile> -wavefile <wavefile>  
[+pa_db+<padesignfile>]
```

Visualizer tries to locate the Power Aware design database file in the directory where you specify the design database file, *<designfile>*. If the Power Aware design database file is not present in the same directory, then provide the path and filename in the *<padesignfile>*.

## Power Aware Debug Support in Visualizer

Visualizer has various debug features, and most of the features are available by default.

### Objects

| Window              | Debug Feature   | vopt option          |
|---------------------|---|----------------------|
| Connectivity Tracer | <ul style="list-style-type: none"> <li>Highlights all Power Aware objects in the window with each power domain in a different color.</li> <li>Displays the Power Aware information when you hover over the Power Aware object.</li> </ul>   | Available by default |
| Design              | <ul style="list-style-type: none"> <li>Specifies the power domain of a design object in brackets.</li> <li>Displays the Power Aware objects such as isolation, level shifter, and repeater cells.</li> <li>Displays the following information when you hover over the Power Aware object: <ul style="list-style-type: none"> <li>Power domain</li> <li>Current simstate</li> <li>UPF file where you define the power domain</li> <li>Simstate behavior, if you specify it using the <code>set_simstate_behavior</code> command</li> <li>UPF or Liberty attribute notification, if any</li> </ul> </li> <li>If the module contains Liberty, Visualizer displays the following icons in front of the module: <ul style="list-style-type: none"> <li> — Module contains Liberty information</li> <li> — Module is defined as a cell using <code>`celldefine</code> attribute</li> <li> — Module is defined as a cell using <code>`celldefine</code> attribute and contains Liberty information</li> </ul> </li> </ul> | Available by default |

| Window            | Debug Feature   | vopt option          |
|-------------------|---|----------------------|
| Logic Cone        | <ul style="list-style-type: none"> <li>Highlights all Power Aware objects in the window with each power domain in a different color.</li> <li>Displays the Power Aware information when you hover over the Power Aware object.</li> </ul> | Available by default |
| Time Cone         | <ul style="list-style-type: none"> <li>Highlights all Power Aware objects in the window with each power domain in a different color.</li> <li>Displays the Power Aware information when you hover over the Power Aware object.</li> </ul> | Available by default |
| PA Crossings      | <ul style="list-style-type: none"> <li>Displays information about the power domain crossings.</li> <li>Displays the static RTL checks performed during the execution of the vopt command.</li> </ul>                                      | -pa_checks=s         |
| PA Domains        | Displays information about the power domains that you define in the UPF files.  | Available by default |
| PA SimChecks      | Displays information about all Power Aware dynamic checks that are performed during the Power Aware simulation.   | -pa_checks=d         |
| PA Liberty        | Displays information about the Liberty cell.  | Available by default |
| PA States         | Displays information about the PST and power states that you define in the UPF files.   | Available by default |
| PA Supply Network | Displays a schematic view of the UPF supply network you specify in the UPF files.   | Available by default |
| Schematic         | <ul style="list-style-type: none"> <li>Highlights all Power Aware objects in the window with each power domain in a different color.</li> <li>Displays the Power Aware information when you hover over the Power Aware object.</li> </ul> | Available by default |

| Window    | Debug Feature  | vopt option          |
|-----------|--|----------------------|
| Source    | <ul style="list-style-type: none"> <li>The title bar displays the power domain and simstate of the module. The title bar also shows the simstate behavior if you specify it using the <code>set_simstate_behavior</code> command.</li> <li>Highlights Power Aware information with different colors.</li> <li>Displays the following information when you hover over a Power Aware object: <ul style="list-style-type: none"> <li>Details of Power Aware cells, such as isolation, and level shifter.</li> <li>Attributes of a signal that you specify with the <code>set_port_attributes</code> command.</li> </ul> </li> </ul> | Available by default |
| Variables | <ul style="list-style-type: none"> <li>The Name column has a PA-Info section, which displays the Power Aware variables.</li> <li>Displays the Power Aware annotations, such as level shifter, and isolation, applied on the variable.</li> <li>Displays power domains, supply sets, supply nets, and ports.</li> <li>Driver and receiver tracing of the following UPF objects: <ul style="list-style-type: none"> <li>Supply Set</li> <li>Supply Net</li> <li>Supply Port</li> <li>Logic Net</li> <li>Logic Port</li> </ul> </li> </ul>  | Available by default |
| Wave      | <ul style="list-style-type: none"> <li>Displays current simstate, module name of the object if the object is originating from a <code>`celldefine</code> module, and name of the liberty file, if any, when you hover over a Power Aware object.</li> <li>The PA Activity Wave menu item displays all domain activity in the entire design.</li> <li>Highlights isolation, corruption, and retention.</li> </ul>   | Available by default |

## Debugging With Power Aware Apps

Power Aware apps are simulator-specific, Tcl-based APIs. With Power Aware debugging apps, including your own, you can catch bugs earlier in the design, determine the impact of low-power issues, and access UPF objects.

|  |            |
|--|------------|
| <b>Power Aware Debugging Apps Overview .....</b>       | <b>230</b> |
| <b>Running the Power Aware Apps .....</b>              | <b>230</b> |
| <b>Finding the Source of a Corrupt Signal.....</b>     | <b>231</b> |
| <b>Reporting Selected Power Aware Information.....</b> | <b>233</b> |

## Power Aware Debugging Apps Overview

Power Aware apps are simulator-specific, Tcl-based APIs. Using Power Aware debugging apps, you can catch bugs earlier in the design cycle, address low-power issues, and more readily access and manipulate UPF objects.

Use the Power Aware apps after the simulation is complete, and all of the waveforms, along with the Power Aware information, are available.

Tool-specific, predefined Power Aware apps are available at the following location:

```
<install_dir>/modeltech/upf_src/upf_query_cmds.do
```

You can create your own Power Aware apps, for example, *apps.tcl*, and source the apps in the Transcript window of Visualizer using the following command:

```
source apps.tcl
```

## Running the Power Aware Apps

Run the Power Aware apps in Visualizer Debug Environment for reporting, debugging, and checking purposes. Use the predefined Power Aware apps or create your own Power Aware apps.

### Procedure

1. 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:

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

2. (Optional) Create your own Power Aware app in a file, such as *apps.tcl*.

### Tip

- i** You may refer to the predefined apps while creating your own Power Aware apps at the following location:

```
<install_dir>/modeltech/upf_src/upf_query_cmds.do
```

3. Initialize the Power Aware apps using the `-pa_apps` argument, which also dumps the Power Aware information model database, *design.bin.padb*, in the current working directory:

```
vopt -pa_apps -pa_upf <upf_file> -o <optimized_output> -designfile  
<designfile>
```

4. Load your design in Visualizer:

```
visualizer -designfile <designfile>
```

5. (Optional) Load the Power Aware apps, which you have created:

```
source apps.tcl
```

6. Run the Power Aware apps in Visualizer:

For example:

```
query_supply_set /alu_tester/dut/PD_TOP_ss  
query_power_domain /alu_tester/dut/PD_TOP
```

## Finding the Source of a Corrupt Signal

An example usage of the Power Aware apps is to find the source of a corrupt signal.

### Procedure

1. Query the domain membership information of the corrupt signal, `dut/mc0/addr`:

```
query_power_domain_element dut/mc0/addr
```

Output from the tool:

```
# /interleaver_tester/dut/PD_mem_ctrl
```

2. Get the supplies of the UPF object (power domain, retention strategy, and so on):

```
query_power_domain /interleaver_tester/dut/PD_mem_ctrl -detailed
```

Output from the tool:

```
# {domain_name PD_mem_ctrl} {elements /interleaver_tester/dut/mc0}  
{supply /interleaver_tester/dut/PD_mem_ctrl.default_isolation  
/interleaver_tester/dut/PD_mem_ctrl.default_retention  
/interleaver_tester/dut/PD_mem_ctrl.isolate_ss  
/interleaver_tester/dut/PD_mem_ctrl.primary  
/interleaver_tester/dut/PD_mem_ctrl.retain_ss}  
{scope /interleaver_tester/dut}
```

3. Get relevant functions of the primary supply set:

```
query_supply_set /interleaver_tester/dut/PD_mem_ctrl.primary  
-detailed
```

Output from the tool:

```
# {set_name primary} {functions {power VDD_0d81_SW} {ground VSS}}
```

4. Check the value of the UPF primary power supply net:

```
examine dut/VDD_0d81_SW -time 50
```

Output from the tool

```
# OFF
```

5. Query isolation cells of the power domain containing the corrupt signal:

```
query_isolation -domain dut/PD_mem_ctrl *
```

Output from the tool:

```
# /interleaver_tester/dut/PD_mem_ctrl.mem_ctrl_iso_0  
/interleaver_tester/dut/PD_mem_ctrl.mem_ctrl_iso_1
```

6. Check the elements of both isolation cells to find out the exact isolation cell that has the corrupt signal:


```
query_isolation mem_ctrl_iso_0 -domain dut/PD_mem_ctrl -detailed
```

Output from the tool:

```
# {isolation_name mem_ctrl_iso_0}  
{domain /interleaver_tester/dut/PD_mem_ctrl}  
{elements {/interleaver_tester/dut/mc0/addr  
/interleaver_tester/dut/mc0/do_acpt}}  
{isolation_power_net /interleaver_tester/dut/VDD_iso@upfSupplyNetT}  
{isolation_ground_net /interleaver_tester/dut/VSS@upfSupplyNetT}  
{no_isolation0} {isolation_supply_set /interleaver_tester/dut/  
PD_mem_ctrl.mem_ctrl_iso_0.isolation_supply_set}  
{isolation_signal /interleaver_tester/dut/mc_iso@upfLogicNetT}  
{isolation_sense high} {clamp_value UPF_CLAMP_ZERO}  
{location PARENT} {force_isolation 0} {diff_supply_only 0}
```



### Note

 You can replace 5 and 6 steps with your own Tcl apps to loop on all isolation cells of this domain.

7. Get drivers of isolation power supply net:

```
pa_query_drivers /interleaver_tester/dut/VDD_iso@upfSupplyNetT
```

Output from the tool:

```
# /interleaver_tester/dut/VDD_iso@upfSupplyNetT <-  
/interleaver_tester/dut/VDD_iso
```

8. Check the value of UPF primary power supply net:

```
examine dut/VDD_iso -time 50
```

Output from the tool:

```
# OFF
```

## Reporting Selected Power Aware Information

An example usage of the Power Aware apps is to report selected Power Aware information, such as finding an isolation strategy whose isolation control is corrupted.

### Procedure

1. Query all power domains in the design:

```
foreach pd [query_power_domain *] { ... }
```

2. Query the isolation strategies of a power domain:

```
foreach iso [query $pd -property upf_isolation_strategies] { ... }
```

3. Query the handle of the isolation signal:

```
set isolation_signal $iso_ctrls(upf_control_signal)  
set lvar1 [vpi_handle_by_name $isolation_signal 0]  
set vcHandle [vpi_handle_vc $lvar1 $waveFileHandle]  
set waveValueIter [vpi_iterate_vc $vcHandle $waveMin $waveMax "*" "  
"*"]
```

4. Iterate over the changed value of the isolation signal:

```
while {[set valueHandle [vpi_scan $waveValueIter]]}  
{  
  set timeNs [vpi_get_str vpiTime $valueHandle]  
  set val [examine $isolation_signal -time $timeNs]  
}
```

5. Verify if the isolation signal is corrupted:

