

Verification Continuum™

# **VC LP**

## **User Guide**

---

Version U-2023.03-SP2, September 2023

**SYNOPSYS®**

# **Copyright Notice and Proprietary Information**

© 2023 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

## **Destination Control Statement**

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

## **Disclaimer**

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

## **Trademarks**

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <http://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

## **Free and Open-Source Software Licensing Notices**

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

## **Third-Party Links**

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

[www.synopsys.com](http://www.synopsys.com)

## **Synopsys Statement on Inclusivity and Diversity**

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.



# 1 Contents

Chapter 1Contents .....	3
Chapter 1Introduction .....	11
1.1 Verification Compiler Platform .....	11
1.2 VC Static and Formal Platform .....	11
1.3 VC LP .....	12
1.3.1 Key Features of VC LP .....	12
1.3.2 VC LP Low Power Checks Flow .....	12
Chapter 2 Getting Started With VC LP .....	17
2.1 Prerequisites .....	17
2.2 Setting Up VC LP Design Environment .....	18
2.2.1 Setting The VC_STATIC_HOME Environment Variable .....	18
2.2.2 Changing the VC Static Session Name and Location .....	19
2.2.3 Updating the vc_static_shell Setup File .....	20
2.2.4 Updating Application Variable Settings .....	21
2.2.5 Configuring Message Tags .....	21
2.2.6 Running Electrical Sign off Checks .....	24
Chapter 3 Reading and Building Design .....	27
3.1 Reading the Liberty Files .....	27
3.1.1 The search_path and link_library Variable .....	27
3.1.2 Tcl Commands to Append Missing Liberty Attributes .....	28
3.2 Reading the Design .....	29
3.2.1 Application Variables that Impact Reading a Design .....	32
3.2.2 Reading RTL Designs Examples .....	32
3.2.3 Reading Netlist Designs Examples .....	36
3.2.4 Support for Reporting Message Database .....	36
3.2.5 Migrating Designs from Other Synopsys Tools .....	38
3.2.6 Support for MULTI_PORT_MEM Operator .....	40
3.2.7 Handling DesignWare Components .....	41
3.2.8 Improving Performance of Design Reads .....	43
3.3 Checking Successfully Built Design .....	44

3.3.1 Checking Design Consistency and Completeness .....	44
3.3.2 Deviations Reported Using the report_cell_classification .....	47
3.3.3 Identifying Black boxes, Empty Modules and Modules Without Definitions .....	55
3.3.4 Obtaining a Report of the Design Read and UPF Messages .....	56
3.4 Specifying Conditional UPF Command .....	57
3.4.1 Use Model .....	59
3.5 Reading Power Intent .....	59
3.5.1 The read_upf Command .....	60
3.5.2 The remove_upf Command .....	61
3.5.3 The get_upf_scope Command .....	62
3.5.4 The report_violations -app UPF Command .....	63
3.5.5 Support Boolean Expressions in All UPF Commands .....	66
3.5.6 Support for Escape Characters .....	66
3.5.7 Support for analyze_domain_crossings Command .....	67
3.5.8 Support for Plato Parser .....	68
3.5.9 Support for Comparing UPF Data Models .....	71
3.5.10 Support for Design Independent UPF Checker Flow .....	75
3.5.11 Support for Smart Database Linking .....	79
3.6 Analyzing Compiler Messages .....	80
3.7 Running VC LP Checks .....	80
3.7.1 VC LP Database .....	82
3.7.2 The check_lp Command .....	94
3.7.3 Merging Reports of Multiple Parallel check_lp Runs .....	98
3.7.4 Reducing Compile Time in Subsequent check_lp Runs .....	103
3.8 Support for Library Checks .....	104
3.8.1 Use Model .....	104
3.8.2 Tags Impacted in this Flow .....	105
3.8.3 New Tags Introduced .....	105
3.9 Checking Reports .....	105
3.10 Saving and Restoring Sessions Using save_session and -restore .....	106
3.10.1 Saving a Session Automatically .....	107
3.10.2 Deleting Files After Saving a Session Automatically .....	108
3.10.3 Deleting Files After Saving a Session Manually .....	109
3.11 Saving and Restoring Sessions Using CR Technology .....	109
3.11.1 Use Model .....	109
3.11.2 New commands Introduced .....	109
3.11.3 New Error Messages Introduced .....	111
3.11.4 GUI support .....	111
3.12 Support for Auto-Saving and Auto-Restarting Sessions .....	111
3.13 Support for Comparing Results Of Two VC LP Sessions .....	112
3.13.1 Use Model .....	112
3.13.2 The dump_vcld_db Command .....	112
3.13.3 The compare_vcld_db Command .....	113
3.13.4 The lp_upf_diff_ignore_patterns Application Variable .....	113
3.13.5 Limitations Comparing Results Of Two VC LP Sessions .....	113
3.14 Handling Encryption .....	114
3.14.1 Naming Encrypted Modules .....	114
3.14.2 Tcl Query Command Support .....	117
3.14.3 LP Specific Checks .....	118
3.14.4 Encryption Support in the GUI .....	118

3.14.5 Design Dumping .....	118
3.14.6 Black Boxing .....	118
3.15 Extracting Test Cases Automatically .....	118
3.15.1 Limitations .....	120
3.16 In-design integration of ICC2 and VC LP .....	120
3.16.1 Use Model .....	120
<b>Chapter 4 Debugging VC LP Reports .....</b>	<b>123</b>
4.1 Understanding VC LP Violation Database .....	123
4.1.1 Message IDs and Tags .....	123
4.1.2 Compressing Violations .....	127
4.2 Getting the List of Prevented Tags .....	134
4.3 Smart Grouping of Violations .....	135
4.3.1 Scenarios of Smart Grouping Violations .....	136
4.3.2 Use Model .....	138
4.3.3 Supported Application Variables .....	139
4.3.4 Violations Introduced for This Support .....	139
4.3.5 Viewing the RCA Results Dashboard .....	141
4.3.6 Viewing Smart Grouping of Violations in GUI .....	142
4.3.7 Tag Coverage Supported .....	143
4.3.8 Limitations on The Smart Grouping of Violation Feature .....	145
4.4 Debugging using Tcl .....	145
4.4.1 Reporting Violations with report_violations -app LP .....	145
4.4.2 Viewing User-defined Object Names in Violation Reports .....	151
4.4.3 Filtering Messages .....	154
4.4.4 Operations on Tag Definitions .....	155
4.4.5 Custom Checks .....	157
4.4.6 Waiving Messages .....	164
4.5 Tcl Query Commands .....	173
4.5.1 Design Query Commands .....	173
4.5.2 Low Power Query Commands .....	176
4.5.3 Support to Query SPA/SDA Attributes .....	213
4.6 Debugging Using GUI .....	214
4.6.1 Handling Memory in Netlist flow .....	215
4.6.2 Using Verdi for Debug .....	216
4.6.3 Support for NLDM Based nSchema Flow .....	225
4.6.4 Using Verdi Power Commands .....	230
4.6.5 Opening Multiple nSchema Windows .....	231
4.6.6 Creating Multiple Filters .....	233
4.6.7 Using the Locator Function in Verdi Schematic Viewer .....	234
4.6.8 Colorize by Root Supply .....	242
4.6.9 Support for Consistent Name Resolution in VC Static and Verdi .....	245
4.6.10 Using VC Static Native GUI .....	250
4.6.11 Using VC Static Console .....	262
<b>Chapter 5 Handling Special LP Scenarios .....</b>	<b>269</b>
5.1 Special VC LP features .....	269
5.1.1 Signal Corruption Checks .....	269
5.1.2 Bias Checks .....	281
5.1.3 Support for Bias Checks for Flipwell Cells .....	291

5.1.4 Support for Self Bias Checks .....	293
5.1.5 Analog Checks .....	294
5.1.6 SCMR Checks .....	299
5.1.7 Advanced Rail Order Checks .....	302
5.1.8 Isolation Control Port Cloning .....	305
5.1.9 Unconnected Net Handling .....	307
5.1.10 Enhanced Hard Macro Handling .....	318
5.1.11 Power Switch Checks .....	324
5.1.12 Constant Aware Checks .....	336
5.1.13 Predictive Checks .....	339
5.1.14 Support for Predictive Checks Related to map_isolation_cell/map_level_shifter_cell Commands .....	349
5.1.15 Support for Architectural Checks in Predictive Flow .....	353
5.1.16 Gray Cloud FUNC Checks in Predictive Flow .....	355
5.1.17 Marking Design Ports as Supply Ports .....	356
5.1.18 Support for Electrostatic Discharge Checks .....	357
5.1.19 The PG_DIODE_EXCESS and the PG_PASSTRAIN_EXCESS Violation .....	361
5.1.20 Device Connectivity Checks .....	363
5.1.21 Support for Controlling Signal Connectivity Checks .....	366
5.1.22 Support for Multiple Power Domains in a Single Voltage Area .....	367
5.1.23 Support for Voltage reconciliation .....	374
5.1.24 Support for Adding Test Circuits in PSW Chains .....	380
5.1.25 Support for RAM Enabled Connectivity Checks .....	383
5.1.26 Support for Synopsys Safety Format (SSF) Checks in VC LP .....	389
5.2 Advanced Low Power Cells .....	391
5.2.1 NOR-style Isolation Cells .....	391
5.2.2 NOR-style ELS Cells .....	398
5.2.3 Support for NOR Style Isolation on Macro Outputs .....	400
5.2.4 Differential LS .....	402
5.2.5 MBIT Retention Cells .....	408
5.2.6 Support for Multi-bit Protection (ISO/LS/ELS) Devices .....	409
5.2.7 Zero Pin Retention Cells .....	409
5.2.8 Diodes .....	412
5.2.9 Block/IP Level ISO Signals Connectivity Checks .....	413
5.2.10 Support for Local Bias For Isolation and Retention Cells .....	418
5.2.11 Support for PAD Cells .....	421
5.2.12 Support for ISO/ELS Cells with No Enable Pin .....	425
5.2.13 Support For Fine Grain Power Switches .....	426
5.2.14 Support for Isolation Clamp Checks .....	428
5.2.15 Support for AON Backplane Isolation Cells .....	430
5.2.16 Support for ISO_DATA_RESET and ISO_DATA_UNDETERMINABLE .....	430
5.2.17 Support for ISO Latch With async set/reset .....	431
5.2.18 Support for Physical Variant Cells .....	435
5.3 Advanced UPF Variations .....	435
5.3.1 Support for Restricting UPF Commands in DIUC .....	435
5.3.2 Support for upf_set_once Command .....	437
5.3.3 Support for Checking UPF Versions .....	438
5.3.4 Support for Allowing Specific enum Values in UPF Commands .....	439
5.3.5 Support for lp_allow_default_version Application Variable .....	440
5.3.6 Support for get_upf_scope Command .....	441