```
if {$val == "1'bx"} {  
  puts "PA_Check_Error: $timeNs ns, Isolation Control of Isolation  
  Strategy '$iso' is corrupted"  
}
```

# Appendix A

## UPF and Tcl Commands

---

Power Aware simulation supports various UPF and Tcl commands. The UPF commands are a part of the IEEE Std 1801 to express the power management architecture of a design. The Tcl commands are specific to the tool, which you specify in either a UPF or Tcl file.

|   |            |
|---|------------|
| <b>UPF Commands and Reference .....</b> | <b>236</b> |
| <b>Tcl Commands .....</b>               | <b>312</b> |

## UPF Commands and Reference

Power Aware simulation supports various UPF commands, options and attributes to express the power management architecture of a design. It also supports various query commands to obtain Power Aware information about the design.

|                                      |            |
|--------------------------------------|------------|
| <b>Supported UPF Versions.....</b>   | <b>236</b> |
| <b>Supported UPF Commands .....</b>  | <b>238</b> |
| <b>Supported Query Commands.....</b> | <b>293</b> |
| <b>Supported Attributes.....</b>     | <b>302</b> |

## Supported UPF Versions

Power Aware simulation supports the following UPF versions: UPF 3.1, 3.0, 2.1, 2.0, and 1.0.

Use the following links to read about the UPF versions:

- UPF 3.1 — IEEE Std 1801-2018:  
<https://standards.ieee.org/standard/1801-2018.html>
- UPF 3.0 — IEEE Std 1801-2015:  
<https://standards.ieee.org/standard/1801-2015.html>
- UPF 2.1 — IEEE Std 1801-2013:  
<https://standards.ieee.org/standard/1801-2013.html>
- UPF 2.0 — IEEE Std 1801-2009:  
<https://standards.ieee.org/standard/1801-2009.html>
- UPF 1.0 — A subset of IEEE Std 1801-2009 (UPF 2.0) that corresponds to the original Accellera UPF definition.

### Power Aware Simulation Modes

Select the syntax and semantics of your UPF commands by specifying the UPF version in the `vopt` command. If you want to use the legacy codes (such as UPF 3.0, 2.1, 2.0, and 1.0) in a context requiring the latest UPF features (such as UPF 3.1), specify the UPF version in the UPF file.

**Table A-1. UPF Command Syntax and Semantics**

| Setting                              | Syntax  | Semantics |
|--------------------------------------|---------|-----------|
| <code>vopt -pa_upfversion=3.1</code> | UPF 3.1 | UPF 3.1   |
| <code>vopt -pa_upfversion=3.0</code> | UPF 3.0 | UPF 3.0   |
| <code>vopt -pa_upfversion=2.1</code> | UPF 2.1 | UPF 2.1   |

**Table A-1. UPF Command Syntax and Semantics (cont.)**

| Setting  | Syntax  | Semantics |
|--|---|-----------|
| vopt -pa_upfversion=2.0 (default)  | UPF 2.0   | UPF 2.0   |
| vopt -pa_upfversion=1.0  | UPF 1.0   | UPF 1.0   |
| <ul style="list-style-type: none"><li>• vopt -pa_upfversion=3.1</li><li>• upf_version (in the UPF file) is 3.0, 2.1, 2.0, or 1.0</li></ul> | According to the upf_version selected in the UPF file | UPF 3.1   |
| <ul style="list-style-type: none"><li>• vopt -pa_upfversion=3.0</li><li>• upf_version (in the UPF file) is 2.1, 2.0, or 1.0</li></ul>      | According to the upf_version selected in the UPF file | UPF 3.0   |
| <ul style="list-style-type: none"><li>• vopt -pa_upfversion=2.1</li><li>• upf_version (in the UPF file) is 2.0, or 1.0</li></ul>           | According to the upf_version selected in the UPF file | UPF 2.1   |

## Related Topics

[Using the Three-Step Flow](#)


## Supported UPF Commands

The tool supports various UPF commands that you use to express the power management architecture of your design.

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             | The tool does not support the command or option.                           |
| Not applicable | The command or option is not a part of the specified UPF version standard. |
| Yes            | The tool completely supports the command or option.                        |
| Yes (partial)  | The tool partially supports the command or option.                         |
| Yes (legacy)   | The tool supports the legacy command.                                      |

### Tip

 The UPF commands follow the given syntax, and supports the use of asterisk (\*), un-escaped square brackets ([]) as indices, and synthesis-style hierarchical names for instances in the UPF commands.

```
<upf_command> -<option> <argument>
```

### Note

 The tool recognizes / character (forward slash) as the hierarchical separator.

**Table A-2. Supported UPF Commands**

| Command                              | Description  |
|--------------------------------------|--|
| <a href="#">add_domain_elements</a>  | Adds design elements to a power domain.  |
| <a href="#">add_port_state</a>       | Adds states to a port.   |
| <a href="#">add_power_state</a>      | Defines power states of an object.   |
| <a href="#">add_pst_state</a>        | Defines the states of each of the supply nets for one possible state of the design.  |
| <a href="#">add_state_transition</a> | Defines named transitions among power states of an object.   |
| <a href="#">add_supply_state</a>     | Adds states to a supply port, a supply net, or a supply set function.  |
| <a href="#">apply_power_model</a>    | 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> . |
| <a href="#">associate_supply_set</a> | Associates two or more supply sets.  |

**Table A-2. Supported UPF Commands (cont.)**

| Command                                   | Description  |
|---|--|
| <a href="#">begin_power_model</a>         | Begins the definition of a power model. This command is used in conjunction with the end_power_model command.  |
| <a href="#">bind_checker</a>              | Inserts checker modules and bind them to instances.  |
| <a href="#">connect_logic_net</a>         | Connects a logic net to a logic port.  |
| <a href="#">connect_supply_net</a>        | Connects a supply net to the supply ports.   |
| <a href="#">connect_supply_set</a>        | Connects a supply set to particular elements.  |
| <a href="#">create_composite_domain</a>   | Defines a composite domain comprised of one or more subdomains.  |
| <a href="#">create_hdl2upf_vct</a>        | Defines a VCT that can be used in converting HDL logic values into state type values.                          |
| <a href="#">create_logic_net</a>          | Defines a logic net.   |
| <a href="#">create_logic_port</a>         | Defines a logic port.  |
| <a href="#">create_power_domain</a>       | Defines a power domain and its characteristics.  |
| <a href="#">create_power_state_group</a>  | Creates a name for a group of related power states.  |
| <a href="#">create_power_switch</a>       | Defines a power switch.  |
| <a href="#">create_pst</a>                | Creates a power state table (PST).   |
| <a href="#">create_supply_net</a>         | Creates a supply net.  |
| <a href="#">create_supply_port</a>        | Creates a supply port on an instance.  |
| <a href="#">create_supply_set</a>         | Creates or updates a supply set, or updates a supply set handle.   |
| <a href="#">create_upf2hdl_vct</a>        | Defines VCT that can be used in converting UPF supply_net_type values into HDL logic values.                   |
| <a href="#">define_power_model</a>        | Defines a power model.   |
| <a href="#">describe_state_transition</a> | Describes the legality of a state transition.  |
| <a href="#">end_power_model</a>           | Terminates the definition of a power model. It is used in conjunction with the begin_power_model command.      |
| <a href="#">load_simstate_behavior</a>    | Loads the simstate behavior defaults for a library.  |
| <a href="#">load_upf</a>                  | Executes commands from the specified UPF file in the current scope or in the scope of each specified instance. |
| <a href="#">load_upf_protected</a>        | Loads a UPF file in a protected environment that prevents corruption of existing variables.                    |
| <a href="#">map_isolation_cell</a>        | Maps a particular isolation strategy to a library cell or range of library cells.                              |

**Table A-2. Supported UPF Commands (cont.)**

| Command                                    | Description   |
|--|---|
| <a href="#">map_level_shifter_cell</a>     | Maps a level shifter strategy to a simulation or implementation model.  |
| <a href="#">map_power_switch</a>           | Specifies which power-switch model to use for the implementation of the corresponding switch instance.                                  |
| <a href="#">map_retention_cell</a>         | Constrains implementation alternatives, or specifies a functional model, for retention strategies.                                      |
| <a href="#">name_format</a>                | Defines the format for constructing names of implicitly created objects.  |
| <a href="#">save_upf</a>                   | Creates a UPF file of the structures relative to the active or specified scope.   |
| <a href="#">set_design_attributes</a>      | Applies attributes to models or instances.  |
| <a href="#">set_design_top</a>             | Specifies the design top module.  |
| <a href="#">set_domain_supply_net</a>      | Sets the default power and ground supply nets for a power domain.   |
| <a href="#">set_equivalent</a>             | Declares that supply nets or supply sets are electrically or functionally equivalent.   |
| <a href="#">set_isolation</a>              | Specifies an isolation strategy.  |
| <a href="#">set_isolation_control</a>      | Specifies the control signals for a previously defined isolation strategy.  |
| <a href="#">set_level_shifter</a>          | Specifies a level shifter strategy.   |
| <a href="#">set_partial_on_translation</a> | Defines the translation of PARTIAL_ON.  |
| <a href="#">set_pin_related_supply</a>     | Defines the related power and ground pair for a library cell.   |
| <a href="#">set_port_attributes</a>        | Defines information on ports.   |
| <a href="#">set_power_switch</a>           | Extends an HDL model containing no more than acknowledge logic to complete switch definition.   |
| <a href="#">set_repeater</a>               | Specifies a repeater (buffer) strategy.   |
| <a href="#">set_retention</a>              | Specifies a retention strategy.   |
| <a href="#">set_retention_control</a>      | Specifies the control signals and assertions for a previously defined retention strategy.   |
| <a href="#">set_retention_elements</a>     | Creates a named list of elements whose collective state shall be maintained if retention is applied to any of the elements in the list. |
| <a href="#">set_scope</a>                  | Specifies the current scope.  |
| <a href="#">set_simstate_behavior</a>      | Specifies the simulation simstate behavior for a model or library.  |



**Table A-2. Supported UPF Commands (cont.)**

| Command                                | Description  |
|--|--|
| <a href="#">set_variation</a>          | Specifies the variation range for a supply source.   |
| <a href="#">sim_assertion_control</a>  | Controls the behavior of Verilog assertions during Power Aware simulation.   |
| <a href="#">sim_corruption_control</a> | Disables the corruptions of a specific set of design elements or types of design elements.   |
| <a href="#">sim_replay_control</a>     | Specifies initial blocks to be replayed when a domain powers up.   |
| <a href="#">upf_version</a>            | Retrieves the version of UPF being used to interpret UPF commands and documents the UPF version for which subsequent commands are written. |
| <a href="#">use_interface_cell</a>     | 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 Version | Support    |
|-------------|------------|
| UPF 3.1     | Deprecated |
| UPF 3.0     | Deprecated |
| UPF 2.1     | Deprecated |
| UPF 2.0     | Yes        |
| UPF 1.0     | Yes        |

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

## add\_port\_state


Adds states to a port.

## Support for UPF Standard

| UPF Version | Support      |
|-------------|--------------|
| UPF 3.1     | Yes (legacy) |
| UPF 3.0     | Yes (legacy) |
| UPF 2.1     | Yes (legacy) |
| UPF 2.0     | Yes          |
| UPF 1.0     | Yes          |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## Usage Notes

The tool supports non-standard UPF names in the -state option. For example:

```
add_port_state p1 -state {lv1 1.1}
```

## add\_power\_state

Defines power states of an object.

## Support for UPF Standard

| UPF Version | Support        | Comments                        |
|-------------|----------------|---------------------------------|
| UPF 3.1     | Yes (partial)  | Unsupported option: -power_expr |
| UPF 3.0     | Yes (partial)  |                                 |
| UPF 2.1     | Yes            |                                 |
| UPF 2.0     | Yes            |                                 |
| UPF 1.0     | Not applicable |                                 |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## Usage Notes

The -supply, -domain, and -complete options are introduced in UPF 2.1, but you can enable the options in UPF 2.0 by using the following argument:

```
vopt -pa_upfextensions=relaxedsyntax
```

## add\_pst\_state

Defines the states of each of the supply nets for one possible state of the design.

### Support for UPF Standard

| UPF Version | Support      |
|-------------|--------------|
| UPF 3.1     | Yes (legacy) |
| UPF 3.0     | Yes (legacy) |
| UPF 2.1     | Yes (legacy) |
| UPF 2.0     | Yes          |
| UPF 1.0     | Yes          |

#### Note

 See “[Table A-1](#)” to select the correct UPF version.


## add\_state\_transition

Defines named transitions among power states of an object.

### Support for UPF Standard

| UPF Version | Support | Comments                                 |
|-------------|---------|--|
| UPF 3.1     | Yes     | Unsupported option: -model and -instance |
| UPF 3.0     | Yes     |  |
| UPF 2.1     | No      |  |
| UPF 2.0     | No      |  |
| UPF 1.0     | No      |  |

**Note**

 By default, the tool enables UPF 2.0 power state semantics. To enable UPF 3.0 power state semantics, use either `-pa_enable=powerstatesemantics` or `-pa_upfversion=3.0` argument with the `vopt` command. See “[Table A-1](#)”.

---

## add\_supply\_state

Adds states to a supply port, a supply net, or a supply set function.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Not applicable |
| UPF 2.0     | Not applicable |
| UPF 1.0     | Not applicable |

**Note**

 See “[Table A-1](#)” to select the correct UPF version.


---

## 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 Version | Support        | Comment                                      |
|-------------|----------------|--|
| UPF 3.1     | Yes (partial)  | Unsupported option: <code>-parameters</code> |
| UPF 3.0     | Yes (partial)  |  |
| UPF 2.1     | Yes            |  |
| UPF 2.0     | Not applicable |  |
| UPF 1.0     | Not applicable |  |

**Note** See “[Table A-1](#)” to select the correct UPF version.

## associate\_supply\_set

Associates two or more supply sets.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

**Note** See “[Table A-1](#)” to select the correct UPF version.

## 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 Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Yes            |
| UPF 2.0     | Not applicable |
| UPF 1.0     | Not applicable |

**Note** See “[Table A-1](#)” to select the correct UPF version.

## Usage Notes

Specify an alphanumeric string with no spaces in *power\_model\_name*.

## bind\_checker

Inserts checker modules and bind them to instances.

## Support for UPF Standard

| UPF Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |
| UPF 1.0     | Yes     |

---

### Note



See “[Table A-1](#)” to select the correct UPF version.

---

## Usage Notes

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 Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

### Usage Notes

The `-reconnect` option is introduced in UPF 2.1, but you can enable the option in UPF 2.0 by using the following argument:

```
vopt -pa_upfextensions=relaxedsyntax
```

## connect\_supply\_net

Connects a supply net to the supply ports.

## Support for UPF Standard

| UPF Version | Support       | Comment  |
|-------------|---------------|--|
| UPF 3.1     | Yes           |  |
| UPF 3.0     | Yes           |  |
| UPF 2.1     | Yes (partial) | Unsupported argument of the -pg_type option: <i>&lt;element_list&gt;</i>                               |
| UPF 2.0     | Yes (partial) | Unsupported options: <i>&lt;element_list&gt;</i> argument of the -pg_type option, and -rail_connection |
| UPF 1.0     | Yes (partial) | Unsupported option: -rail_connection   |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## Usage Notes

- The tool creates a port when the connect\_supply\_net command identifies a port that is missing in the HDL, but is specified as a pg\_pin in the Liberty model.
- The tool generates an error when the connect\_supply\_net command identifies a port that is missing in the HDL, and is not specified as a pg\_pin in the Liberty model.
- The tool supports connections to internal wires or registers representing supply ports.
- If you specify the -cells and -domain options together, the tool filters the effective element list to pins of the appropriate type (power or ground) that exists on the instances of the specified cells in the cell\_list present in the extent of the specified domain.

## Precedence Order

The tool issues a warning if you specify two connect\_supply\_\* commands that results in the same precedence, and follows the listed semantics:

- **Two connect\_supply\_net Commands** — The tool honors the command that you specify latter in the UPF file.
- **A connect\_supply\_net and a connect\_supply\_set command** — The tool honors the connect\_supply\_net command.

## connect\_supply\_set

Connects a supply set to particular elements.



## Support for UPF Standard

| UPF Version | Support        | Comment                               |
|-------------|----------------|---------------------------------------|
| UPF 3.1     | Yes (partial)  | Unsupported option: -exclude_elements |
| UPF 3.0     | Yes (partial)  |                                       |
| UPF 2.1     | Yes (partial)  |                                       |
| UPF 2.0     | Yes (partial)  |                                       |
| UPF 1.0     | Not applicable |                                       |

### Note

 See “[Table A-1](#)” to select the correct UPF version.

## Usage Notes

- The -transitive option is introduced in UPF 2.1, but you can enable the option in UPF 2.0 by using the following argument:  

```
vopt -pa_upfextensions=relaxedsyntax
```
- 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.
- If you specify a connect\_supply\_net and a connect\_supply\_set command with the same precedence, the tool honors the connect\_supply\_net command.

## create\_composite\_domain

Defines a composite domain comprised of one or more subdomains.

## Support for UPF Standard

| UPF Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |

| UPF Version | Support        |
|-------------|----------------|
| UPF 1.0     | Not applicable |

---

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

---

## Usage Notes

- The -supply option is introduced in UPF 2.1, but you can enable the option for UPF 2.0 by using the following argument:

```
vopt -pa_upfextensions=relaxedsyntax
```

- 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 tool 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 tool 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 tool 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 tool 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 tool displays an error if the primary supply handle already exists in a subdomain and points to a different supply set.

- `create_power_switch`

The tool creates the power switch in the creation scope of the composite domain.

- `create_supply_net`

The tool 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 tool applies the command without the `-reuse` option.

- `create_supply_port`

The tool 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 tool does not apply the command to that subdomain.

For example:

```
create_composite_domain cd -subdomains {pd2 pd3} -supply {primary
pdsb_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 — The tool 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 tool 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 tool 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 Version | Support       | Comment   |
|-------------|---------------|---|
| UPF 3.1     | Yes (partial) | Unsupported argument of <code>-hdl_type</code> option: user-defined |
| UPF 3.0     | Yes (partial) |   |
| UPF 2.1     | Yes (partial) |   |
| UPF 2.0     | Yes (partial) |   |
| UPF 1.0     | Yes (partial) |   |

#### Note

 See “[Table A-1](#)” to select the correct UPF version.


## create\_logic\_net

Defines a logic net.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

---

## Usage Notes

You can use *net\_name* as a control signal.


## create\_logic\_port

Defines a logic port.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

**Note**

 See “[Table A-1](#)” to select the correct UPF version.


---

## create\_power\_domain

Defines a power domain and its characteristics.

### Support for UPF Standard

| UPF Version | Support       | Comment                              |
|-------------|---------------|--------------------------------------|
| UPF 3.1     | Yes           |                                      |
| UPF 3.0     | Yes           |                                      |
| UPF 2.1     | Yes (partial) | Unsupported option: -simulation_only |
| UPF 2.0     | Yes (partial) |                                      |
| UPF 1.0     | Yes           |                                      |

**Note** See “[Table A-1](#)” to select the correct UPF version.

## Usage Notes

- The -atomic and -available\_supplies options are introduced in UPF 2.1, you can enable the option for UPF 2.0 by using the following argument:

```
vopt -pa_upfextensions=relaxedsyntax
```

- Explicit instance names in the -elements option take precedence over those from wildcard expansion.
- You cannot explicitly add an instance of a SystemVerilog interface to the extent of a power domain. For example, the tool 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 Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Not applicable |
| UPF 2.0     | Not applicable |
| UPF 1.0     | Not applicable |

**Note** See “[Table A-1](#)” to select the correct UPF version.

## create\_power\_switch

Defines a power switch.

## Support for UPF Standard

| UPF Version | Support       | Comment   |
|-------------|---------------|---|
| UPF 3.1     | Yes (partial) | <ul style="list-style-type: none"><li>Unsupported option: -switch_type</li><li>Supported option (not a part of IEEE Std 1801): -output_voltage</li></ul> See “Usage Notes”. |
| UPF 3.0     | Yes (partial) |   |
| UPF 2.1     | Yes (partial) | <ul style="list-style-type: none"><li>Unsupported option: -update</li><li>Supported option (not a part of IEEE Std 1801): -output_voltage</li></ul> See “Usage Notes”.      |
| UPF 2.0     | Yes           | Supported option (not a part of IEEE Std 1801):<br>-output_voltage<br>See “Usage Notes”.  |
| UPF 1.0     | Yes           |   |

---

### Note



See “Table A-1” to select the correct UPF version.

---

## Usage Notes

- If the boolean expression of an ON or PARTIAL\_ON state evaluates to X or Z, and the corresponding state of the input supply port is FULL\_ON or PARTIAL\_ON, then the output of the switch is set to UNDETERMINED state.
- If you specify a boolean\_function in the -ack\_port option, the tool drives the result of the boolean\_function on the ack port with a delay of -ack\_delay time units after a control port transition.
- The -instances option is introduced in UPF 2.1, but you can enable the option for UPF 2.0 by using the following argument:

```
vopt -pa_upfextensions=relaxedsyntax
```

- 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 Version | Support      |
|-------------|--------------|
| UPF 3.1     | Yes (legacy) |
| UPF 3.0     | Yes (legacy) |
| UPF 2.1     | Yes (legacy) |
| UPF 2.0     | Yes          |
| UPF 1.0     | Yes          |

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

## create\_supply\_net


Creates a supply net.

### Support for UPF Standard

| UPF Version | Support | Comment  |
|-------------|---------|--|
| UPF 3.1     | Yes     | Supported argument of the -resolve option (not a part of IEEE Std 1801): either, weak, and strong<br>See “ <a href="#">Usage Notes on -resolve either, weak, and strong Options</a> ”. |
| UPF 3.0     | Yes     |  |
| UPF 2.1     | Yes     |  |
| UPF 2.0     | Yes     |  |
| UPF 1.0     | Yes     |  |

---

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

---

## 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:

```
-resolve [<unresolved | one_hot | parallel | parallel_one_hot | either |  
weak | strong>]
```

The output voltage is governed by the voltage divider rule.

---

**Note**

 The either, weak and strong options of the -resolve argument are not a part of the IEEE Std 1801.


---

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

You can 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:
2. Refer to this resolution function in the UPF `create_supply_net` command with a fully qualified name. For example:

```
function supply_net_type resolution_func(supply_net_type bus[]);
```

```
create_supply_net N1 -resolve sv_package::resolution_function
```

## create\_supply\_port


Creates a supply port on an instance.

### Support for UPF Standard

| UPF Version | Support       | Comment  |
|-------------|---------------|--|
| UPF 3.1     | Yes (partial) | Unsupported argument of the -direction option: inout |
| UPF 3.0     | Yes (partial) |  |
| UPF 2.1     | Yes (partial) |  |
| UPF 2.0     | Yes (partial) |  |
| UPF 1.0     | Yes (partial) |  |

---

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## create\_supply\_set


Creates or updates a supply set, or updates a supply set handle.

### Support for UPF Standard

| UPF Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |

| UPF Version | Support        |
|-------------|----------------|
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

**Usage Notes**

The tool accepts the `-reference_gnd` option during parsing. However, it has no impact on simulation.


**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 Version | Support       | Comment   |
|-------------|---------------|---|
| UPF 3.1     | Yes (partial) | Unsupported argument of the <code>-hdl_type</code> option: user-defined |
| UPF 3.0     | Yes (partial) |   |
| UPF 2.1     | Yes (partial) |   |
| UPF 2.0     | Yes (partial) |   |
| UPF 1.0     | Yes (partial) |   |

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

**define\_power\_model**

Defines a power model.


**Support for UPF Standard**

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Not applicable |

| UPF Version | Support        |
|-------------|----------------|
| UPF 2.1     | Not applicable |
| UPF 2.0     | Not applicable |
| UPF 1.0     | Not applicable |

---

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

---

## describe\_state\_transition


Describes the legality of a state transition.

### Support for UPF Standard

| UPF Version | Support        | Comment  |
|-------------|----------------|--|
| UPF 3.1     | Deprecated     |  |
| UPF 3.0     | Deprecated     |  |
| UPF 2.1     | Yes            | Supported objects in the -object option: power domain, supply set, port, power switch, and PST |
| UPF 2.0     | Yes            |  |
| UPF 1.0     | Not applicable |  |

---

**Note**


 See “[Table A-1](#)” to select the correct UPF version.

---

### Usage Notes

---

**Note**

 The describe\_state\_transition command is deprecated in UPF 3.0. Use the [add\\_state\\_transition](#) command instead.

---

## 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 Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |
| UPF 1.0     | Yes     |

### Note

 See “[Table A-1](#)” to select the correct UPF version.


## load\_simstate\_behavior

Loads the simstate behavior defaults for a library.

## Support for UPF Standard

| UPF Version | Support        | Comment   |
|-------------|----------------|---|
| UPF 3.1     | Yes            | Unsupported argument of the -file option: <i>&lt;file_list&gt;</i><br>Load only one file per occurrence of the command. |
| UPF 3.0     | Yes            |   |
| UPF 2.1     | Yes            |   |
| UPF 2.0     | Yes            |   |
| UPF 1.0     | Not applicable |   |

### Note

 See “[Table A-1](#)” to select the correct UPF version.

## 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 Version | Support | Comment   |
|-------------|---------|---|
| UPF 3.1     | Yes     | The -version option is deprecated in UPF 3.0, but the tool supports it. |
| UPF 3.0     | Yes     |   |
| UPF 2.1     | Yes     |   |
| UPF 2.0     | Yes     |   |
| UPF 1.0     | Yes     |   |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## load\_upf\_protected

Loads a UPF file in a protected environment that prevents corruption of existing variables.

## Support for UPF Standard

| UPF Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |
| UPF 1.0     | Yes     |

---

### Note


 See “[Table A-1](#)” to select the correct UPF version.

---

## Usage Notes

---

### 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 tool 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 Version | Support    |
|-------------|------------|
| UPF 3.1     | Deprecated |
| UPF 3.0     | Deprecated |
| UPF 2.1     | Deprecated |
| UPF 2.0     | Yes        |
| UPF 1.0     | Yes        |

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

### Usage Notes

The `map_isolation_cell` command has an effect only if isolation cells are inserted. See “[set\\_isolation](#)”.

- The tool supports the `<libraryname/cellname>` format in the `-lib_cells` option. For example:  

```
map_isolation_cell PD_ONE -domain PD_SUP -lib_cells {LIB/NODE}
```
- The `lib_cell_type` specifies the attribute of a library cells that is used to identify cells that have isolation behavior and are otherwise identical to the inferred RTL behavior of the underlying sequential element
- The cell or module that you specify in the `-lib_model_name` option must be present in a library visible to the `vopt` command.
- When you use the `-lib_model_name` option of the `map_isolation_cell` command to specify the functional isolation model for a given isolation strategy, the tool inserts the instances of the functional model in the design.

To connect the logic and supply ports of the module, the tool follows the listed semantics, and in the given order:

- a. Based on the explicit connection details present in the `-port` option.
- b. Based on the attribute information that you specify in the Liberty or HDL source files.

### **Restriction**



When the tool performs connections using the attribute information, it connects only the logic and supply ports, and does not connect any non-ports, such as internal wires.