5.3.7 Precedence for Choosing ISO Strategies .....	442
5.3.8 Retention Strategy Precedence .....	446
5.3.9 Support for reference_only Attribute .....	447
5.3.10 Support for lower_domain_boundary Attribute .....	448
5.3.11 VC LP Simplifies PST Using Voltage Triplets .....	450
5.3.12 Support for Logic Nets in -elements option of set_design_attributes command .....	452
5.3.13 Support for the terminal_boundary Attribute .....	452
5.3.14 Support for create_power_domain -exclude_elements .....	456
5.3.15 Support for soft_macro Attributes .....	456
5.3.16 Support for Leaf cells as Elements in the create_power_domain Command .....	458
5.3.17 Support for Leaf Cell Pins in the -control_port Switch of Power Switch Strategy .....	459
5.3.18 Support for -force_shift Option in the set_level_shifter Command .....	459
5.3.19 Support for voltage_map attribute for Level Shifter Checks .....	462
5.3.20 Support for -name_suffix/name_prefix in the set_isolation/set_level_shifter Commands .....	464
5.3.21 Detecting PST Equivalent Source and Sink Crossover .....	467
5.3.22 Support for input_signal_level for Level Shifting Checks .....	472
5.3.23 Support for iso_source and iso_sink Attribute in set_port_attributes .....	474
5.3.24 Support for set_port_attributes with -applies_to and without -elements .....	475
5.3.25 Support for set_port_attributes -is_analog Attribute .....	478
5.3.26 Support for Name Based ISO Strategy Association .....	479
5.3.27 Support for Name Based Association for ELS Cells .....	480
5.3.28 Support for resolved_iso_strategy in set_port_attribute .....	480
5.3.29 Support for -feedthrough and unconnected Attributes in set_port_attributes .....	483
5.3.30 Support for Feedthrough/Unconnected SPA Attribute Block Level Consistency Check .....	487
5.3.31 UPF 2.0/2.1 Enhancements in Commands .....	488
5.3.32 Support for set_retention/set_isolation UPF 3.0 Versions .....	490
5.3.33 Support for -update option for -source and -sink option in set_isolation command .....	491
5.3.34 Support for clamp_value in set_port_attributes Command .....	491
5.3.35 Support for -exclude_elements/-exclude_ports in set_port_attributes .....	493
5.3.36 Support for -literal_supply in set_port_attributes .....	494
5.3.37 Support for set_port_attributes -attribute is_pad .....	498
5.3.38 Enhancements to set_port_attributes on inout ports .....	499
5.3.39 Support for set_port_attributes -attribute antenna_diode_* .....	500
5.3.40 Support for set_retention_elements UPF Command .....	501
5.3.41 Support for set_retention -use_retention_as_primary UPF Command .....	505
5.3.42 Support for exclude_elements with set_retention Command .....	505
5.3.43 Support for -supply_set in create_power_switch UPF Command .....	505
5.3.44 Support for add_power_state .....	509
5.3.45 Enhancement to State Propagation of add_power_state .....	523
5.3.46 Support for add_supply_state Command .....	530
5.3.47 Support for set_equivalent Command .....	531
5.3.48 Support for Power Down Function .....	535
5.3.49 Support for -pg_function Attribute .....	538
5.3.50 Support for create_power_domain -available_supplies .....	541
5.3.51 Support for create_power_domain -elements {} .....	545
5.3.52 Support for -resolve Function in create_supply_net .....	545
5.3.53 Support for enable_bias Design Attribute .....	547
5.3.54 Support for -elements Option in the set_design_attribute -attribute correlated_supply_group Command .....	547
5.3.55 Support for ignore_voltage_difference User Defined Attribute .....	548

5.3.56 Support for sim_corruption_control Command .....	548
5.3.57 Support for sim_assertion_control Command .....	549
5.3.58 Support for sim_replay_control Command .....	551
5.3.59 Support for Compatibility with Other Synopsys Implementation Tools .....	551
5.3.60 Support for Syntax and Semantics Checks for VCT commands .....	553
5.3.61 Support for Name Space Conflict between UPF and Design Logic Net/Port .....	556
5.3.62 Support for is_on_clock_path Attribute .....	556
5.3.63 Support for describe_state_transition and add_state_transition UPF Command .....	557
5.3.64 Support for Relative Power Ground Supply .....	557
5.3.65 Support for set_repeater -source / -sink .....	560
5.3.66 Support for relax_constant_corruption_for_macro_inputs_and_primary_outputs Design At-tribute .....	562
5.3.67 Support for set_level_shifter input_supply and output_supply .....	564
5.3.68 Support for -location other in LS strategy in UPF 3.0 .....	566
5.3.69 Support to Compare Power and Ground voltages of Supply Pairs .....	569
5.3.70 Support for Checks on Simstate .....	572
5.3.71 Support for Duplicate Port State Name with Different Voltages .....	573
5.3.72 Support for get_resolved_elements Command .....	574
5.3.73 Support for Treating HighConn of Macro as Domain Boundary .....	574
5.3.74 Support for SPA -driver/-receiver and SRSN on Hard Macro .....	576
5.3.75 Support for Defining Isolation Strategy .....	578
5.3.76 Support for isolation_enable_condition_allow_pg_pin Attribute .....	580
5.3.77 Support for Automatic Supply Net Generation from create_supply_port .....	581
5.3.78 Support for ISO_SIGSUP_MISSING Violation .....	581
5.3.79 Support for Defining and Applying Power Models .....	582
5.3.80 Support for Automatic Supply Net Generation from create_supply_port .....	584
5.3.81 Support for APS_LOGICEXP_EQUIVALENT Violation .....	584
5.3.82 Support for isolation_sink_power_pin and isolation_data_power_pinLiberty Attributes .....	585
5.3.83 Support for coupled_supplies States Check .....	586
5.3.84 Support for upf_cmd_wrapper_prefix Application Variable .....	587
5.3.85 Support for Atomic Power Domains .....	588
5.3.86 Support for ISO_STRATEGY_SELF Violation .....	588
5.3.87 Support for LS_COMBO_FUNC Violation .....	590
5.3.88 Supports for Full SoC Validation of LS Source - Sink Strategies .....	591
5.3.89 Support for is_virtual Design Attributes .....	591
Chapter 6 Using VC LP in Various Design and Verification Flows .....	593
6.1 Hierarchical Verification Flows .....	593
6.1.1 Black Box or Stub Flows .....	593
6.1.2 ETM Based UPF Hierarchical Flow .....	612
6.1.3 Common Parser Level Limitations for Black box and ETM flow .....	621
6.2 Golden UPF Flow .....	621
6.2.1 Supplement UPF File .....	623
6.2.2 Name Mapping File .....	624
6.2.3 Use Models .....	624
6.2.4 Limitations .....	625
6.3 Support for Signoff Abstraction Model Based Hierarchical Flow .....	625
6.3.1 Prerequisites .....	626
6.3.2 Use Model .....	627
6.3.3 New SAM Abstraction (Lightweight SAM) Model .....	630

6.3.4 Enhancements to Stub and Black box Modules .....	630
6.3.5 Macro Cells with Internal Isolation .....	631
6.3.6 PSW Port Punching .....	631
6.3.7 Debug diagnostics .....	631
6.3.8 Parser Messages, Tcl Commands and Application Variables Introduced .....	632
6.3.9 Support to Retain Control Sources Driving Primary Outputs of a SAM Boundary .....	638
6.3.10 Support for Comparing Architectural Checks .....	638
6.3.11 Support for SAM with NPS Flow .....	640
6.3.12 Support for Single Run SAM Flow .....	641
6.3.13 Support for OnTheFly SAM .....	642



# 1 Introduction

This chapter provides an introduction to Verification Compiler Platform, VC Static Platform and VC LP. The chapter is organized into the following sections:

- ❖ “Verification Compiler Platform”
- ❖ “VC Static and Formal Platform”
- ❖ “VC LP”

## 1.1 Verification Compiler Platform

Today's electronic consumer market is driven by a huge demand for mobility, portability, and reliability. Additional functionality, performance, and bandwidth are very important for maximizing semiconductor sales in addition to faster time-to-market and product quality. The evolution of applications, such as cellular phones, laptops, PDAs, computers, mobile multimedia devices, and portable systems, has seen an exponential growth in battery operated systems.

The increase in design complexities and shrinking technologies, where more and more functionality is being added into smaller area of a chip has brought in a new set of challenges in System-on-Chip (SoC) verification. With adoption of advanced techniques and sophisticated tools, which helps in verifying SoC connectivity, signal integrity, power management, and functionality of analog components, hardware-software co-verification has become inevitable.

This brings in a need for a unified and integrated verification environment with seamless flow and reuse of the information across different domains/levels to achieve faster results.

Verification Compiler Platform is a next-generation verification solution that provides a scalable environment, where sophisticated tools work seamlessly with each other throughout the flow to accomplish various verification tasks using integration of technologies. It helps in optimizing design iterations and recompilations, shortens debug cycles, and enables steady integration and interoperability between individual verification tools.

## 1.2 VC Static and Formal Platform

Traditionally, simulation-based dynamic verification techniques have been the mainstay of functional verification. As modern day SoC designs become more complex, the adoption of static verification techniques is important.

Synopsys' VC Static and Formal offers the next-generation comprehensive VC formal verification solution, and VC LP verification solution.

Synopsys' VC Static and Formal verification solution combines the best-in-class technologies for improved ease-of-use, accuracy, and performance. It also provides with low violation noise and excellent debug capabilities. This solution enables designers and verification engineers to quickly and easily find and fix

bugs in RTL before simulation; thus reducing the time needed before software bring-up, hardware emulation, and prototyping.

VC Formal verification offers property checking that consists of mathematical techniques to test properties or assertions to ensure the correct functionality of RTL designs. RTL is further verified for functionality and policy compliance. Model checking technique exhaustively and automatically checks whether a model adheres to a given specification and verifies correct properties of finite-state systems. For more information, see the *VC Formal Verification User Guide*.

## 1.3 VC LP

VC LP is a multi-voltage, static low power rule checker that allows engineers to rapidly verify designs that use voltage control based techniques for power management. VC LP is part of the Synopsys Eclypse Flow.

VC LP also helps in pipe-cleaning the power intent of the design that is captured in IEEE 1801 Unified Power Format (UPF) before such intent is used as a golden reference for implementation and other verification tools. Further, VC LP verifies the implemented power-intent later in the design flow.

VC LP is integrated with Verdi to provide designers and verification engineers access to the combined power of low power specific debug features and use Verdi's de facto industry-standard workflow, interface and powerful debug capabilities.

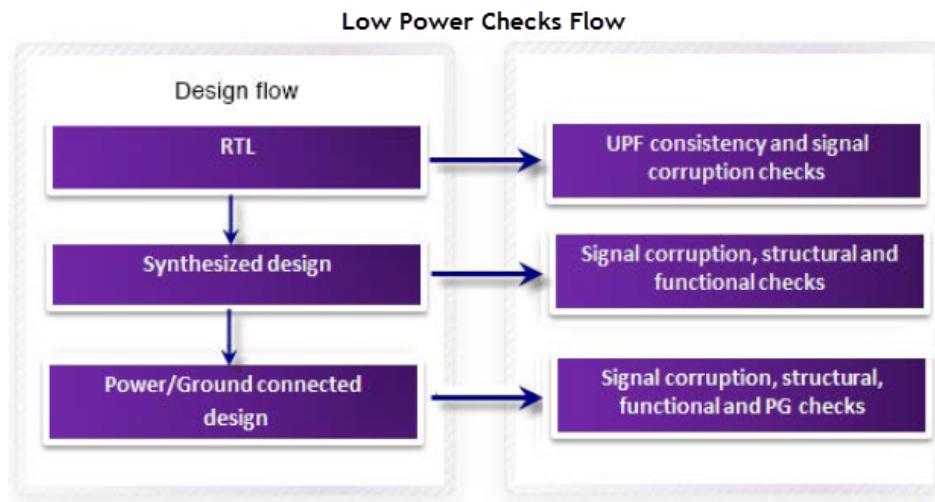
### 1.3.1 Key Features of VC LP

The key features and benefits of using VC LP for low power static verification in a typical design flow are as follows:

- ❖ Power Intent Consistency Checks
  - Performs syntax and semantic checks on the UPF that help validate the consistency of the UPF before starting with the implementation.
- ❖ Signal Corruption Checks
  - Detects the violating power architecture at the gate-level netlist.
- ❖ Structural Checks
  - Validates insertion and connection of special cells used in low power design such as isolation cells, power switches, level shifters, retention registers, and always-on cells through out the implementation flow.
- ❖ Power and Ground (PG) Checks
  - Check the PG consistency against the UPF specification for power network routing on physical netlists.
- ❖ Functional Checks
  - Validates the correct functionality of isolation cells and power switches.

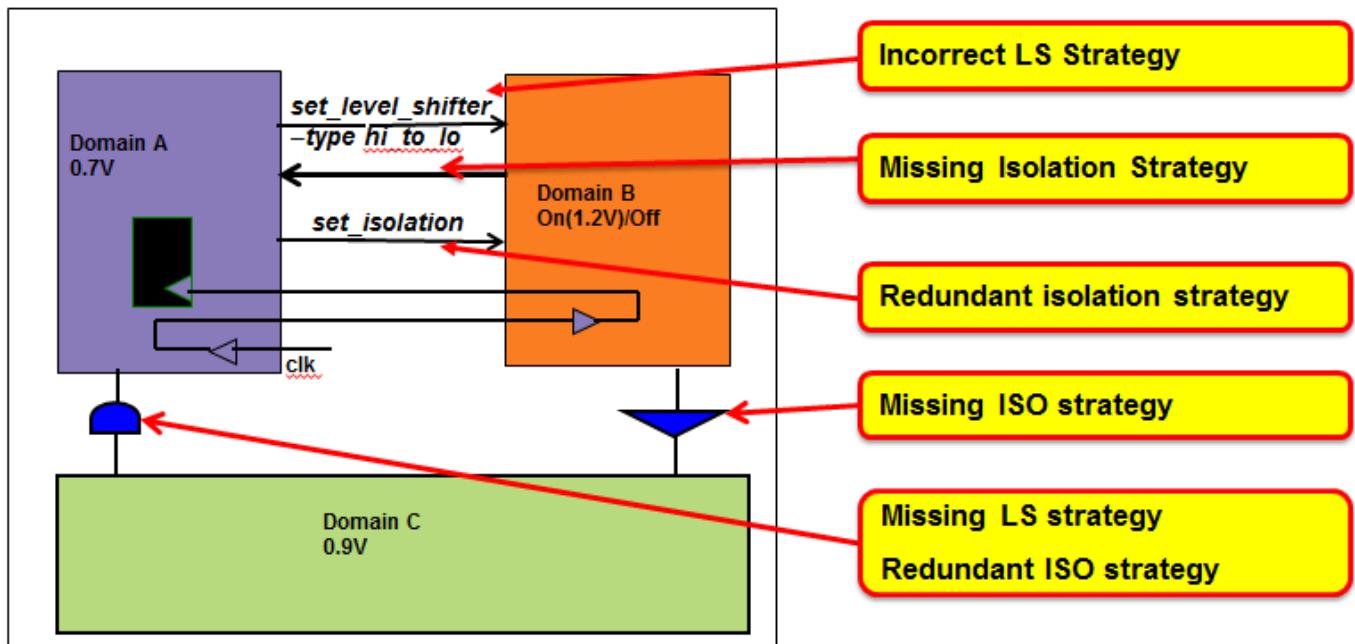
### 1.3.2 VC LP Low Power Checks Flow

VC LP UPF and functional checks provide a comprehensive checking methodology that enable you to identify low power related issues much earlier in the design cycle and through out the implementation flow.

**Figure 1-1 VC LP Low Power Checks Flow**

- ❖ At RTL-level, the VC LP UPF checks help in identifying power intent issues early in the design life-cycle and enable you to arrive at a clean UPF before starting the design flow. The VC LP UPF checks ensure that the UPF is complete and the design conforms to all the isolation and level shifter rules for all power-modes.

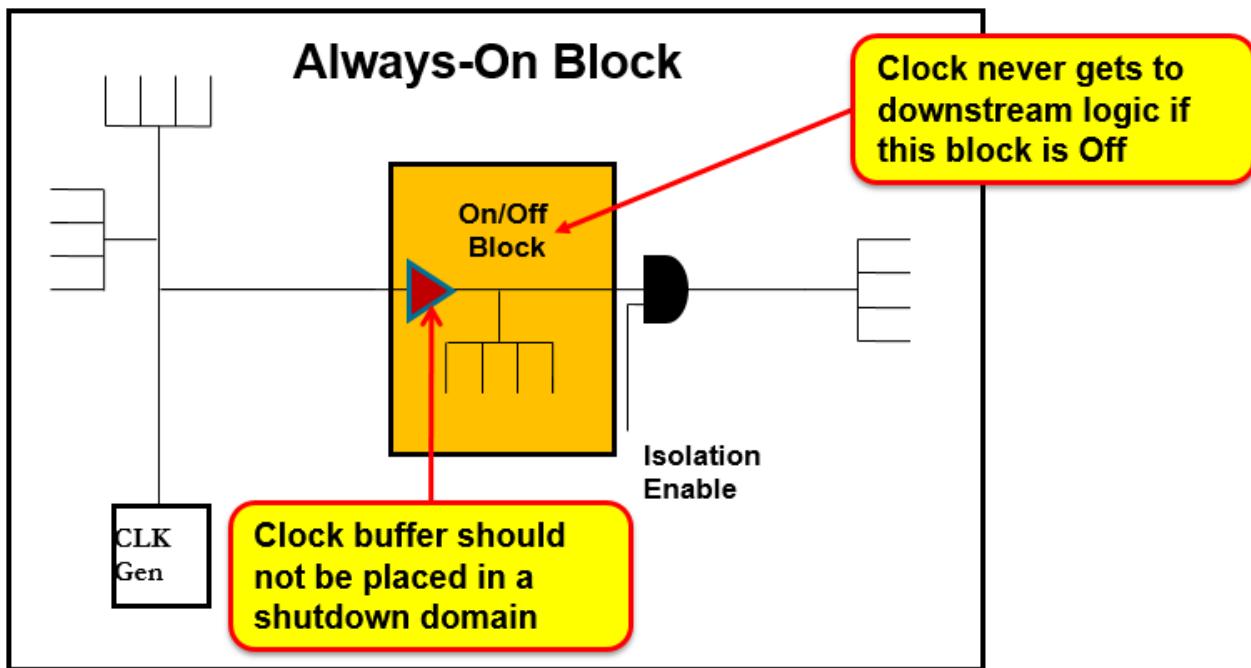
For example, the VC LP UPF checks help in power intent validation by identifying missing or redundant ISO/LS policies as illustrated in [Figure 1-2](#).

**Figure 1-2 RTL Level Low Power Checks**

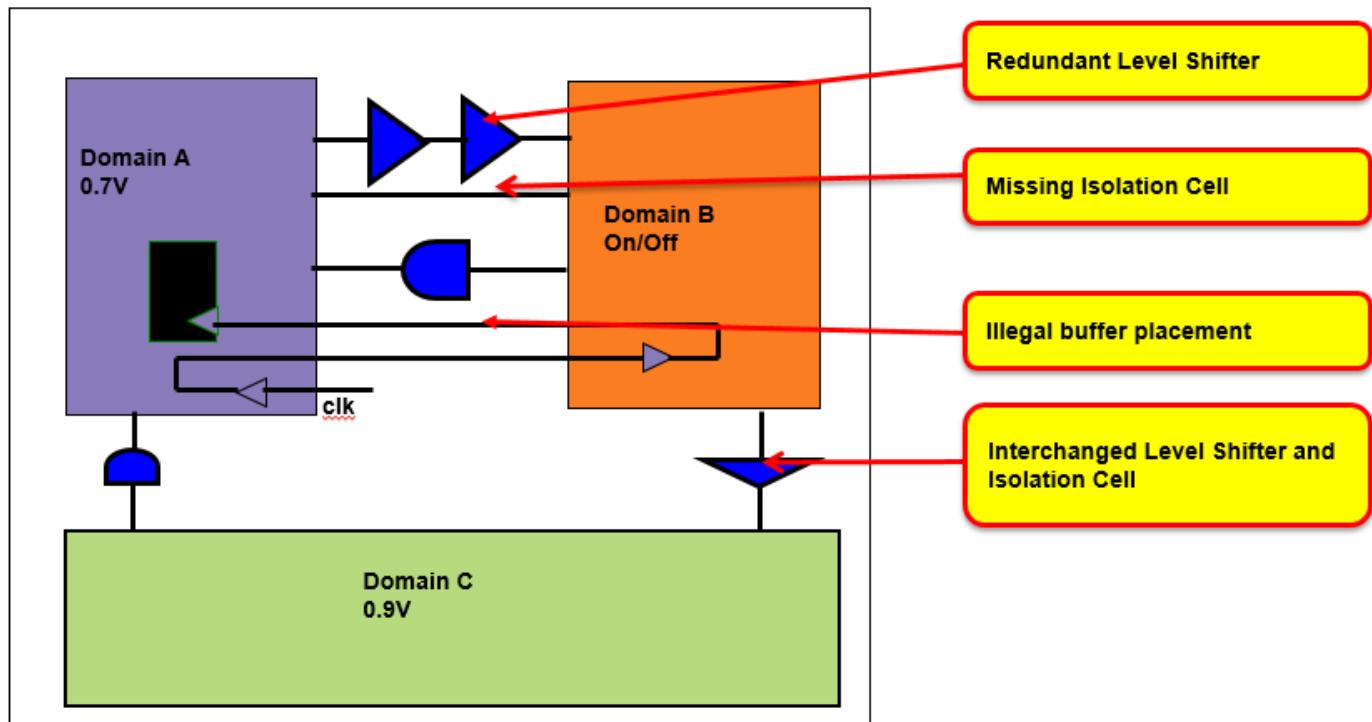
- ❖ At Netlist level, the VC LP UPF and functional (architectural) checks ensure that the netlist is consistent with UPF in structure and function. The UPF checks ensure the design instances are consistent with UPF. The architectural checks ensure that the implemented design is functionally

correct. There might be cases where the design is structurally correct but functionally incorrect. The VC LP architectural checks identify these low power functional issues even though the implemented design might be structurally correct. The VC LP UPF and Structural checks also help in identifying implementation issues by verifying if the low power cells (ISO/LS/RET) inserted in the design is consistent with the UPF and the library. The other VC LP functional checks include Analog checks, Inout checks, Bias checks, Diode checks and so on.