---

- The -ports option connects *<port\_name>* to *<net\_name>*, and accepts the following for *<net\_name>*:
  - A logic net name
  - A supply net name
  - One of the following symbolic references:
    - UPF\_ISO\_ENABLE (Specific to the tool) — Isolation control signal of associated isolation strategy
    - UPF\_ISO\_PWR (Specific to the tool) — Isolation power net of associated isolation strategy
    - UPF\_ISO\_GND (Specific to the tool) — Isolation ground net of associated isolation strategy
    - UPF\_GENERIC\_DATA — Port on which the cell is to be placed
    - UPF\_GENERIC\_OUTPUT — Output of the isolation cell
    - isolation\_signal — Isolation control signal of associated isolation strategy
    - isolation\_supply\_set.function\_name — 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 Version | Support    |
|-------------|------------|
| UPF 3.1     | Deprecated |
| UPF 3.0     | Deprecated |
| UPF 2.1     | Deprecated |
| UPF 2.0     | Yes        |
| UPF 1.0     | Yes        |

**Note**

 See “Table A-1” to select the correct UPF version.

**Usage Notes**

The tool supports the <libraryname/cellname> format in the -lib\_cells option. For example:

```
map_level_shifter_cell PD_ONE -domain PD_SUP -lib_cells {LIB/NODE}
```


**map\_power\_switch**

Specifies which power-switch model to use for the implementation of the corresponding switch instance.

**Support for UPF Standard**

| UPF Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |
| UPF 1.0     | Yes     |

**Note**

 See “Table A-1” to select the correct UPF version.


**map\_retention\_cell**

Constrains implementation alternatives, or specifies a functional model, for retention strategies.

**Support for UPF Standard**

| UPF Version | Support       | Comment  |
|-------------|---------------|--|
| UPF 3.1     | Yes (partial) | Unsupported option: -exclude_elements, and -port_map |
| UPF 3.0     | Yes (partial) |  |
| UPF 2.1     | Yes (partial) |  |
| UPF 2.0     | Yes (partial) |  |
| UPF 1.0     | Yes (partial) | Unsupported option: -port_map                        |

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

---

## Usage Notes

- The tool supports the <libraryname/cellname> format in the -lib\_cells option. For example:  

```
map_retention_cell PD_ONE -domain PD_SUP -lib_cells {LIB/NODE}
```
- 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 Version | Support       | Comment   |
|-------------|---------------|---|
| UPF 3.1     | Yes (partial) | Unsupported option: -implicit_supply_suffix, -implicit_logic_prefix, and -implicit_logic_suffix |
| UPF 3.0     | Yes (partial) |   |
| UPF 2.1     | Yes (partial) |   |
| UPF 2.0     | Yes (partial) |   |
| UPF 1.0     | No            |   |

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

---

## save\_upf

Creates a UPF file of the structures relative to the active or specified scope.

## Support for UPF Standard

| UPF Version | Support | Comment  |
|-------------|---------|--|
| UPF 3.1     | Yes     | <ul style="list-style-type: none"> <li>Supported option (not a part of IEEE Std 1801): -u</li> <li>Unsupported options in the uninterpreted mode: -scope, and -version</li> </ul> See “ <a href="#">Usage Notes</a> ”. |
| UPF 3.0     | Yes     |  |
| UPF 2.1     | Yes     |  |
| UPF 2.0     | Yes     |  |
| UPF 1.0     | Yes     |  |

### Note

 See “[Table A-1](#)” to select the correct UPF version.

## Usage Notes

The tool supports the following modes for using the save\_upf command:

- Interrupted Mode (default)

The output UPF 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. Any command or option that is not supported by Power Aware simulation is written as a comment at the end of the UPF file.

- Uninterrupted Mode

The output UPF file contains all the UPF commands 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 UPF file in uninterpreted mode, do either of the following:

- Use the following command:

```
save_upf -u
```


The -scope and -version options of the save\_upf command are not supported. The output UPF file is a complete replica of the original UPF file (but without any Tcl constructs).

- Use the following command:

```
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 Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

### Note



See “[Table A-1](#)” to select the correct UPF version.

---

### Usage Notes

- The `-is_leaf_cell` and `-is_macro_cell` options are introduced in UPF 2.1, but you can enable the options for UPF 2.0 by using the following argument:

```
vopt -pa_upfextensions=relaxedsyntax
```

- See “[Supported Attributes](#)” for the list of supported UPF attributes in the `-attribute` option.

## set\_design\_top

Specifies the design top module.

## Support for UPF Standard

| UPF Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |
| UPF 1.0     | Yes     |

### Note

 See “[Table A-1](#)” to select the correct UPF version.

## 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 Version | Support      |
|-------------|--------------|
| UPF 3.1     | Yes (legacy) |
| UPF 3.0     | Yes (legacy) |
| UPF 2.1     | Yes (legacy) |
| UPF 2.0     | Yes          |
| UPF 1.0     | Yes          |

### Note

 See “[Table A-1](#)” to select the correct UPF version.

## set\_equivalent


Declares that supply nets or supply sets are electrically or functionally equivalent.

## Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Yes            |
| UPF 2.0     | Not applicable |
| UPF 1.0     | Not applicable |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## set\_isolation

Specifies an isolation strategy.

## Support for UPF Standard

| UPF Version | Support       | Comment  |
|-------------|---------------|--|
| UPF 3.1     | Yes (partial) | <ul style="list-style-type: none"><li>Supported argument of the -location option: self, other, parent, fanout, <i>&lt;instance_name&gt;</i>. The <i>&lt;instance_name&gt;</i> argument is not a part of the IEEE Std 1801. See “locationinstance” in the “<a href="#">vopt -pa_upfextensions</a>” command.</li></ul>   |
| UPF 3.0     | Yes (partial) | <ul style="list-style-type: none"><li>Supported argument of the -location option: self, other, parent, fanout, <i>&lt;instance_name&gt;</i>. The <i>&lt;instance_name&gt;</i> argument is not a part of the IEEE Std 1801. See “locationinstance” in the “<a href="#">vopt -pa_upfextensions</a>” command.</li></ul>   |
| UPF 2.1     | Yes (partial) | <ul style="list-style-type: none"><li>Unsupported arguments of the -clamp_value option: <i>&lt;any&gt;</i></li><li>The <i>&lt;value&gt;</i> argument of the -clamp_value option does not support ports of unpacked type.</li><li>Supported argument of the -location option: self, other, parent, fanin, fanout, faninout, <i>&lt;instance_name&gt;</i>. The <i>&lt;instance_name&gt;</i> argument is not a part of the IEEE Std 1801. See “locationinstance” in the “<a href="#">vopt -pa_upfextensions</a>” command.</li></ul> |



| UPF Version | Support       | Comment   |
|-------------|---------------|---|
| UPF 2.0     | Yes (partial) | <ul style="list-style-type: none"> <li>Unsupported options: sink_off_clamp, source_off_clamp, and -transitive</li> <li>Supported argument of the -location option: self, other, parent, fanin, fanout, faninout, &lt;instance_name&gt;. The &lt;instance_name&gt; argument is not a part of the IEEE Std 1801. See “locationinstance” in the “<a href="#">vopt -pa_upfextensions</a>” command.</li> </ul> |
| UPF 1.0     | Yes           |   |

**Note**

See “[Table A-1](#)” to select the correct UPF version.

**Usage Notes**

- The tool enables you to use the -applies\_to option with the -source or -sink option, which is an error according to UPF 2.0.
- The -exclude\_elements, and -use\_equivalence options are introduced in UPF 2.1, but you can enable the options for UPF 2.0 by using the following argument:  

```
vopt -pa_upfextensions=relaxedsyntax
```
- The tool uses the following semantics for UPF 3.0 when the isolation control signal is asserted:
  - If you do not specify the -isolation\_supply option, the tool corrupts the output of the inferred isolation cell only when the isolation control signal is corrupted.
  - If you specify a supply set in the -isolation\_supply option, the tool corrupts the output of the inferred isolation cell in either of the following conditions:
    - The isolation control signal is corrupted.
    - The current power state of the specified isolation supply set has a simstate other than NORMAL.
  - If you specify an empty list ({} ) in the -isolation\_supply option, the isolation supply is connected to the primary supply set of the domain, and the isolation cell is a single rail isolation cell. The clamp value of the isolation strategy determines the type of the isolation cell:
    - If the clamp value is zero, NOR isolation cell is inferred.
    - If the clamp value is one, NAND isolation cell is inferred.

The tool corrupts the output of the inferred isolation cell in either of the following conditions:

- The isolation control signal is corrupted.
- The rail of the primary supply set of the domain, which is required for the clamp value of the isolation cell, has a supply state other than FULL\_ON. An error condition occurs if it is not possible to determine the state of the rail that is required for the clamp value.


## set\_isolation\_control

Specifies the control signals for a previously defined isolation strategy.

### Support for UPF Standard

| UPF Version | Support       | Comment   |
|-------------|---------------|---|
| UPF 3.1     | Deprecated    |   |
| UPF 3.0     | Deprecated    |   |
| UPF 2.1     | Deprecated    |   |
| UPF 2.0     | Yes (partial) | Unsupported argument of the -location option: sibling |
| UPF 1.0     | Yes (partial) |   |

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

## set\_level\_shifter


Specifies a level shifter strategy.

### Support for UPF Standard

| UPF Version | Support       | Comment   |
|-------------|---------------|---|
| UPF 3.1     | Yes           | Supported simstates of the -input_supply, -output_supply, and -internal_supply options: CORRUPT, and NORMAL   |
| UPF 3.0     | Yes           |   |
| UPF 2.1     | Yes (partial) | <ul style="list-style-type: none"><li>• Unsupported arguments of the -location option: automatic, and sibling</li><li>• Supported simstates of the -input_supply, -output_supply, and -internal_supply options: CORRUPT, and NORMAL</li></ul> |

| UPF Version | Support       | Comment   |
|-------------|---------------|---|
| UPF 2.0     | Yes (partial) | <ul style="list-style-type: none"> <li>• Unsupported option: -transitive</li> <li>• Unsupported argument of the -threshold option: <i>&lt;list&gt;</i></li> <li>• Unsupported arguments of the -location option: automatic, and sibling</li> <li>• Supported simstates of the -input_supply, -output_supply, and -internal_supply options: CORRUPT, and NORMAL</li> </ul> |
| UPF 1.0     | Yes           |   |

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

**Usage Notes**

- The -exclude\_elements, and -use\_equivalence options are introduced in UPF 2.1, but you can enable the options for UPF 2.0 by using the following argument:
 

```
vopt -pa_upfextensions=relaxedsyntax
```
- The tool enables you to use the -applies\_to option with the -source or -sink option, which is an error according to UPF 2.0.
- For the -location option, the tool follows the listed semantics:
  - If you specify the fanout argument, a level shifter is identified for placement at all fanout locations (for valid level shifters) and the count is incremented accordingly in the report.
  - If you specify the self, parent, sibling, or automatic argument, 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.

**set\_partial\_on\_translation**

Defines the translation of PARTIAL\_ON.

**Support for UPF Standard**

| UPF Version | Support | Comment |
|-------------|---------|---------|
| UPF 3.1     | Yes     |         |
| UPF 3.0     | Yes     |         |
| UPF 2.1     | Yes     |         |

| UPF Version | Support        | Comment  |
|-------------|----------------|--|
| UPF 2.0     | Yes            | Supported argument of the -full_on_tools, and -off_tools options: questa |
| UPF 1.0     | Not applicable |  |

---

**Note**

 See “[Table A-1](#)” to select the correct UPF version.


---

## Usage Notes

By default, the tool translates PARTIAL\_ON as OFF. Therefore, you must use this command to change the default translation behavior to FULL\_ON.

---

**Note**

 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.

---

## 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\_enable=rslnvnetcheck 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.

## set\_pin\_related\_supply

Defines the related power and ground pair for a library cell.

### Support for UPF Standard

| UPF Version | Support    |
|-------------|------------|
| UPF 3.1     | Deprecated |
| UPF 3.0     | Deprecated |
| UPF 2.1     | Deprecated |
| UPF 2.0     | Yes        |
| UPF 1.0     | Yes        |

#### Note



See “[Table A-1](#)” to select the correct UPF version.

### Usage Notes

The tool 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
```

See the “[vopt -warning](#)” command in the *Command Reference Manual*.

## set\_port\_attributes

Defines information on ports.