Figure 1-3 Netlist Level Low Power Checks



- ❖ The VC LP Power Ground checks help validate the power network implementation by verifying if the Power/Ground pin connectivity in the post-layout design is consistent with UPF and cell library.

**Figure 1-4 LP Checks on Power/Ground Connected Design**



# 2 Getting Started With VC LP

This section describes how you can get started with VC LP. This section assumes you have a licensed copy of the software and have it installed on your system.

This chapter is organized into the following sections:

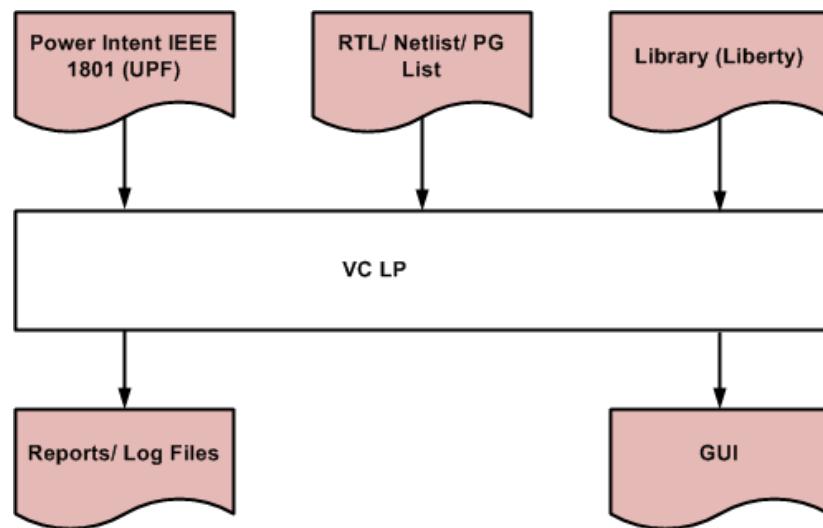
- ❖ “[Prerequisites](#)”
- ❖ “[Setting Up VC LP Design Environment](#)”

## 2.1 Prerequisites

VC LP takes in an RTL (Verilog, VHDL, SVD), netlist (Verilog) or post-layout netlist of the design. It reads the Liberty DB file for resolving, elaborating the design, recognizing special cells and annotating power connections. It accepts the power intent specified in the UPF.

As an output, VC LP creates a log file, an error and warnings report for all violations related to low power static rule checks. VC LP provides a Tcl infrastructure that helps in debugging these violations. You can also use the VC LP GUI to debug your design violations.

**Figure 2-1 VC LP Checks**



## UPF Requirements

VC LP supports the industry-standard UPF IEEE 1801 subset that is used to capture low power design requirements. UPF is a standard set of Tcl commands used to specify the power intent of the design. Using UPF commands, you can specify the supply network, switches, isolation, retention, and other aspects pertinent to the power management of the design. This single set of commands can be used for verification, analysis, and implementation of the design.

For more details on the list of supported UPF commands, see section “[Supported LP Checks](#)”.

## Design Requirements

VC LP supports RTL (Verilog, VHDL, MX, SVD) and PG netlists (Verilog).

## Liberty Requirements

VC LP requires industry standard liberty files (compiled .db files) for gate level netlists (with or without PG routing). Low power cells in the design need to have specific library attributes as recommended for Synopsys Eclipsy Flow.

## 2.2 Setting Up VC LP Design Environment

### 2.2.1 Setting The VC\_STATIC\_HOME Environment Variable

VC LP uses the pivotal environment variable: VC\_STATIC\_HOME. This variable must be set to point to the installation directory as shown in the following code snippet. In the installation directory, you can find the bin, lib, doc and other directories.

```
% setenv VC_STATIC_HOME /tools/synopsys/vcst
```

Optionally, you can add \$VC\_STATIC\_HOME/bin to your \$PATH. To start the VC Static tool, execute the following command:

```
% $VC_STATIC_HOME/bin/vc_static_shell
```

This command starts a VC Static shell session and you see the following prompt:

```
%vc_static_shell>
```

#### 2.2.1.1 VC Static Shell Command Line Options

The following command line options are available for VC Static. The options may be abbreviated by leaving out the text in parenthesis; for example, either -f or -file can be used to give the name of a script file to execute.

### Syntax

```
vc_static_shell -help
Usage: ~/monet/Release//bin/vc_static_shell
      [-batch]                      Start tool in batch mode (non-interactive).
      [-out_dir <dir_name>]          Name of output directory.
      [-cmd_log_file <log_name>]    Name of command log file in current directory.
      [-f(file) <file_name>]         Script file to exec after setup.
      [-gui]                         Start the GUI ActivityView.
      [-h(elp)]                      Print this help message.
      [-id | -ID]                   Give more information about application build/env.
      [-lic_wait <minutes>]          Wait for license for #minutes.
      [-mode64 | -full64]            Start tool in 64bit mode.
      [-mode32 ]                     Start tool in 32bit mode.
      [-no_init]                     Don't load .synopsys_vcst.setup files.
      [-no_restore]                 Remove previous session and start a new one.
      [-no_ui]                       Starts the tool without the GUI/UI process.
```

```

[-output_log_file <log_name>] Capture console output in given log file.
[-read_only] Restore a previous session in read-only mode.
[-remoteConfig <conf_file>] Remote config file for running VCS compile process
on bsub/qsub.
[-reset] Clean up the old data of the work session.
[-restore] Restore a previous session.
[-session <session_name>] Use the <session_name> directory for the runtime
database.
[-unbuffered] Output all screen messages in unbuffered mode.
[-use_ipv4] Use IPv4 protocol for inter process communications.
[-use_ipv6] Use IPv6 protocol for inter process communications.
[-no_color] Start tool in non-color mode.
[-x] Execute configuration TCL command.
[-y] Execute post configuration TCL command.
[-scl_span] Enable grabbing licenses across multiple servers.
[-container] Start tool within a container environment.
[-lic_report] Enable reporting of license check-in check-out
information.
[-noecho] Disable reporting commands in the screen output and
session log.
[-restart <session_name>] Session name required for restarting from
checkpoint.
[-fml_auto_restore_level <fml_auto_restore_level>] Restore
level of auto saved formal session.
[-rma_off] Turn off RMA if ultra or elite license is provided.
[-prompt <prompt_name>] Name of prompt.
debug options:
[-echo] Echo the environment but do not invoke the
executable.

```

## Use Model

### Using vc\_static\_shell in batch mode

```
%vc_static_shell -f vcst.tcl -batch -o vcst_screen.log
```

### Using vc\_static\_shell in interactive mode

```
%vc_static_shell -f vcst.tcl -o vcst_screen.log
```



#### Note

When you use the -batch option, VC LP automatically quits the shell even when quit is not explicitly specified in the vcst.tcl file or when an unexpected error occurs and the full run is not complete. This is useful for regression runs.



#### Note

Set the `stop_run_on_license_failure` application variable to true to ensure that if any license failure occurs during the run, VC LP does not exit from the shell. However, the execution of the current Tcl script stops. By default, the variable is set to false.

## 2.2.2 Changing the VC Static Session Name and Location

Once `vc_static_shell` is run in any user work directory, VC Static creates a default session (work database directory) in the current working area called `vcst_rtdb` [VC Static Run Time Data Base] along with default log files. The default session name is `vcst`.

You can change the name of the session and the location of the session at the time of invoking `vc_static_shell`.

```
%vc_static_shell -session my_path/my_session <other commands>
```

This command creates a database directory `my_session_rtdb` inside the directory `/my_path`

In addition, if you want to fork a new session from the existing session and save the new session, use the `save_session` command.

```
%vc_static_shell> save_session my_forked_session
```

However, forking (taking a snapshot) a new session is allowed only after design is loaded successfully into the current session.

### 2.2.3 Updating the `vc_static_shell` Setup File

VC LP uses the `.synopsys_vcst.setup` file to configure its environment for the design files for the VC LP run. You can create your own custom setup file with all the required settings (application variables, settings, custom scripts) that must be loaded each time you invoke the `vc_static_shell`. The key components of the `.synopsys_vcst.setup` file are the name mappings in the design libraries and the variable assignments.

When you invoke VC Static, VC LP looks for the `.synopsys_vcst.setup` files in the following three directories in the following order:

#### 1. Master Setup Directory

The `.synopsys_vcst.setup` file in the `$VC_STATIC_HOME/bin` directory contains default settings for the entire installation. If this file exists, VC LP reads it first.

#### 2. User Home Directory

VC LP reads the setup file in your home directory second, if present. The settings in this file take precedence over the conflicting settings in the `.synopsys_vcst.setup` file in the master setup directory, and carry over the rest.

#### 3. User Run Directory

VC LP reads the setup file in your design directory last. The settings in this file take precedence over the conflicting settings in the `.synopsys_vcst.setup` file in the master setup directory, and the `.synopsys_vcst.setup` file in your home directory, and will carry over the rest. You can use this file to customize the environment for a particular design.

**Table 2-1 The `.synopsys_vcst.setup` Files**

File	Location	Function
<code>.synopsys_vcst.setup</code>	Master setup directory ( <code>\$VC_STATIC_HOME/bin</code> )	Contains general VC LP setup information.
<code>.synopsys_vcst.setup</code>	User home directory	Contains your preferences for the VC LP working environment.
<code>.synopsys_vcst.setup</code>	User run directory	Contains design-specific VC LP setup information.

**Note**

If you want to prevent VC LP from reading setup files, you can use the `-no_init` command line switch.

## 2.2.4 Updating Application Variable Settings

VC LP offers a list of application variables that can be used as per your requirements. To see the list of all the available application variables and their current settings in the `vc_static_shell`, use the following command:

```
%vc_static_shell> printvar (or)
%vc_static_shell> report_app_var
```

The `printvar` command reports all variables including user defined variables, while the `report_app_var` command reports only the VC Static and Formal application variable settings.