### Support for UPF Standard

| UPF Version | Support       | Comment   |
|-------------|---------------|---|
| UPF 3.1     | Yes (partial) | <ul style="list-style-type: none"><li>Unsupported option: <code>-feedthrough {port_name}</code><br/> <b>Note:</b> The tool supports the <code>-feedthrough</code> option.</li><li>Unsupported value of the <code>-pg_type</code> option: UPF-created supply port</li><li>Supported values of the <code>-clamp_value</code> option: hex, and binary</li><li>Supported simstates of the <code>-driver_supply</code> and <code>-receiver_supply</code> option: CORRUPT, NORMAL, CORRUPT_ON_ACTIVITY</li></ul> |
| UPF 3.0     | Yes (partial) | <ul style="list-style-type: none"><li>Unsupported value of the <code>-pg_type</code> option: UPF-created supply port</li><li>Supported values of the <code>-clamp_value</code> option: hex, and binary</li><li>Supported simstates of the <code>-driver_supply</code> and <code>-receiver_supply</code> option: CORRUPT, NORMAL, CORRUPT_ON_ACTIVITY</li></ul>  |
| UPF 2.1     | Yes (partial) | <ul style="list-style-type: none"><li>Unsupported value of the <code>-pg_type</code> option: UPF-created supply port.</li><li>Supported values of the <code>-clamp_value</code> option: hex, and binary</li><li>Supported simstates of the <code>-driver_supply</code> and <code>-receiver_supply</code> option: CORRUPT, NORMAL, CORRUPT_ON_ACTIVITY</li></ul>   |

| UPF Version | Support        | Comment   |
|-------------|----------------|---|
| UPF 2.0     | Yes (partial)  | <ul style="list-style-type: none"><li>• Unsupported options: -exclude_domains, and -transitive</li><li>• Unsupported value of the -pg_type option: UPF-created supply port</li><li>• Supported values of the -clamp_value option: hex, and binary</li><li>• Supported simstates of the -driver_supply and -receiver_supply option: CORRUPT, NORMAL, CORRUPT_ON_ACTIVITY</li></ul> |
| UPF 1.0     | Not applicable |   |

**Note**

See “[Table A-1](#)” to select the correct UPF version.

## Usage Notes

- The -clamp\_value, -source\_off\_clamp, and -sink\_off\_clamp options affect the filtering of ports specified by the [set\\_isolation](#) command.

## Related Supplies

Power Aware simulation assumes the driver and 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.

You can specify the driver\_supply and receiver\_supply using the following methods:

- **Using the set\_port\_attributes command**
  - If you specify the receiver\_supply to the output port or driver\_supply to the input port, the tool considers the specified supply and inserts the buffer at parent location.
  - If you specify the receiver\_supply to the input port, and this supply does not match with the power domain supply, the tool gives a vopt error message (vopt-9814) if there is single receiver and it is within the scope of the attributed port, else tool uses the receiver\_supply in path based source-sink analysis. The tool does not insert any buffer and there is no corruption.
  - If you specify driver\_supply to the output port, and this supply does not match with the power domain supply, the tool gives a vopt error message (vopt-9814) if there is single driver and it is within the scope of the attributed port, else the tool inserts buffers because of the driver\_supply, and if the supply is switched OFF, it leads to corruption.

Use the -warning argument of vopt to change the severity of (vopt-9814) message to a warning so that simulation may continue, and the tool inserts a buffer for the

driver\_supply with a prefix “\_DR”, which controls the power to the driver port in the self location:

```
vopt -warning 9814
```

See the “[vopt -warning](#)” command in the *Command Reference Manual*.

- By default, the tool uses anonymous supply as the driver\_supply and receiver\_supply for top-level ports, unless you specify them explicitly.

---

**Note**

---



Use the vopt -pa\_disable=anonymoustopsupply argument to use the top level domain supply as the driver\_supply and receiver\_supply instead of the anonymous supply.

---

- **Using the Liberty files**

If you define related\_power\_pin or related\_ground\_pin in the Liberty files, the tool adds a buffer with prefix “\_PD” for liberty corruption at self-location. The tool uses the related\_power and related\_ground option for input, and power\_down\_function for output. If the related supplies of the output port are not equivalent to the power domain supply with prefix “\_DR”, the tool inserts one more driver buffer at the output.

### Precedence Order

The tool inserts the driver and related power supply buffers based on either the Liberty file or the set\_port\_attribute command. If there is a conflict between the added buffers on a specific port, then the tool follows the listed precedence:

1. The Liberty file
2. The set\_port\_attribute

---

**Note**

---



Use the vopt -pa\_enable=overridelibrelsupplies argument to override the Liberty file precedence.

---

### The -clamp\_value Option

The -clamp\_value option is reported under the isolation strategy section in the Architecture Report.

- Specify the clamp value to the complete array, <my\_array> as following:

```
set_port_attributes -ports my_array -clamp_value 3'b101
```

- Specify the clamp value to the individual bits of the array, <my\_array[n]> as following:

```
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
```



### The -is\_analog Option

If you specify the `is_analog` attribute on a particular bit of the port, the tool considers the entire port as an analog port.

### 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 tool 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 Version | Support    |
|-------------|------------|
| UPF 3.1     | Deprecated |
| UPF 3.0     | Deprecated |
| UPF 2.1     | Deprecated |
| UPF 2.0     | Yes        |
| UPF 1.0     | Yes        |

---

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

### Usage Notes

The tool ignores the name clash error that occurs between the ports that you specify in the -input\_supply\_port or -output\_supply\_port options and the RTL.

## set\_repeater

Specifies a repeater (buffer) strategy.

### Support for UPF Standard

| UPF Version | Support        | Comment  |
|-------------|----------------|--|
| UPF 3.1     | Yes            | Supported simstates of the -repeater_supply option: CORRUPT, CORRUPT_ON_ACTIVITY, and NORMAL |
| UPF 3.0     | Yes            |  |
| UPF 2.1     | Yes            |  |
| UPF 2.0     | Not applicable |  |
| UPF 1.0     | Not applicable |  |

---

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

---


## set\_retention

Specifies a retention strategy.

### Support for UPF Standard

| UPF Version | Support       | Comment   |
|-------------|---------------|---|
| UPF 3.1     | Yes           | Supported options (not a part of IEEE Std 1801): assert_r_mutex, assert_s_mutex, and assert_rs_mutex  |
| UPF 3.0     | Yes           |   |
| UPF 2.1     | Yes (partial) | <ul style="list-style-type: none"><li>Unsupported options: -transitive</li><li>Supported options (not a part of IEEE Std 1801): assert_r_mutex, assert_s_mutex, and assert_rs_mutex</li></ul> |
| UPF 2.0     | Yes (partial) |   |
| UPF 1.0     | Yes           | Supported options (not a part of IEEE Std 1801): assert_r_mutex, assert_s_mutex, and assert_rs_mutex  |

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

## 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, use the [upf\\_dont\\_touch](#) attribute.

### 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 Version | Support    |
|-------------|------------|
| UPF 3.1     | Deprecated |
| UPF 3.0     | Deprecated |
| UPF 2.1     | Deprecated |
| UPF 2.0     | Yes        |
| UPF 1.0     | Yes        |

---

#### Note



See “[Table A-1](#)” to select the correct UPF version.

---


## 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 Version | Support        | Comment   |
|-------------|----------------|---|
| UPF 3.1     | Yes            |   |
| UPF 3.0     | Yes            |   |
| UPF 2.1     | Yes            |   |
| UPF 2.0     | Yes (partial)  | Unsupported argument of the -expand option: FALSE |
| UPF 1.0     | Not applicable |   |

### Note

 See “[Table A-1](#)” to select the correct UPF version.

## Usage Notes

- The default behavior of the -expand option is equivalent to -expand TRUE.

## set\_scope

Specifies the current scope.

## Support for UPF Standard

| UPF Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |
| UPF 1.0     | Yes     |

### Note

 See “[Table A-1](#)” to select the correct UPF version.

## Usage Notes

- The tool supports the use of generate blocks in the set\_scope command.

- 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 tool
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 tool 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 ..
set_scope /                /*Comment - The tool 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 Version | Support        | Comment                               |
|-------------|----------------|---------------------------------------|
| UPF 3.1     | Yes (partial)  | Unsupported option: -exclude_elements |
| UPF 3.0     | Yes (partial)  |                                       |
| UPF 2.1     | Yes (partial)  |                                       |
| UPF 2.0     | Yes            |                                       |
| UPF 1.0     | Not applicable |                                       |

---

#### Note



See “[Table A-1](#)” to select the correct UPF version.

---

### Usage Notes

The -elements option has higher priority over -model and -lib options.

## set\_variation

Specifies the variation range for a supply source.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Not applicable |
| UPF 2.0     | Not applicable |
| UPF 1.0     | Not applicable |

---

#### Note



See “[Table A-1](#)” to select the correct UPF version.

---




## sim\_assertion\_control

Controls the behavior of Verilog assertions during Power Aware simulation.

### Support for UPF Standard

| UPF Version | Support        | Comments                              |
|-------------|----------------|---------------------------------------|
| UPF 3.1     | Yes (partial)  | Unsupported option: -exclude_elements |
| UPF 3.0     | Not applicable |                                       |
| UPF 2.1     | Not applicable |                                       |
| UPF 2.0     | Not applicable |                                       |
| UPF 1.0     | Not applicable |                                       |

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

### Restrictions

- The command controls only Verilog assertions.
- The command does not apply to assertions within instances instantiated by the `bind_checker` command.
- The tool disables the assertions only when the simstate of the primary supply of the controlling domain is **CORRUPT**.
- The `-element` option with the `-model` option does not refine the element list.

### Examples

The following command selects the assertions `L1/clock_not_x`:

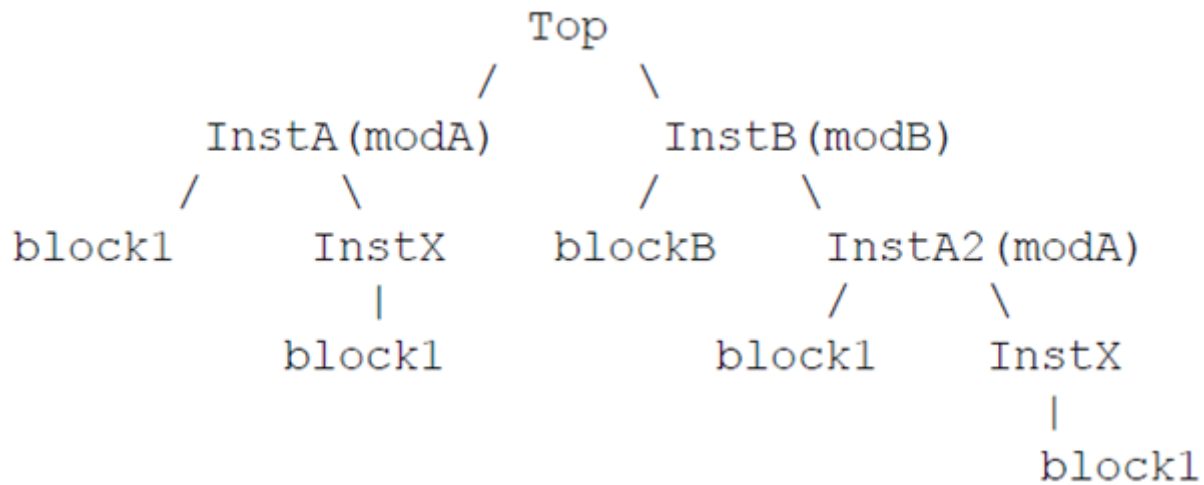
```
sim_assertion_control -elements L1/clock_not_x -type kill
```

The following command selects all assertions inside the instances that belong to domain `PD1`:

```
sim_assertion_control -domain PD1 -type suspend
```

The diagram below describes the hierarchy used for the subsequent examples:

**Figure A-1. Design Hierarchy**



The following command selects all assertions in module modA:

```
sim_assertion_control -model modA -controlling_domain PD1
```

The command selects InstA/Assert1, InstA/InstX/Assert1, InstB/InstA2/Assert1 and InstB/InstA2/instX/Assert1.

The following command selects all assertions labeled InstX/Assert1 in the module modA:

```
sim_assertion_control -model modA -elements InstX/Assert1
```

The command selects InstA/InstX/Assert1 and InstB/InstA2/InstX/Assert1.

## sim\_corruption\_control

Disables the corruptions of a specific set of design elements or types of design elements.

### Support for UPF Standard

| UPF Version | Support        | Comments |
|-------------|----------------|----------|
| UPF 3.1     | Yes            |          |
| UPF 3.0     | Not applicable |          |
| UPF 2.1     | Not applicable |          |
| UPF 2.0     | Not applicable |          |
| UPF 1.0     | Not applicable |          |

## sim\_replay\_control


Specifies initial blocks to be replayed when a domain powers up.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Not applicable |
| UPF 2.1     | Not applicable |
| UPF 2.0     | Not applicable |
| UPF 1.0     | Not applicable |

---

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## 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 Version | Support |
|-------------|---------|
| UPF 3.1     | Yes     |
| UPF 3.0     | Yes     |
| UPF 2.1     | Yes     |
| UPF 2.0     | Yes     |
| UPF 1.0     | Yes     |

---

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## use\_interface\_cell

Specifies the functional model and a list of implementation targets for isolation and level-shifting.

## Support for UPF Standard

| UPF Version | Support        | Comment   |
|-------------|----------------|---|
| UPF 3.1     | Yes (partial)  | Unsupported options: update_any, and -inverter_supply_set |
| UPF 3.0     | Yes (partial)  |   |
| UPF 2.1     | Yes (partial)  |   |
| UPF 2.0     | Yes (partial)  |   |
| UPF 1.0     | Not applicable |   |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## Usage Notes


When you use the `-force_function` option of the `use_interface_cell` command to specify the functional isolation model for a given isolation strategy, the tool inserts the instances of the functional model in the design.