### Example 1

```
%vc_static_shell> printvar
allow_upf_lrm_extension = "false"
analyze_skip_translate_body = "true"
annotate_iso_in_corr_control_state = "false"
...
...
...
....
```

### Example 2

Variable	Value	Type	Default	Constraints
vdd_type_diode_cell_name		string		
vdd_vss_type_diode_cell_name		string		
vss_type_diode_cell_name		string		

### Example 3

Variable	Value	Type	Default	Constraints
upf_buffer_list		string		# list of user specified buffers

For more information on the available application variables and their functions, the man pages of the application variables.

You can change the default behavior of VC LP by changing the default settings of the application variables. You can use the `set_app_var` command to change the setting of an application variable. The following example shows how to set a variable to true

```
%vc_static_shell>set_app_var lp_ack_port_from_strategy true
```

## 2.2.5 Configuring Message Tags

VC LP provides a large number of low power checks. All these checks have a message tags and a predefined reporting format. Based on collective user feedback, by default, VC LP has certain checks enabled and certain checks disabled. Also, the severity is predefined for each tag of a violation.

VC LP provides the flexibility for you to pick and choose which checks are relevant for your design. You can change the default enable/disable status of check and also change the severity of a check using the `configure_tag -app LP` command. In summary, you must configure the VC LP tags in the following cases:

- ❖ When you want to permanently skip certain tags without the local administrative overhead of a waiver.
- ❖ When you want to change the severity of some messages between error and warning.

It is recommended that you use the `configure_tag -app LP` command before reading a design or UPF.

### Note

Configuring the low power checks for a given design/run is one of the most important review that must be done by the design engineer. Disabling an LP check/tag using the `configure_tag -app LP` command does not necessarily improve the VC LP run time.

### Syntax

```
%vc_static_shell> configure_tag -app LP
Usage: configure_tag -app LP # Displays and configures low power violation tags
      [-tag <tag list>]      (Defines the tag(s) operated on)
      [-stage <stage list>]    (Apply tag alternations to the entire group(s):
                                Values: all, design, pg, upf)
      [-enable]                (Enables the tag(s))
      [-disable]               (Disables the tag(s))
      [-severity <error|warning|info>]
                                (Sets the tag(s) severity level:
                                Values: all, error, info, warning)
      [-clear]                 (Restores all tags to their original state)
      [-tcl]                   (Displays changes to the low power tag set in a TCL format
suitable for replay)
      [-regexp]                (Allows regexp expressions in the tag list (default glob-
style))
      [-all]                   (Displays all messages, even with default enable and
severity status)
      [-verbose]                (Displays additional details for each message)
      [-goal <goal name>]     (Defer effect until goal is checked)
```

### Use Models

To enable all tags in VC LP, use the following command:

```
%vc_static_shell> configure_tag -app LP -tag * -enable
```

### Other possible use models

```
%vc_static_shell> configure_tag -app LP -tag/-stage <tag/stage list> -disable/-enable
%vc_static_shell> configure_tag -app LP -tag/-stage <tag/stage list> -severity
<error|warning|info>
%vc_static_shell> configure_tag -app LP -regexp -tag <tag list> -disable/-enable
%vc_static_shell> configure_tag -app LP -regexp -tag <tag list> -severity
<error|warning|info>
%vc_static_shell> configure_tag -app LP -clear
%vc_static_shell> configure_tag -app LP -tcl
%vc_static_shell> configure_tag -app LP
%vc_static_shell> configure_tag -app LP -enable -tag
%vc_static_shell> configure_tag -app LP -all -verbose > all_tags.rpt
```

The following example shows VC LP reports of a design before and after configuration of certain tags/checks.

The following is the output without the use of the `configure_tag -app LP` command:

```
%vc_static_shell>report_violations -app LP
-----
Tree Summary
-----
Severity Stage Tag Count
----- -----
error UPF LS_STRATEGY_MISSING 5
error Design ISO_INST_MISSING 2
error Design ISO_STRATEGY_MISSING 2
----- -----
Total 9
```

The following is the output after the use of the `configure_tag -app LP` command:

```
%vc_static_shell> configure_tag -app LP -tag ISO_* -disable
%vc_static_shell>check_lp -stage upf
%vc_static_shell>check_lp -stage design
%vc_static_shell>report_violations -app LP
-----
Tree Summary
-----
Severity Stage Tag Count
----- -----
error UPF LS_STRATEGY_MISSING 5
----- -----
Total 5
```

### 2.2.5.1 Segregating Violations Based on Goals

If you want to define a cluster of VC LP tags that should run together at certain point in design flow, then you can develop your own goal, and use these goals at different stages.

For example, you can disable functional checks when you are running the PG stage checks, as the functional checks are useful at RTL only.

To segregate VC LP violations, you can use the `-goal` option in the `configure_tag -app LP` command.

When the `configure_tag -app LP` command is executed it takes immediate effect, however, the `configure_tag -app LP -goal` command has no immediate effect. This command is added to the bucket of that goal. When the `check_lp` command is executed with the `-goal` option, the `configure_tag -app LP` commands with the defined goals stored in the bucket are executed.

#### Examples

```
configure_tag -app LP -tag ISO_INST_MISSING -severity warning
configure_tag -app LP -disable -tag RET_INST_MISSING -disable -goal g1
configure_tag -app LP -tag ISO_INST_MISSING -disable -goal g2
check_lp -stage all -goal g1
check_lp -stage all -goal g2
```

- ❖ When the first `check_lp` command is executed, the configuration is reset (so `ISO_INST_MISSING` is reset to its default severity of error) and then the configuration commands of `g1` are executed. The `RET_INST_MISSING` is disabled.

- ❖ When the second `check_lp` command is executed, the configuration is reset and the `ISO_INST_MISSING` violation is disabled. The violation database contains `ISO_INST_MISSING` errors from goal `g1`, and `RET_INST_MISSING` errors from goal `g2`.
- ❖ When the same goal is used in multiple `configure_tag -app LP` commands, the commands get stored in the same bucket, and they are executed when `check_lp` is used with the `-goal` option.
- ❖ Wild card is not supported as the goal name, VC LP reports the following error.  
`INVALID_GOAL_NAME: Goal name 'g*' is not valid. The name must be a valid tcl identifier.`
- ❖ Goal is not defined and used in `check_lp`, VC LP reports the following parsing warning, and it will run in the default mode.  
`[Warning] GOAL_NOT_DEFINED: Goal name 'g1' is not defined. Goal ignored, running all related checks`
- ❖ All the violation in the `report_violations -app LP` command, reports the `Goal` debug field. You can waive the violation based on the goal name.

```

Tag      : PG_SUPPLY_NOPORT
Description : UPF supply port [SupplyPort] defined but does not exist in design
Violation   : LP:8
SupplyPort
  PortName   : VDD
  PortType   : UPF
  Instance    : top
  Cell        : top
  Goal        : g1

```

## 2.2.6 Running Electrical Sign off Checks

When you are in the sign-off stage for tape-out, and you are not concerned about correctness of UPF. Therefore, all missing, redundant, multiple, conflicting and unused checks related to UPF strategy are not sign-off checks. You can check for the correctness of UPF much earlier in the design. During the final sign-off you are only concerned about electrical correctness of the design. In such cases, you can use the `configure_lp_electrical` command to check for a limited set of checks.

Use the `configure_lp_electrical` command before `check_lp -stage upf`, `check_lp -stage design`, and `check_lp -stage pg` commands.

### Syntax

```
%vc_static_shell>configure_lp_electrical -help
Usage: configure_lp_electrical # Enables a minimal set of Low Power checks for
electrical correctness
```

### Use Model

#### **report\_violations -app LP command output without use of configure\_lp\_electrical command:**

```
%vc_static_shell>check_lp -stage upf
%vc_static_shell>check_lp -stage design
%vc_static_shell>report_violations -app LP
-----
Tree Summary
-----
Severity Stage Tag Count
-----
error     UPF   ISO_STRATSUPPLY_INCORRECT      1
error     Design ISO_INST_MISSING             12
```

warning	UPF	ISO_STRATEGY_REDUND	4
warning	Design	ISO_STRATEGY_UNUSED	16
<hr/>			
Total			33

**report\_violations -app LP command output after use of configure\_lp\_electrical command:**

```
%vc_static_shell>configure_lp_electrical
%vc_static_shell>check_lp -stage upf
%vc_static_shell>check_lp -stage design
%vc_static_shell>report_violations -app LP
```

---

Tree Summary

---

Severity	Stage	Tag	Count
error	Design	ISO_INST_MISSING	12

---

**2.2.6.1 Signoff\_Tags included in configure\_lp\_electrical:**

- ❖ ISO\_INST\_MISSING
- ❖ ISO\_INSTMISSING\_NOISO
- ❖ LS\_INGND\_CONN
- ❖ LS\_INPWR\_CONN
- ❖ LS\_INST\_INCORRECT
- ❖ LS\_INST\_MISSING
- ❖ LS\_INST\_RANGEI
- ❖ LS\_INST\_RANGEO
- ❖ LS\_OUTGND\_CONN
- ❖ LS\_OUTPWR\_CONN
- ❖ PG\_BIAS\_INSUFFICIENT
- ❖ PG\_BIASOFF\_STATE
- ❖ PG\_DOMAIN\_SETUP
- ❖ PG\_GNDNODE\_STATE
- ❖ PG\_PIN\_TIED
- ❖ PG\_PIN\_UNCONN
- ❖ PG\_PWRDIODE\_STATE
- ❖ PG\_RANGEDIODE\_STATE
- ❖ PST\_STATE\_ZERO
- ❖ ISO\_BUFINV\_LEAKAGE
- ❖ ISO\_BUFINV\_STATE
- ❖ ISO\_ELSINPUT\_STATE
- ❖ ISO\_ELSOUTPUT\_STATE

- ❖ LS\_INTRACELL\_LEAKAGE
- ❖ ISO\_INPUT\_STATE
- ❖ ISO\_OUTPUT\_STATE
- ❖ ISO\_LSINPUT\_LEAKAGE
- ❖ ISO\_LSINPUT\_STATE
- ❖ ISO\_LSOUPUT\_LEAKAGE
- ❖ ISO\_LSOUPUT\_STATE
- ❖ ISO\_SINK\_STATE

# 3 Reading and Building Design

Before performing various analysis operations on the design, you must first read and build the design. You can then later analyze, debug and resolve any potential problems.

The high level flow for a LP design check has the following steps.