To connect the logic and supply ports of the module, the tool follows the listed semantics, and in the given order:

1. Based on the explicit connection details present in the `-port_map` option.
2. Based on the attribute information that you specify in the Liberty or HDL source files.

---

### Restriction


 When the tool performs connections using the attribute information, it connects only the logic and supply ports, and does not connect any non-ports, such as internal wires.

---

## 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 1801-2015 standard, but the tool supports them. Use the upf\_query\_\* commands instead of the query\_\* commands. See “Supported Tcl APIs”.

**Table A-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 Version | Support        |
|-------------|----------------|
| UPF 3.1     | Yes            |
| UPF 3.0     | Yes            |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

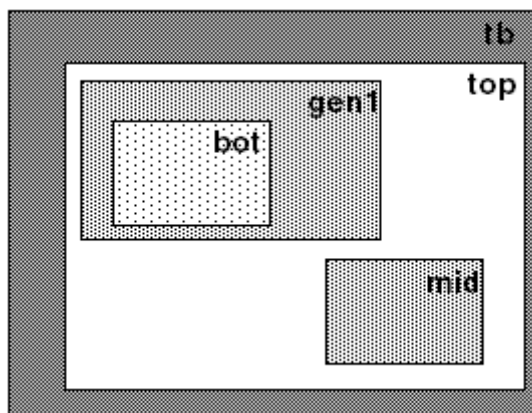
## Usage Notes

To write the output of the `find_objects` command to an external file, use the `-pa_dbgfindobj=<filename>` argument to the `vopt` command.

## Examples

The following example shows the output of the `find_objects` command.

**Figure A-2. Design Hierarchy**



```
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
```

## Limitations

For struct and record, the tool has the following limitations:

- If you specify the wildcard character (\*) in an array index, then you must specify it for all dimensions and the array element must be the last element in the pattern.

For example, the tool accepts the following patterns:

```
a.bs2[*][*]
a.b[0].c[*]
```

The tool does not accept the following patterns:

```
a.bs2[*][2]           //The wildcard character is not specified for all
                        //dimensions
a.bs2[*].c[0] }        //The wildcard character is not the last element
```

- If you specify the wildcard character (\*) in the field select, and it is not last in the pattern then the tool does not match it with the array elements unless you specify the array indexes without any wildcard character.

For example, the tool matches a.b\*[0].c pattern with the following patterns:

```
a.b1[0].c
a.bs2[0].c
```

The tool does not match a.\*.c pattern with the following patterns:

```
a.b1[0].c
a.b1[1].c
a.bs2[0].c
a.bs3[0][0].c
```

The tool does not match the a.b\*.c pattern with the following patterns:

```
a.b1[0].c
a.b1[1].c
a.bs2[0].c
a.bs2[1].c
a.bs3[0][0].c
```

- If you use the field select pattern, then after the first part-select ([y:x]) in the pattern, you should not further select any field, bit, or part.

For example, the tool accepts the following patterns:

```
a.bs2.c[2:0]
a.b[2:0]
a[1:0]
```

The tool does not accept the following patterns:

```
a.bs3[2:0].c           //Field is selected after part-select
a.bs3[1:0][0]          //Bit is selected after part-select
a.bs3[1:0][1:0]        //Part is selected after part-select
```

## query\_cell\_instances


Queries the instances of a mapped cell within the active scope.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## query\_cell\_mapped


Queries the cell that is mapped to a specified instance.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## query\_design\_attributes

Queries attributes for a design element or model.




## Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## query\_isolation


Queries information about previously defined isolation strategies for the specified power domain.

## Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## query\_port\_state

Queries the state information for a specified port.

## Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## query\_power\_domain


Queries a power domain.

## Support for UPF Standard

| UPF Version | Support        | Comments                 |
|-------------|----------------|--------------------------|
| UPF 3.1     | Deprecated     |                          |
| UPF 3.0     | Deprecated     |                          |
| UPF 2.1     | Yes (partial)  | Unsupported option: -all |
| UPF 2.0     | Yes (partial)  | Unsupported option: -all |
| UPF 1.0     | Not applicable |                          |

---

### Note

 See “[Table A-1](#)” to select the correct UPF version.

---


## query\_power\_domain\_element

Queries the domain membership information for a design element.

## Support for UPF Standard

| UPF Version | Support    |
|-------------|------------|
| UPF 3.1     | Deprecated |

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

**Note** See “[Table A-1](#)” to select the correct UPF version.

## query\_power\_state

Queries the state information for a power domain or supply set.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

**Note** See “[Table A-1](#)” to select the correct UPF version.

## query\_power\_switch

Queries information for a UPF power switch.


### Support for UPF Standard

| UPF Version | Support    |
|-------------|------------|
| UPF 3.1     | Deprecated |
| UPF 3.0     | Deprecated |
| UPF 2.1     | Yes        |
| UPF 2.0     | Yes        |

| UPF Version | Support        |
|-------------|----------------|
| UPF 1.0     | Not applicable |

---

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

---

## query\_pst


Queries a power state table.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes (legacy)   |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

**Note**

 See “[Table A-1](#)” to select the correct UPF version.

---


## query\_pst\_state

Queries state information for a PST.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes (legacy)   |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

**Note** See “[Table A-1](#)” to select the correct UPF version.

---

## query\_retention

Queries the retention strategy information for a domain.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

**Note** See “[Table A-1](#)” to select the correct UPF version.

---


## query\_retention\_control

Queries the retention control information for a domain.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Deprecated     |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

**Note** See “[Table A-1](#)” to select the correct UPF version.

---

## query\_supply\_net


Queries a supply net.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## query\_supply\_port


Queries a supply port.

### Support for UPF Standard

| UPF Version | Support        |
|-------------|----------------|
| UPF 3.1     | Deprecated     |
| UPF 3.0     | Deprecated     |
| UPF 2.1     | Yes            |
| UPF 2.0     | Yes            |
| UPF 1.0     | Not applicable |

---

#### Note

 See “[Table A-1](#)” to select the correct UPF version.

---

## Supported Attributes

Power Aware simulation supports UPF, Liberty and tool-specific attributes that you use to express the power intent in a UPF, RTL, and Liberty file.

The tool supports the following UPF attributes, which are a part of the IEEE Std 1801.

**Table A-4. Supported UPF Attributes**

|                                 |                            |
|---------------------------------|----------------------------|
| UPF_clamp_value                 | UPF_related_bias_ports     |
| UPF_feedthrough                 | UPF_related_ground_port    |
| UPF_is_analog                   | UPF_related_power_port     |
| UPF_is_hard_macro               | UPF_retention              |
| UPF_is_isolated                 | UPF_simstate_behavior      |
| UPF_is_soft_macro               | UPF_sink_off_clamp_value   |
| UPF_literal_supply <sup>1</sup> | UPF_source_off_clamp_value |
| UPF_pg_type                     | UPF_unconnected            |

1. By default, the tool supports the attribute in UPF 3.0 and above versions only. To enable support in all UPF versions, use the vopt -pa\_enable=literalsupply command.

The tool supports the following Liberty attributes, which are a part of the Liberty User Guides and Reference Manual Suite.

**Table A-5. Supported Liberty Attributes**

|                           |                     |
|---------------------------|---------------------|
| always_on                 | preset              |
| apply_power_down_function | related_bias_pin    |
| clear                     | related_ground_pin  |
| clock                     | related_power_pin   |
| is_analog                 | restore_action      |
| is_isolated               | restore_condition   |
| is_isolation_cell         | restore_edge_type   |
| is_level_shifter          | retention_cell      |
| is_macro_cell             | retention_condition |
| isolation_cell_data_pin   | retention_pin       |
| isolation_cell_enable_pin | save_condition      |
| level_shifter_data_pin    | std_cell_main_rail  |
| level_shifter_enable_pin  | switch_cell_type    |
| level_shifter_type        | switch_function     |
| pg_function               | switch_pin          |
| power_down_function       | user_pg_type        |

The tool supports the following attributes, which are not a part of any standard and are specific to the tool.

**Table A-6. Tool-Specific Attributes**

| Attribute  | Value(s)     | Equivalent UPF Commands  |
|--|--------------|--|
| <a href="#">ignore_upf_over_liberty</a>          | TRUE   FALSE | set_design_attributes -elements<br><design_top> -attribute<br>{ignore_upf_over_liberty TRUE}   |
| <a href="#">iso_nand</a>                         | TRUE   FALSE | set_design_attributes -attribute {iso_nand<br>TRUE}  |
| <a href="#">iso_nor</a>                          | TRUE   FALSE | set_design_attributes -attribute {iso_nor<br>TRUE}   |
| <a href="#">qpa_attr_active_reset_duration</a>   | <Twidth>     | set_design_attributes -elements<br><design_top> -attribute<br>{attr_active_reset_duration Twidth}  |
| <a href="#">qpa_attr_inactive_reset_duration</a> | <Tmax>       | set_design_attributes -elements<br><design_top> -attribute<br>{attr_inactive_reset_duration Tmax}  |
| <a href="#">qpa_dont_replay_init</a>             | TRUE   FALSE | set_design_attributes -elements<br><design_top> -attribute<br>{qpa_dont_replay_init TRUE}  |
| <a href="#">qpa_override_pbp_corruption</a>      | TRUE   FALSE | set_design_attributes -attribute<br>{override_pbp_corruption TRUE}   |
| <a href="#">qpa_power_switch_on_delay</a>        | <delay>      | set_design_attributes -elements<br><design_top> -attribute<br>{qpa_power_switch_on_delay delay}  |
| <a href="#">qpa_replay_init</a>                  | TRUE   FALSE | set_design_attributes -elements<br><design_top> -attribute {qpa_replay_init<br>TRUE}   |
| <a href="#">upf_dont_touch</a>                   | TRUE   FALSE | <ul style="list-style-type: none"> <li>• set_port_attributes -elements<br/>    &lt;design_top&gt; -attribute<br/>    {upf_dont_touch TRUE}</li> <li>• set_design_attributes -elements<br/>    &lt;design_top&gt; -attribute<br/>    {upf_dont_touch TRUE}</li> </ul> |
| <a href="#">upper_boundary_only</a>              | TRUE   FALSE | set_design_attributes -elements<br><design_top> -attribute<br>{upper_boundary_only TRUE}   |

### **ignore\_upf\_over\_liberty**

Ignores the load\_upf command provided there is a Liberty file corresponding to the scope. The attribute is not a part of the IEEE Std 1801.



Usage:

```
set_design_attributes -elements <design_top> -attribute  
{ignore_upf_over_liberty TRUE}
```

Assume that for a scope S1, there is a Liberty file, and a load\_upf command:

```
load_upf s1.upf -scope /top/S1
```

If you set the ignore\_upf\_over\_liberty attribute to TRUE, the tool ignores the load\_upf command.

---

**Note**

---



When you use the ignore\_upf\_over\_liberty attribute with the set\_design\_attributes command, the tool applies the attribute on the entire design. Hence, you must specify the top module in the -elements option. If you do not specify the top module in the -elements option, the tool flags a warning message.

---

## iso\_nand

Specifies that an isolation cell is a nand\_style isolation cell. The attribute is not a part of the IEEE Std 1801.

Usage:

```
set_design_attributes -attribute {iso_nand TRUE}
```

## iso\_nor

Specifies that an isolation cell is a nor\_style isolation cell. The attribute is not a part of the IEEE Std 1801

Usage:

```
set_design_attributes -attribute {iso_nor TRUE}
```

## qpa\_attr\_inactive\_reset\_duration

Sets the maximum time unit (Tmax) within which the asynchronous reset must be asserted after power up. The attribute is not a part of the IEEE Std 1801.

Usage:

```
set_design_attributes -elements <design_top> -attribute  
{qpa_attr_inactive_reset_duration Tmax}
```

where Tmax represents integer time units.

For example:

```
set_design_attributes -elements <design_top> -attribute  
{qpa_attr_inactive_reset_duration 10}
```

Refer to the *QPA\_NRET\_ASYNCFF* check in “[Table 4-19](#)” for more information.

---

**Note**



When you use the `qpa_attr_inactive_reset_duration` attribute with the `set_design_attributes` command, the tool applies the attribute on the entire design. Hence, you must specify the top module in the `-elements` option. If you do not specify the top module in the `-elements` option, the tool flags a warning message.

---

### **qpa\_attr\_active\_reset\_duration**

Sets the minimum time unit (Twidth) for which the asynchronous reset must remain asserted. The attribute is not a part of the IEEE Std 1801.

---

**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.

---

Usage:

```
set_design_attributes -elements <design_top> -attribute  
{qpa_attr_active_reset_duration Twidth}
```

where Twidth represents integer time units.

For example:

```
set_design_attributes -elements <design_top> -attribute  
{qpa_attr_active_reset_duration 5}
```

Refer to the *QPA\_NRET\_ASYNCFF* check in “[Table 4-19](#)” for more information.