- ❖ “[Reading the Liberty Files](#)”
- ❖ “[Reading the Design](#)”
- ❖ “[Checking Successfully Built Design](#)”
- ❖ “[Reading Power Intent](#)”
- ❖ “[Analyzing Compiler Messages](#)”
- ❖ “[Running VC LP Checks](#)”
- ❖ “[Checking Reports](#)”
- ❖ “[Saving and Restoring Sessions Using save\\_session and -restore](#)”
- ❖ “[Saving and Restoring Sessions Using CR Technology](#)”
- ❖ “[Support for Auto-Saving and Auto-Restarting Sessions](#)”
- ❖ “[Support for Comparing Results Of Two VC LP Sessions](#)”
- ❖ “[Handling Encryption](#)”
- ❖ “[Extracting Test Cases Automatically](#)”
- ❖ “[In-design integration of ICC2 and VC LP](#)”

## 3.1 Reading the Liberty Files

For pure RTL designs, supplying liberty files is not necessary. However, for RTL designs with pre-instantiated LP cells (such as ISO, LS, macros, pads) and for most logical/physical netlist designs supplying liberty files are necessary. You must provide all the required liberty files using `search_path` and `link_library` application variables before reading the design. Once the `link_library` and `search_path` are set, read the design using the `read_file` command.

### 3.1.1 The `search_path` and `link_library` Variable

The `search_path` application variable specifies a sequence of directories where VC LP looks for the library (.db) files. The specified directories are searched when a new library file is to be loaded.

```
%vc_static_shell> set_app_var search_path <list of all the paths>
```

- ❖ Specify all the paths where the library files, design or UPF files must to be searched. The paths may be absolute or relative to the directory in which VC LP is invoked.
- ❖ If multiple paths are present, they should be provided as space separated values within double quotes.
- ❖ The `search_path` variable supports environment variables.
- ❖ The `search_path` variable does not support Wildcards characters.

The `link_library` application variable specifies a list of .db library files to be searched when a cell instantiation is to be resolved.

```
%vc_static_shell> set_app_var link_library <list of .db files>
```

- ❖ Specify all the library files that are required to be read.
- ❖ Only Liberty .db files (not .lib files) will be read into the tool.
- ❖ If multiple .db files are present, they should be provided as space separated values within double quotes.
- ❖ The `link_library` variable does not support environment variables.

### Example

```
%vc_static_shell> set_app_var set search_path ". path1 path2 ..."  
%vc_static_shell> set_app_var set link_library "lib1 lib2 ... libN"
```

## 3.1.2 Tcl Commands to Append Missing Liberty Attributes

Synopsys flow recommends liberty files to be low power ready, that is, they have pg pins defined and have all low power specific attributes as relevant. If the supplied liberty files are not compliant with Synopsys LP flow recommendations, you can add the missing attributes using the following on the fly PG commands as below.

- ❖ `set_pin_model`
- ❖ `set_pg_pin_model`
- ❖ `set_always_on_cell`
- ❖ `set_isolation_cell`
- ❖ `set_level_shifter_cell`
- ❖ `set_retention_cell`

For more details on each of these commands, refer to *VC Static Platform Command Reference Guide*.

### Examples

```
set_pg_pin_model -cells MACRO_LIB/M_ABC*  
-pg_pin_name {VDD VDDBAK VSS VSSBAK} -pg_voltage_name {VDD VDDBAK VSS VSSBAK} -  
pg_pin_type {primary_power backup_power primary_ground backup_ground }  
  
set_isolation_cell ISO* -enable_pin EN -data_pin A  
set_pg_pin_model ISO* \  
-pg_pin_name { VDD VSS } \  
-pg_pin_type { primary_power primary_ground} \  
-pg_voltage_name {VDD VSS}  
  
set_power_switch_cell SW* -cell_type coarse_grain -switch_pin {SLEEP  
SLEEPSOUT} -pg_pin {VDDG SLEEP VDD}
```

```

set_pg_pin_model SW* \
-pg_pin_name { VDD VDDG VSS } \
-pg_pin_type {internal_power primary_power primary_ground} \
-pg_voltage_name {VDD VDDG VSS}
set_pin_model SW* \
-pins {SLEEP SLEEPOUT} \
-related_power_pin VDDG \
-related_ground_pin VSS

set_retention_cell RET* -cell_type DRFF -retention_pin {RETN save_restore 1}
set_pg_pin_model RET* \
-pg_pin_name { VDD VDDG VSS VSSG } \
-pg_pin_type { primary_power backup_power primary_ground backup_ground } \
-pg_voltage_name { VDD VDDG VSS VSSG }
set_pin_model RET* \
-pins { D SI SE RN CK Q QN } \
-related_power_pin VDD \
-related_ground_pin VSS
set_pg_pin_model RET* \
-pins { RETN } \
-related_power_pin VDDG \
-related_ground_pin VSSG

```

The `set_pg_pin_model` command creates a new pg pin and sets the relevant attribute on it if it is executed before the design is read. If the command is executed after the design has been read, no new pg pin is added, and if the pin referred to is found, the appropriate attribute is added to it.

Additionally, when these commands are provided before the design is read, they are not executed immediately. The syntax check happens immediately and after the design loads. The commands are executed and they affect the design or the library. If the hierarchical cell name <lib/cell> is specified, the lib name is ignored and the command is applied to the matching cell in all libraries.

## 3.2 Reading the Design

VC LP reads design in RTL (verilog, VHDL, System Verilog) and netlist (verilog) formats.

VC LP provides the following commands to read a design:

- ❖ `read_file`: Read in design source files, and link design in memory. This command can be used to load design in a single language (Verilog/SV or VHDL). Using this command, you can specify all source files in one command in a single language environment. The files get analyzed and then elaborated. Upon completion of the command the complete design has been loaded and is ready to be used. The command returns 1 on success and 0 on failure.

### Syntax

```
%vc_static_shell> read_file -help
Usage: read_file      # Reading design files
       [-top <top_design>]      (Name of the top design)
       [-library <library_name>]    (Remaps work library to library_name)
       [-define <list of verilog defines>]   (Verilog/SV defines)
       [-work <work_library>]   (alias for -library)
       [-netlist]                (Verilog Netlist Reader)
       [-parameters <string containing ordered or named parameters seperated by comma>]
                                         (design parameters)
```

```

[-vcs <vcs command line>]
    (VCS Command line for reading design)
[-vcs_elab <vcs elaborate command line>]
    (VCS Command line for elaborating design)
[-format <file format>]
    (Verilog/SV defines:
     Values: verilog, sverilog, vhdl, mdb)
[-sva]
    (Process SVA/PSL during compilation using 2009 semantics)
[-sva2005]
    (Process SVA/PSL during compilation using 2005 semantics)
[-v2kconfig <configuration-name>]
    (Specifies the v2k configuration)
[-buildTop <dut name>] (Specifies the DUT down from which synthesis model is
                         generated)
[-multi_step]
    (Load design in multi step mode)
[-cov <metric_type>]
    (Enables coverage instrumentation during compilation)
[-l1k <l1k_type>]
    (Creates livelock goals during compilation)
[-aep <aep_type>]
    (Enables AEP extraction during compilation)
[-inject_fault <fault_type>]
    (Injects behavioral faults in the design for doing sign-off with formal)
[-j <number_of_processes>] (Specifies the number of processes to use for
                           parallel compilation: Value >= 1)
[slist]
    (List of input files)

```

### Note

The `read_file -j 0` command starts the design read in serial mode.

### Note

For more details on how to compile a design using the VCS standard switches, see the *VCS® MX/VCS® MXi™ User Guide*. You can download this document from [SolvnetPlus](#).

- ❖ `analyze`: Analyzes the specified HDL source files and stores the design templates they define into the specified library in a format ready to specialize and elaborate to form linkable cells of a full design. Using this command, you can specify multiple source files in a single language in one command. Upon completion of the command all specified files are analyzed and ready to be elaborated. The command returns 1 on success and 0 on failure.

### Syntax

```

%vc_static_shell> analyze -help
Usage: analyze      # Analyze the source files
       [-format <file_format>]
           (Specify file format:
            Values: verilog, vhdl, sverilog, sysc,
                    spi)
       [-library <library_name>]
           (Remaps the work library to library_name)
       [-work <library_name>] (Remaps the work library to library_name)
       [-define <define_macros>]
           (Specify list of top-level macros)
       [-vcs <vcs_cmd>]      (VCS Command line for reading design)
       [design_file_list]     (List of source files)

```

### Note

When you use the `-vcs` switch, the `vlogan` and `vhdlan` arguments and switches must be enclosed in curly braces.

- ❖ **elaborate:** Builds a design from the intermediate format of a Verilog module, a VHDL entity and architecture, or a VHDL configuration. Using this command, the user can elaborate design from pre-analyzed design files, from a specified top module. This command returns 1 on success and 0 on failure.

## Syntax

```
%vc_static_shell> elaborate -help
Usage: elaborate      # Elaborate the design, which is analyzed using analyze command
       [-work <library_name>] (Specifies the library name to which work is to be
mapped)
       [-library <library_name>]
                           (Specifies the library name to which work is to be
mapped)
       [-architecture <arch_name>]
                           (Specifies the name of the architecture)
       [-parameters <param_list>]
                           (Specifies a list of design parameters enclosed in
quotes)
       [-file_parameters <file_list>]
                           (Specifies a list of files that contain parameter
specifications)
       [-vcs <vcs_cmd>]      (VCS Command line for elaborating design)
       [-sva]                  (Process SVA/PSL during compilation using 2009 semantics)
       [-sva2005]               (Process SVA/PSL during compilation using 2005 semantics)
       [-v2kconfig <configuration-name>]
                           (Specifies the v2k configuration)
       [-buildTop <dut name>] (Specifies the DUT down from which synthesis model is
generated)
       [-cov <metric_type>]    (Enables coverage instrumentation during compilation)
       [-l1k <l1k_type>]        (Creates livelock goals during compilation)
       [-aep <aep_type>]        (Enables AEP extraction during compilation)
       [-inject_fault <fault_type>]
                           (Injects behavioral faults in the design for doing sign-
off with formal)
       [-j <number_of_processes>]
                           (Specifies the number of processes to use for parallel
compilation:
                           Value >= 1)
       design_name            (Specifies the name of the design to build)
```



## Note

- (1) If there is one design top, it must not be passed using vcs arguments, that is, elaborate -vcs {designtop}. It must be passed as follows: elaborate designtop
- (2) For a model with several top modules (in following example: "dut\_top" and "tb\_top"), you must pass the arguments as follows: elaborate dut\_top -vcs "tb\_top"
- (3) Use the enable\_non\_lrm application variable to disable LRM-compliant macro expansion. If this variable is set to false, VC LP produces a result that is more LRM-compliant and accurate especially for SystemVerilog macros.

- ❖ **read\_verilog:** Reads in one or more design or library files in Verilog format.

## Syntax

```
%vc_static_shell> read_verilog -help
Usage: read_verilog      # Read one or more verilog files
       [-netlist]          (Use structural Verilog netlist reader)
       [-rtl]              (Use RTL Verilog)
       file_names          (Files to read)
```

- ❖ `read_vhdl`: Reads in one or more designs or library files in VHDL format.

### Syntax

```
%vc_static_shell> read_vhdl -help
Usage: read_vhdl      # Read one or more vhdl files
      [-netlist]          (Use structural VHDL netlist reader)
      file_names           (Files to read)
```

- ❖ `read_sverilog`: Reads in one or more design or library files in SystemVerilog format.

### Syntax

```
%vc_static_shell> read_sverilog -help
Usage: read_sverilog    # Read one or more systemverilog files
      [-netlist]          (Use structural Verilog netlist reader)
      [-rtl]               (Use RTL Systemverilog)
      file_names           (Files to read)
```

For more details on each of these commands, refer to the *VC Static Platform Command Reference Guide*.

## 3.2.1 Application Variables that Impact Reading a Design

There are few application variables that impact the design read and database generation. Before you start reading the design, ensure that you review and set these application variables as per your design read requirements.

- ❖ `autobb_unresolved_modules`
- ❖ `ignore_multiple_module_def`
- ❖ `enable_dirty_data`
- ❖ `analyze_skip_translate_body`
- ❖ `hierarchy_delimiter`
- ❖ `sh_continue_on_error`
- ❖ `vnr_enable_file_split`
- ❖ `preserve_assignment_direction`

For more information on these application variables, see the man pages.

## 3.2.2 Reading RTL Designs Examples

It is recommended that you use analyze and elaborate flow for RTL design read. VC LP uses VCS as the front-end compiler for RTL design read and hence if you have an existing VCS run environment, it can easily be ported to the VC LP environment as shown in the examples below.

Use the `analyze` and `elaborate` commands to read the RTL designs as shown in the following code snippet:

```
%vc_static_shell>analyze -format verilog "inv.v top.v"
%vc_static_shell>elaborate top
```

### 3.2.2.1 Verilog RTL

#### Example 1

```
analyze -format verilog -vcs "-f filelist +incdir+../Input"
elaborate ChipTop
```

**Example 2**

```
read_file -top bt -vcs "-file list.f"
read_file -top top -format verilog -vcs "-f <source_list> +incdir+<dir1>+<dir2> " -
verbose
```

**3.2.2.2 System Verilog RTL****Example 1**

```
analyze -vcs "-f CaptureFsm.fl +define+RTL+VCS+NORAM+TOP_ON" -format sverilog
analyze -format sverilog -vcs "-f test_TrafficLightCtrl.fl
+define+RTL+VCS+NORAM+TOP_ON"
elaborate LightCtrl -verbose
```

**Example 2**

```
read_file -top top -format sv -vcs "-f <source_list> +incdir+<dir1>+<dir2> " -verbose
```

**3.2.2.3 VHDL RTL****Example 1**

```
analyze -format vhdl -work lib1 vhdlTop.vhd
elaborate -work lib1 top
```

**3.2.2.4 RTL/Netlist Mixed RTL Flow**

When RTL and netlist files are present in the Tcl script, then you must read the netlist designs using the `analyze -netlist` command. This enables VC Static to invoke different design readers for RTL file and netlist file, improving the performance of the `analyze` command.

The `analyze -netlist` command invokes Verilog Tcl Reader (VNR) to read netlist designs. The VNR writes a Verilog that has the skeleton (module having ports, but without a body) of the each circuit in the netlist, except DB cells. For the instantiated DB cells, VNR generates a list of black box which is passed to Simon during elaboration. When the RTL is analyzed, VC Static invokes `vhdlan/vlogan` depending on the format and creates the `analyze db`.

When you perform the `elaborate <top>` command, VC Static first invokes `vlogan` over the VNR generated Verilog file, and then VC Static invokes Simon (`vcs -scm ...`). Simon creates objects for the RTL. The `elaborate` command links the RTL and netlist, and then creates the SCM.

VC Static invokes different design readers for the following situations:

- ❖ RTL instantiates the whole or part of the Tcl
- ❖ Both Tcl and RTL may have same or different instance of DB cell
- ❖ RTL has a simulation library and netlist has DB library for the same cell

**Note**

Assignment to supply nets is allowed in the netlist flow when the `enable_supply_net_assignment` application variable is set to true. The `enable_supply_net_assignment` application variable is disabled by default.

VC LP reports the SVR-63 design read error when the supply nets of LHS and RHS are of different type. For example, the following definition is allowed when the `enable_supply_net_assignment` application variable is set to true.

```
supply0 vssx_0;
supply0 spare_out_0;
assign spare_out_0 = vssx_0;
```

## Example

Consider that the design files have the following hierarchy:

```
top.v
|--- rtl.vhd
    |--- netlist.v
```

Then the VC Static Tcl script must have the following commands:

```
set search_path ". . . ."
set link_library ".../sample1.db"
set target_library ".../sample1.db"
analyze -netlist netlist.v
analyze -format verilog top.v
analyze -format vhdl rtl.vhd
elaborate top
check_lp
report_violations -app LP
exit
```

## Limitations

- ❖ You cannot use the same module name for RTL and Tcl.
- ❖ You cannot read a netlist with the -netlist switch when the netlist file is a top file.
- ❖ You cannot use the -netlist option for encrypted netlist files.
- ❖ You can use \*.Gzip file as an input to the analyze command, but not the \*.tar file

### 3.2.2.5 Handling Config files as Top



#### Note

V2k configuration cannot be directly specified as a design top.

If you intend to specify v2k configuration, then you must explicitly specify the v2k configuration name as illustrated by the following command:

```
elaborate <top_name> -v2kconfig <config_name>
```

## Example

```
elaborate <top_name> -vcs {-top cfg_name ...}
```

This command option is not legal and it will error out. If there is a v2k configuration, it is recommended you pass it to the elaborate command using the -v2kconfig switch and never specify the configuration name to -vcs. A sample command option is illustrated below:

```
elaborate <top_name> -v2kconfig <config_name> -vcs <other options>
```

### 3.2.2.6 Elaborating Design Files and Config Files Parsed to Logical Library

If the design top (say: sctop) is parsed to logical library sc\_top\_li, and if the V2K config (say: m0\_design) is parsed to logical library sc\_cfg\_lib, then use the following command to elaborate a design:

```
elaborate sctop -work sc_cfg_lib -v2kconfig m0_design -vcs { ... }
```

### 3.2.2.7 Elaborating Design From a Top Module

The -buildTop option in the elaborate command helps you specify a module which is a few levels down the hierarchy from the top of the design instead of the actual design top. This helps you save time by not processing any unwanted modules not under the design hierarchy, and directly go to the problematic module that has issues.

Consider the following hierarchical representation of the design:

```

TB
 /-----DUT
 /
 /-----CPU
 /
 /-----RCG
 /
 /-----DRIVER

```

### Usage of buildTop:

For RTL:

```

analyze -format verilog "xyz_mod1.v"
elaborate TB -buildTop CPU

```



#### Note

For the example above, VC Static considers only the files under the CPU hierarchy. The UPF hierarchy must also be as per the CPU hierarchy as it is considered as the design top with the -buildTop option.

### 3.2.2.8 Elaborating Design Having Multiple Tops

When you are elaborating a design with multiple tops using the `elaborate -vcs {}` command, use the follow syntax:

In the following example, there are two tops: `dut_top` and `tb_top`.

```
%vc_static_shell>elaborate dut_top -vcs "tb_top"
```

### 3.2.2.9 Overriding Parameters for Verilog/System Verilog During Elaboration

In specific scenarios, VC Static provides an option to override parameters defined in the RTL without changing it, as follows:

```

elaborate <design_name>
    [-library <library name>]
    [-work <library name>]
    [-architecture <arch name>]
    [-parameters <list of design parameters>]
    [-file_parameters <file_list>]
    [-vcs <vcs command>]

```

The following is an example, specifying the analyze and elaborate flow on the command line:

```
>elaborate top -parameters "N=>9, M=>18"
```

In case there are several parameters, you can specify them in a file and pass it to elaborate command as follows:

```
>elaborate top -file_parameters param.txt
param.txt:
N=>9
```

M=>18

The following is an example, specifying a single compilation flow on the command line:

```
>read_file -parameters "N=>9, M=>18"
```

### 3.2.3 Reading Netlist Designs Examples

#### 3.2.3.1 Logic Netlist

To read the design, set the `search_path` and `link_library` application variables and then execute the `read_file` with `-netlist` option command. The following example is for netlist runs only:

```
%vc_static_shell> set search_path ". ../lib"
%vc_static_shell> set link_library tiny.db
%vc_static_shell> read_file -netlist iso1.v -top top
```

Only a single `read_file` command is allowed. This enables VC LP to optimize the design build process. The `-top` argument is required. If you have multiple files, specify them as a Tcl list with quotes or curly braces. For more information, see section [Building the Design](#).

The `-netlist` option must be used for compiling netlist designs as VC LP uses optimized netlist reader(VNR) for compiling netlist. If you do not use `-netlist`, VC LP uses VCS flow (RTL compiler) for reading the netlist design which can be slower compared to VNR flow.

#### 3.2.3.2 Verilog Netlist/PG Netlist

```
read_file -format verilog -netlist icc_out/chip.icc.pg.v -top chip_top
```

You can also use other design-build commands like `read_verilog`, `read_sverilog`, `read_vhdl`. For more information, see the design query commands in the *VC Static Platform User Guide*.



It is recommended that you clean up the `vcst_rtddb` database for successive VC LP runs.

#### 3.2.3.3 Mixed RTL and Netlists

For design which are largely RTL with some netlist portions, user can elaborate such designs as follows:

```
analyze -netlist netlist.v
analyze -format verilog top.v
analyze -format vhdl rtl.vhd
elaborate top
```

### 3.2.4 Support for Reporting Message Database

You can get a report of all the messages when reading design, SDC and UPF using the `report_read_violations` command. These messages can be viewed as GUI violations. This feature is available under the `enable_message_database_migration` application variable.

By default, the `enable_message_database_migration` application variable is false. If this application variable is not set, no violations are reported under `report_read_violations` command.

The `display_design_read_messages` application variable is introduced, which is enabled by default. When this application variable is enabled, the messages are reported on the screen (previous behavior), else they will be reported only when you use the `report_read_violations`.

The following three commands are introduced for this feature:

- ❖ `report_read_violations`

- ❖ remove\_read
- ❖ waive\_read

### 3.2.4.1 The report\_read\_violations Command

Use the `report_read_violations` to get a list of all the design read, SDC and UPF violations. The options of this command is similar to the other reporting commands like `report_violations -app LP`, and provide same options and filter mechanism.