---

**Note**



When you use the `qpa_attr_active_reset_duration` attribute with the `set_design_attributes` command, the tool applies the attribute on the entire design. Hence, you must specify the top module in the `-elements` option. If you do not specify the top module in the `-elements` option, the tool flags a warning message.

---


### **qpa\_dont\_replay\_init**

Disables re-triggering of a Verilog initial block at power up. The 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 -elements <design_top> -attribute  
{qpa_dont_replay_init TRUE}
```

Assume you include the following two commands in a UPF file:


```
set_design_attributes -elements <design_top> -attribute qpa_replay_init  
TRUE  
  
set_design_attributes -attribute {qpa_dont_replay_init TRUE}  
-elements PD_Core
```

In this case, the tool re-triggers all initial blocks present in the design, except the initial blocks present in the power domain, `PD_Core`.

---

**Note**

---

 When you use the `qpa_dont_replay_init` attribute with the `set_design_attributes` command, the tool applies the attribute on the entire design. Hence, you must specify the top module in the `-elements` option. If you do not specify the top module in the `-elements` option, the tool flags a warning message.

---

## **qpa\_override\_pbp\_corruption**

When the attribute is set to `TRUE`, the tool applies Liberty-based corruption to all macro cells, even when the models are Power Aware. The 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 tool 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 attribute is not a part of the IEEE Std 1801.

---

**Note**

 The attribute `qpa_replay_init` does not re-trigger an initial block containing the following statements: `forever`, `fork join`, `break continue`, `wait`, `disable`, or `rand seq`.

---

Usage:

```
set_design_attributes -elements <design_top> -attribute {qpa_replay_init TRUE}
```


Examples:

```
set_design_attributes -elements <design_top> -attribute {qpa_replay_init TRUE} -models { t* }
```

```
set_design_attributes -elements { top/bot1/sram } -attribute {qpa_replay_init TRUE} -transitive TRUE
```

---

**Note**

 When you use the `qpa_replay_init` attribute with the `set_design_attributes` command, the tool applies the attribute on the entire design. Hence, you must specify the top module in the `-elements` option. If you do not specify the top module in the `-elements` option, the tool flags a warning message.

---

### Priority of Commands

If a UPF file contains multiple `set_design_attributes` commands with the `qpa_replay_init` attribute set to `TRUE`, then the tool 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, `ModuleForRetrigger`:

```
set_design_attributes -attribute {qpa_replay_init TRUE} -elements NamedInitialBlock -models ModuleForRetrigger
```

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}
```

## qpa\_power\_switch\_on\_delay

Delays the output supply of a power switch. The attribute is not a part of the IEEE Std 1801.

Usage:

```
set_design_attributes -elements <design_top> -attribute  
{qpa_power_switch_on_delay <delay>}
```

where *<delay>* represents integer time units.

For example:

```
set_design_attributes -elements <design_top> -attribute  
{qpa_power_switch_on_delay 5}
```

---

### Note



When you use the `qpa_power_switch_on_delay` attribute with the `set_design_attributes` command, the tool applies the attribute on the entire design. Hence, you must specify the top module in the `-elements` option. If you do not specify the top module in the `-elements` option, the tool flags a warning message.

---

## upf\_dont\_touch

Excludes Power Aware behavior on certain instances, modules, or signals. The attribute is not a part of the IEEE Std 1801.

The following are some of the reasons to exclude Power Aware behavior on certain instance, modules, or signals:

- They 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 a module, an architecture, or an 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 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.

Examples:

```
set_design_attributes -attribute {upf_dont_touch TRUE} -models fpu_*
set_design_attributes -attribute {upf_dont_touch TRUE} -elements top/inst2

set_design_attributes -attribute {upf_dont_touch TRUE}
-elements inst1/blk/*/s*

set_design_attributes -attribute {upf_dont_touch TRUE}
-elements top/inst2/sig1[2:0]

set_port_attributes -attribute {upf_dont_touch TRUE} -models mid
-ports {p1 p2}
```

The following is an HDL example using the attribute:

```
(*upf_dont_touch = "TRUE"*)
module mid();
...
endmodule
```

---

#### **Note**



When you use the `upf_dont_touch` attribute with the `set_design_attributes` command, the tool applies the attribute on the entire design. Hence, you must specify the top module in the `-elements` option. If you do not specify the top module in the `-elements` option, the tool flags a warning message.

---

### **upper\_boundary\_only**

Applies isolation on upper boundary ports only. The attribute is not a part of the IEEE Std 1801.

For example:

```
set_design_attributes -elements <design_top> -attribute  
{upper_boundary_only TRUE}
```

---

**Note**

---



When you use the `upper_boundary_only` attribute with the `set_design_attributes` command, the tool applies the attribute on the entire design. Hence, you must specify the top module in the `-elements` option. If you do not specify the top module in the `-elements` option, the tool flags a warning message.

---

## Related Topics

[set\\_port\\_attributes](#)

## 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.

---

**Note**

---



The Tcl commands are tool-specific and are not a part of the IEEE Std 1801.

---

**Table A-7. Tool-Specific Tcl Commands**

| Command                                       | Description  |
|---|--|
| <a href="#">describe_state_cross_coverage</a> | Provides results of cross coverage for a specified list of power domains in a design.  |
| <a href="#">getenv</a>                        | Returns the value of any environment variable.   |
| <a href="#">pa_checks</a>                     | Provides granular control of static RTL and dynamic checks.  |
| <a href="#">save_checks_config</a>            | Saves the configuration of static RTL and dynamic checks performed by the tool, to a specified file.   |
| <a href="#">set_corruption_extent</a>         | Changes the corruption extent of the power domains created by the UPF file.  |
| <a href="#">set_feedthrough_object</a>        | 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.        |
| <a href="#">set_pgpin_vct</a>                 | Specifies the VCT to be used for pins of any particular <code>pg_type</code> attribute.  |
| <a href="#">set_related_supply_net</a>        | 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. |



## 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>]
```

### Description

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_upf` or `-pa_tclfile` argument.

For a depth of 1 (default), the tool 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 tool provides cross coverage results for this particular group of power domains.

### Arguments

- `-domains <domain_list>`  
Creates a list of dependent power domains for which the tool 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.

### Examples

```
describe_state_cross_coverage -domains PDSYS1 PDSYS2 -depth=2
```

## getenv

Returns the value of any environment variable.

### Syntax

```
getenv {<variable>}
```

### Description

Specify the getenv command in either a UPF or a Tcl file, and include the file in the vopt run using the -pa\_upf or -pa\_tclfile argument

### Arguments

- <variable>

The name of an environment variable. Enclose the variable in curly braces ({}).

### 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-chkIds> [-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>] [-psts <list-of-pst-names>]
          [-chksctrl_expr {<ctrl-signal-expression>}]
```

### Note



Not all arguments of the `pa_checks` command may be applicable to all Power Aware checks. If an argument is not applicable to a Power Aware check, then the tool flags an appropriate warning message and discards that argument. See “Supported Options — Dynamic Checks”.

### Description

Specify the `pa_checks` command in either a UPF or Tcl file, and include the file in the vopt run using the `-pa_upf` or `-pa_tclfile` argument.

For multiple `pa_checks` commands that apply to the same check, the tool resolves the priority of the commands based on the precedence of arguments. See “[Precedence Order of Arguments](#)”.

### Arguments

- `-enable | -disable`

(Optional) Enables or disables the checks selected by the `-checkIds` argument in a `pa_checks` command.

For example:

```
pa_checks -enable -checkIds {smi sri}
pa_checks -disable -checkIds {sml}
```

If you do not use either the `-enable` or the `-disable` argument with a `pa_checks` command, then `-enable` is applied by default. If you use the `-enable` and `-disable` arguments on the same check in the `pa_checks` command, the last argument takes precedence.

The tool interprets the following command:

```
pa_checks -disable -chksctrl_expr {<expr>}
```

as the following:

```
pa_checks -enable -chksctrl_expr {!<expr>}
```

- **-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 static RTL and 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 static RTL and 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 the “[Power Aware Checks Command Reference](#)” section.

- **-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 **-domains** 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} -domains PD_mid1 -strategies
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}
```

- -psts <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 -psts 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} -psts {PowerStateTable_3}
```

- **-chksctrl\_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}
```

**Table A-8. Supported Options — Dynamic Checks**


| Dynamic Check     | Mnemonic              | Supported Option   |
|-------------------|-----------------------|--|
| -pa_checks=cp     | QPA_CTRL_SIG_CRPT     | -domains, -strategies, -powerswitches, -elements, -transitive, -model, -chksctrl_exp<br>You can specify the scope of strategy in the -elements and the -model options.   |
| -pa_checks=icp    | QPA_ISO_CLAMP_CHK     | -domains, -strategies, -elements, -transitive, -model, -sources, -sinks, -chksctrl_expr<br>The -model option matches the instance of port for which an error is reported. It is exclusive with the -transitive option. |
| -pa_checks=idp    | QPA_ISO_DIS_PSO       | -domains, -strategies, -sources, -sinks  |
| -pa_checks=iep    | QPA_ISO_EN_PSO        | -domains, -strategies, -sources, -sinks  |
| -pa_checks=idpcoa | QPA_ISO_DIS_COA       | -domains, -strategies, -sources, -sinks  |
| -pa_checks=iepcoa | QPA_ISO_EN_COA        | -domains, -strategies, -sources, -sinks  |
| -pa_checks=ifc    | QPA_ISO_FUNC_CHK      | -domains, -strategies, -elements, -transitive, -model, -sources, -sinks, -chksctrl_expr<br>The -model option matches the instance of port for which an error is reported. It is exclusive with the -transitive option. |
| -pa_checks=ira    | QPA_ISO_REDUNDANT_ACT | -domains, -strategies, -sources, -sinks  |

**Table A-8. Supported Options — Dynamic Checks (cont.)**

| Dynamic Check   | Mnemonic                           | Supported Option   |
|-----------------|------------------------------------|--|
| -pa_checks=irc  | QPA_ISO_PORT_TOGGLE                | -domains, -strategies, -elements, -transitive, -model, -sources, -sinks, -chkscrtl_expr<br><br>The -model option matches the instance of port for which an error is reported. It is exclusive with the -transitive option. |
| -pa_checks=ircn | QPA_ISO_PORT_TOGGLE_NEDGE          | -domains, -strategies, -elements, -transitive, -model, -sources, -sinks, -chkscrtl_expr<br><br>The -model option matches the instance of port for which an error is reported. It is exclusive with the -transitive option. |
| -pa_checks=ircp | QPA_ISO_PORT_TOGGLE_POS<br>EDGE    | -domains, -strategies, -elements, -transitive, -model, -sources, -sinks, -chkscrtl_expr<br><br>The -model option matches the instance of port for which an error is reported. It is exclusive with the -transitive option. |
| -pa_checks=it   | QPA_ISO_ON_ACT                     | -domains, -strategies, -elements, -transitive, -model, -sources, -sinks, -chkscrtl_expr<br><br>The -model option matches the instance of port for which an error is reported. It is exclusive with the -transitive option. |
| -pa_checks=npu  | QPA_NRET_ASYNCFF                   | -domains, -elements, -transitive, -model, -chkscrtl_expr   |
| -pa_checks=t    | QPA_PD_OFF_ACT,<br>QPA_PD_BIAS_ACT | -domains, -elements, -transitive, -model, -sources, -sinks   |
| -pa_checks=rcs  | QPA_RET_CLK_STATE                  | -domains, -strategies, -elements, -chkscrtl_expr, -transitive, -model  |
| -pa_checks=rop  | QPA_RET_OFF_PSO                    | -domains, -strategies, -elements, -chkscrtl_expr, -transitive, -model  |
| -pa_checks=rpo  | QPA_RET_PD_OFF                     | -domains, -strategies, -elements, -chkscrtl_expr, -transitive, -model  |



**Table A-8. Supported Options — Dynamic Checks (cont.)**

| Dynamic Check     | Mnemonic                      | Supported Option   |
|-------------------|-------------------------------|--|
| -pa_checks=rsa    | QPA_RET_SEQ_ACT               | -domains, -strategies, -elements, -chksctrl_expr, -transitive, -model  |
| -pa_checks=rtcon  | QPA_RET_ON_RTC                | -domains, -strategies, -elements, -chksctrl_expr, -transitive, -model  |
| -pa_checks=rtcoff | QPA_RET_OFF_RTC               | -domains, -strategies, -elements, -chksctrl_expr, -transitive, -model  |
| -pa_checks=rtctog | QPA_RET_RTC_TOGGLE            | -domains, -strategies, -elements, -chksctrl_expr, -transitive, -model  |
| -pa_checks=pis    | QPA_UPF_ILLEGAL_STATE_REACHED | -elements, -transitive, -model<br>-chksctrl_expr<br> <b>Note:</b> Supported only for port and PST state, and not for power state.<br>You can specify the scope of strategy in the -elements and the -model options. |
| -pa_checks=uil    | QPA_UPF_INCORRECT_LS_CHK      | -domains, -strategies, -elements, -model, -sources, -sinks<br>The -elements option should be either of source or sink ports. It cannot be instance.<br>The -model option matches the instance of source or sink ports for which an error is reported.  |
| -pa_checks=umi    | QPA_UPF_MISSING_ISO_CHK       | -elements, -model, -sources, -sinks<br>The -elements option should be either of source or sink ports. It cannot be instance.<br>The -model option matches the instance of source or sink ports for which an error is reported.   |
| -pa_checks=uml    | QPA_UPF_MISSING_LS_CHK        | -elements, -model, -sources, -sinks<br>The -elements option should be either of source or sink ports. It cannot be instance.<br>The -model option matches the instance of source or sink ports for which an error is reported.   |

**Table A-8. Supported Options — Dynamic Checks (cont.)**

| Dynamic Check  | Mnemonic              | Supported Option                         |
|----------------|-----------------------|--|
| -pa_checks=upc | QPA_UPF_PG_CHK        | -domains, -strategies                    |
| -pa_checks=ugc | QPA_GLITCH_ASSERT_ERR | -domains, -strategies,<br>-powerswitches |

## save\_checks\_config

Saves the configuration of static RTL and dynamic checks performed by the tool, to a specified file.

### Syntax

```
save_checks_config <file_name>
```

### 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 or -pa\_tclfile argument.

### Arguments

- <file\_name>  
Specifies the file name in which the configuration of static RTL and dynamic checks performed by the tool, is saved.

### Examples

```
save_checks_config config_file
```

## 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}
```

### 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_upf` or `-pa_tclfile` argument.

### Arguments

- `-domains {<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 {o | os | osw | sc | scb}`

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

### 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>]
```

### 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\_upf or -pa\_tclfile 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.

### Arguments

- -function <function\_list>  
A required list of one or more function names (you must specify at least one function name).
- -package <package\_name>  
(Optional) Detects only functions from the specified package, package\_name.

## set\_pgpin\_vct

Specifies the VCT to be used for pins of any particular pg\_type attribute.

### Syntax

```
set_pgpin_vct <pg_type> -vct <vct_name>
```

### 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\_upf or -pa\_tclfile argument.

### 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.

### 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> -power <power_net_name> -ground  
    <ground_net_name> -reset
```

### 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` or `-pa_tclfile` argument.

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>
```

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
```

See the “[vopt -warning](#)” command in the *Command Reference Manual*.

### Default Behavior

For top-level ports (in the absence of boundary information), it is left up to the tool 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 tool 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 tool 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. 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.

#### **Tool 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 tool 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 is 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.

## 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. 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. Specify this switch by itself or in combination with -power.
- -reset  
Not currently supported.



# Appendix B

## Model Construction for Power Aware Simulation

---

Power Aware verification uses Verilog behavioral models of power management 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>331</b> |
| <b>Named Events in Power Aware</b> ..... | <b>333</b> |
| <b>Attributes</b> .....                  | <b>334</b> |
| <b>Model Interface Ports</b> .....       | <b>335</b> |
| <b>Model Construction Examples</b> ..... | <b>337</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 are 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 management 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 power-up. 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 management 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 management 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 the following:

```
"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 `s` 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 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

## Supplemental Information

---

This appendix provides supplemental information on applications of Power Aware.

|  |            |
|--|------------|
| <b>Power Aware Verification of ARM-Based Designs .....</b> | <b>342</b> |
|--|------------|

## 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>342</b> |
| <b>Introduction</b> .....                   | <b>343</b> |
| <b>Active Power Management</b> .....        | <b>343</b> |
| <b>Power Management Techniques</b> .....    | <b>343</b> |
| <b>Power Management Specification</b> ..... | <b>344</b> |
| <b>Power Management Architecture</b> .....  | <b>345</b> |
| <b>Power Managed Behavior</b> .....         | <b>352</b> |
| <b>Power Control Logic</b> .....            | <b>352</b> |
| <b>Power Aware Verification Flow</b> .....  | <b>353</b> |
| <b>Summary</b> .....                        | <b>356</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 powerdown. 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

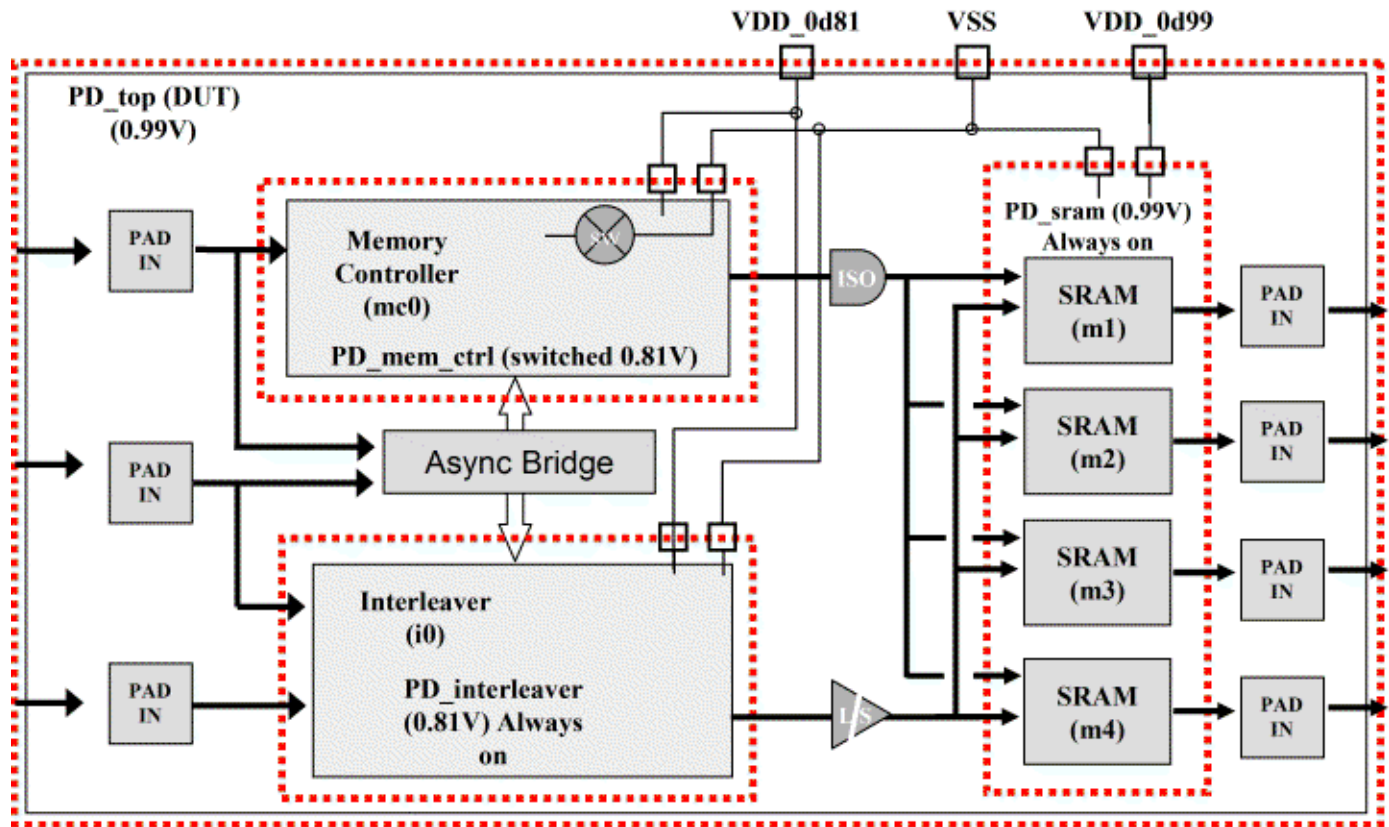
In order to employ active power management techniques such as power gating and multiple voltage supplies, the design must be partitioned into separate functional areas that can be independently powered. Additional logic must be inserted into the design to perform special functions such as power switching, 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 C-1](#) shows a block diagram of a design that has two operating modes: ON and SLEEP.

Figure C-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 shares 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 discontinuous regions of the chip.

The design in [Figure C-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 C-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 {i0}  
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 C-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 C-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    ON    OFF   }
add_pst_state OFF \
  -pst top_pst -state {    OFF       OFF   ON    OFF   }
```

The power states of the system in [Figure C-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 C-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 C-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 powerdown. 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 C-1](#).



```
#-----  
# Setup retention strategy for mem ctrl  
#-----  
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.

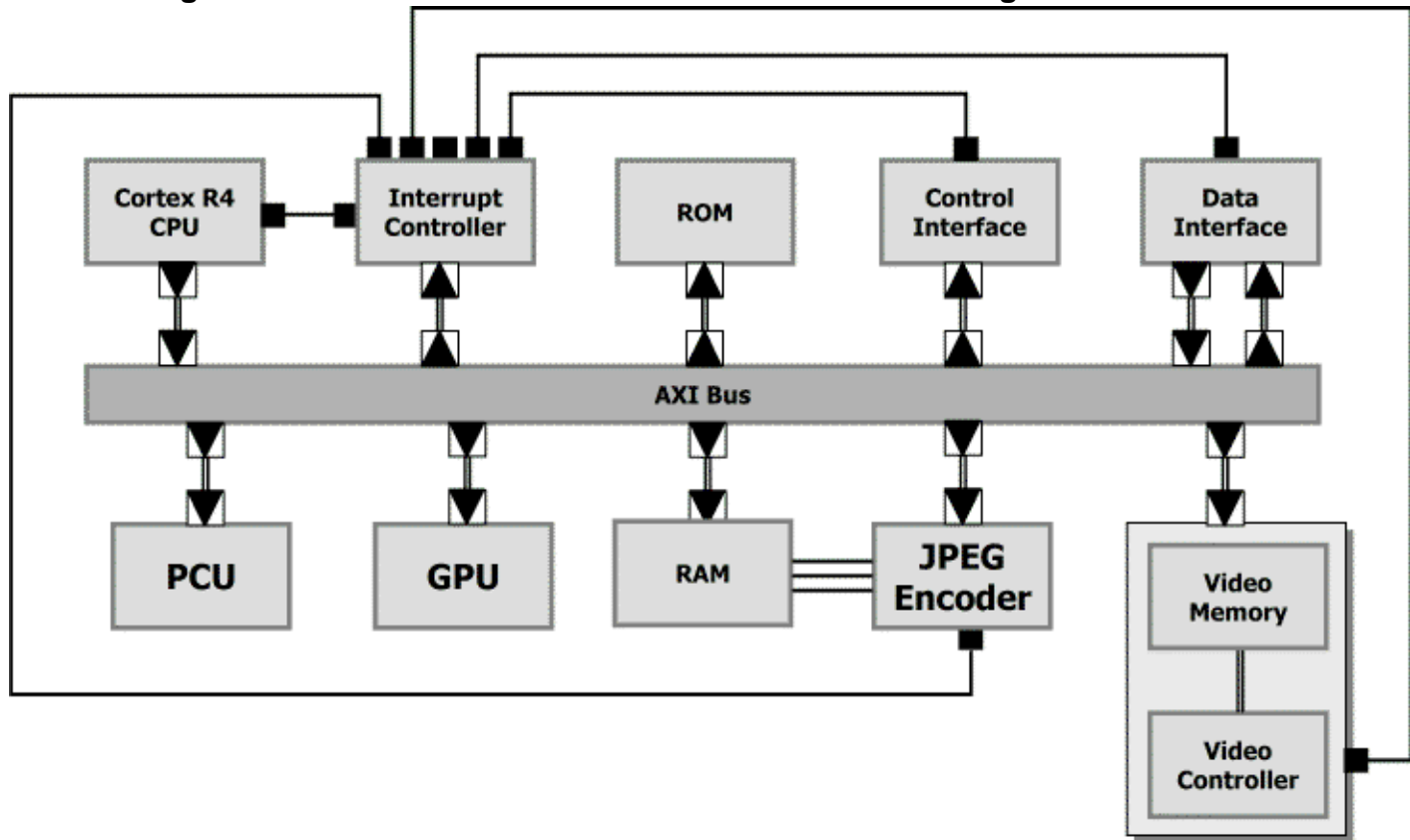
## Power Aware Verification Flow

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 C-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 C-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 interacts 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, if 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>.



## — D —

Design flow, [16](#)

## — H —

Hard macro, [200](#)

## — L —

Liberty libraries, [217](#)

## — M —

Macro, hard, [200](#)

## — N —

Named events, [333](#)

## — P —

Power State Table (PST), [52](#)

PST, [52](#)

## — U —

Unified Power Format (UPF), [235](#)

UPF, [235](#)

## — V —

Verilog

named events, [333](#)





# **End-User License Agreement with EDA Software Supplemental Terms**

Use of software (including any updates) and/or hardware is subject to the End-User License Agreement together with the Mentor Graphics EDA Software Supplement Terms. You can view and print a copy of this agreement at:

[mentor.com/eula](http://mentor.com/eula)