#### Syntax

```
%vc_static_shell> report_read_violations -help
Usage: report_read_violations      # help
       [-no_summary]          (Suppresses summary information)
       [-list]                 (List all messages in simple form)
       [-verbose]              (List all messages in detail form)
       [-limit <count>]        (Limit the number of output records per tag)
       [-include_waived]       (Include waived messages in the report)
       [-only_waived]          (Report on waived messages)
       [-all_tags]              (Include all tested tags)
       [-tag <tag>]             (Select violations based on tag)
       [-waived <list>]        (Select violations based on waiver name)
       [-id <tag>]              (Select violations based on IDs)
       [-family <family>]       (Select violations based on family:
                                Values: all, design, sdc, upf)
       [-severity <list>]        (Select violations based on severity:
                                Values: all, error, info, warning)
       [-filter <expression>]    (Select violations based on expression)
       [-regexp]                (Indicates filter expression type to be regular expression
(default glob-style))
       [-nocase]                (Filter expressions ignore case when matching string
values)
       [-file <filename>]        (Write the results to the designated file)
       [-append]                (Append results to the designated file)
```

#### Examples

- ❖ `report_read_violations -verbose -file report_read_violations.txt`
- ❖ `report_read_violations -filter file==file.upf`
- ❖ `report_read_violations -filter file==file.upf -list`
- ❖ `report_read_violations -filter file==file.upf -list -no_summary`

#### Limitations

Many possible messages produced during RTL design read are not relevant for static analysis. Only the relevant messages for static analysis is reported by the `report_read_violations` command.

### 3.2.4.2 The remove\_read Command

Use the `remove_read` command to clear the messages generated at different stages. This command and its options re similar to the existing `remove_lpViolation` command.

#### Syntax

```
%vc_static_shell> remove_read -help
Usage: remove_read      # removes violations from the design database
       [-tag <name>]          (Remove violations based on Tag name)
       [-id <id>]              (Remove violations based on IDs)
```

[-nosave] [-save] [-family <family>]	(Skip updating removed violation in disk) (Update all removed violation(s) in disk) (Remove violations based on stage: Values: all, design, sdc, upf)
--	--

**Example**

- ❖ remove\_read -tag UPF\_FILE\_PARSING
- ❖ remove\_read -id UPF:4 -nosave
- ❖ remove\_read -family design

**3.2.4.3 The waive\_read Command**

Use the `waive_read` command to create waiver on messages generated at different stages. The `waive_read` command is similar to the existing `waive_lp` command.

**Syntax**

```
vc_static_shell> waive_read -help
Usage: waive_read      # waive design read violations
       [-verbose]           (List all messages in detail form)
       [-add <name>]        (Add waiver)
       [-append name]       (Append additional filter parameters to an existing
waiver)
       [-comment <comment>] (Waiver comment)
       [-delete <name(s)>] (Delete waiver)
       [-delete_all]        (Delete all waivers)
       [-tcl]                (Display the waivers in TCL command format)
       [-force]              (Create a container for waive_read append operations)
       [-tag <tag>]          (Waive violations based on tag)
       [-id <tag>]          (Waive violations based on IDs)
       -family <family>     (Waive violations based on family:
                           Values: upf, design, sdc)
       [-severity <list>]   (Waive violations based on severity:
                           Values: all, error, info, warning)
       [-filter <expression>] (Waive violations based on expression)
       [-regexp]             (Indicates filter expression type to be regular expression
(default glob-style))
       [-nocase]             (Filter expressions ignore case when matching string
values)
```

**Example**

- ❖ waive\_read -family design -delete Design\_info
- ❖ waive\_read -add UPF\_FILE\_PARSING\_1 -tag UPF\_FILE\_PARSING -family UPF -comment "GUPF parsing complete...."

**3.2.5 Migrating Designs from Other Synopsys Tools****3.2.5.1 Migrating from VCS Setup**

You can migrate a VCS or VCSMX simulation setup to VC Static setup as VC Static uses VCS as its front end RTL compiler. In addition, VC Static can read VCS simulation setups such as `synopsys_sim.setup` (that contains the pre-compiled libraries, logical to physical library mapping, and so on).

You can provide your own `synopsys_sim.setup`:

- ❖ In current working directory

- ❖ Home directory
- ❖ Setup file pointed by the SYNOPSYS\_SIM\_SETUP environment variable

In conflicting settings, VC LP takes precedence in the following order:

- ❖ Setup file pointed by environment variable SYNOPSYS\_SIM\_SETUP
- ❖ synopsys\_sim.setup in the current working directory
- ❖ synopsys\_sim.setup in the home directory

**Figure 3-1** **synopsys\_sim.setup Example**

```
user's synopsys_sim.setup
DW
MY_LIB : /u/library/precompiled/DW
/u/library/my_lib
```

### Example 1

Compiling MX designs into physical libraries

```
%vc_static_shell > analyze -format vhdl -library my_lib my_pkg.vhd
```



The /u/library/my\_lib must exist and have write permission.

```
%vc_static_shell> analyze -format vhdl sys_pd.vhd
```



The /u/library/precompiled/DW must exist and have read permission.

The analyze, elaborate and read\_file commands take -vcs as an option where you can pass on all the VCS (vlogan, vhdlan and so on) options directly.

```
%vc_static_shell> analyze -format vhdl -library my_lib { dut.e.vhd dut.a.vhd } -vcs {-xlrn }
```

### Example 2

In the following sample design, three types of sample RTL files are defined (Verilog, VHDL and SystemVerilog). VCS follows a three step process; VC LP analyzes and elaborates on the flow.

#### VCS command options:

The VCS command which can be used in VC LP to read an RTL file is as follows:

```
vlogan -sverilog -work work1 bubbled_or.sv bubbled_and.sv
vhdlan -work work2 module_sub_1.vhd module_sub_2.vhd top_module.vhd
vlogan -work work3 inv.v
vcs work2.top_module
simv
```

#### synopsys\_sim.setup:

A common synopsys\_sim.setup is used for VCS as well as VC LP. The following are the common settings that work for both the tools:

```
WORK > DEFAULT
DEFAULT : work2
work1 : ./result/work1
```

```
work2 : ./result/work2
work3 : ./result/work3
```

### VC LP command options:

VC LP analyzes and elaborates the flow. Use the following VC LP commands to read an RTL file:

```
set search_path { . /remote/DEMO_LIB/pgdb}
set link_library { \
    demo_iso_12v_pg.db \
    demo_std_12v_pg.db }
analyze -format sverilog -vcs "-work work1 bubbled_or.sv bubbled_and.sv"
analyze -format vhdl -vcs "-work work2 module_sub_1.vhd module_sub_2.vhd
top_module.vhd"
analyze -format verilog -vcs "-work work3 inv.v"
elaborate top_module -work work2
read_upf vhdl_vhdl_sv_verilog.upf
check_lp -stage upf -verbose
check_lp -stage design -verbose
report_violations -app LP -verbose -file report_lp.txt
quit
```



#### Note

You must manually create a directory structure called `synopsys_sim.setup` in VCS; however, VC LP creates these directories automatically.

### 3.2.5.2 Migrating Designs from DC/PT setup

The netlist reader (VNR) that VC Static uses is shared by DC/PT, and hence if you have a DC/PT setup, you can set up VC Static on it.

For reading mixed RTL designs that do not have `synopsys_sim.setup` in VC Static, you can still specify the logical to physical mapping in DC style from `vc_static_shell`.

```
%vc_static_shell> define_design_lib my_lib -path /u/library/my_lib
%vc_static_shell> analyze -format vhdl -library my_lib my_pkg.vhd
```

VC static internally creates a mapping at `my_lib : /u/library/my_lib`

If you are not concerned about the physical location where the RTL is compiled, then you can specify the `-library` option in the `analyze` command. The physical path is present in the `vcst_rtdb` directory.

```
%vc_static_shell> analyze -format vhdl -library my_lib { dut.e.vhd dut.a.vhd }
```

VC static internally creates the following mapping

```
my_lib : vcst_rtdb/.internal/design/my_lib
```

For reading a netlist design in VC Static, it is recommended that you use `-netlist` option with `read_file`.

```
%vc_static_shell> read_file -format Verilog -netlist top.vg -top top
```

### 3.2.6 Support for MULTI\_PORT\_MEM Operator

VC LP supports the MULTI\_PORT\_MEM NLDM operator. Whenever VC LP encounters a multi-port memory in the design, VC LP internally represents it as a MULTI\_PORT\_MEM NLDM.

There is no change in terms of quality of results when this feature is enabled. As the internal representation is a single operator which is compact as compared to earlier representation, improvement in runtime can be

seen for designs having multi port memories in RTL. There would not be any performance impact if the multi-port memories are in the DB format, VC LP uses multi-port memories from the database.

### 3.2.6.1 Use Model

To turn on this feature, you must set the following application variable:

```
set_app_var disable_memx_transform false
```

and the following command must be added in the script:

```
configure_mem_macro_inference
```

By default, the `disable_memx_transform` application variable is set to true. To enable the feature, you must set it to `false`.

## 3.2.7 Handling DesignWare Components

The VC Static Platform supports designs that include the Synopsys DesignWare (DW) components. The DW source libraries shipped by Synopsys contain two versions of the DW parts. Therefore, additional steps are needed to ensure the correct version is used. The following are the two versions behavioral models written in Verilog and used for simulation. If used in a VC Static application, the behavioral code is surrounded by Synopsys translate off/on pragmas that result in an empty model or black box.

The encrypted source files written in Verilog/VHDL. These are synthesizable and can be used within Synopsys applications. These parts may be hierarchical and use other encrypted DW subparts. To use these parts in the VC Static flow, all of the parts referenced by a DW part must be analyzed prior to elaboration.

Supporting DW components in the VC Static platform is a two step approach.

- ❖ Pre-analyze the DW library as part of the VC Static installation. This is done once per customer site.
- ❖ Compile designs with DW using the pre-analyzed library.

### 3.2.7.1 Pre-analyzing DesignWare Components

You can pre-analyze your DW components into a specific location as a library in your VC LP compile flow without having to reanalyze the parts each time.

A script `$VC_STATIC_HOME/auxx/monet/scripts/vsi_dw2vcslib.csh` is provided with each VC LP release. This script compiles both 32 and 64 bit versions of the library for use with VC LP.

Inputs required by `vsi_dw2vcslib.csh`:

- ❖ Location of Synopsys DW tree (DWROOT) can be given with the `hdlin_dwroot` application variable.
- ❖ Location to install analyzed files (VSI\_DWROOT) can be given with the `vsi_dwroot` application variable.

The following are the outputs created by the `vsi_dw2vcslib.csh` script:

- ❖ Analyzed DW tree (VSI\_DWROOT).
- ❖ Mapping file containing logical to physical mapping of DW components.

The `vsi_dw2vcslib.csh` script is a wrapper around the script that is shipped as part of the Synopsys DW library(<DWROOT>/dw/scripts/dw\_analyze\_idp.csh).

The `dw_analyze_idp.csh` script has the necessary `vhdlan`/`vlogan` calls for analyzing all of the DW parts shipped in that version of the DW library. The mapping file created as part of this process specifies the logical to physical map of the library.

Pre-analysis is done by calling the `dw_analyze` command. The `dw_analyze` command is a wrapper around the `dw_analyze_idp.csh` script. The `dw_analyze_idp.csh` script for VC is placed in:

```
$VC_STATIC_HOME/monet/scripts/vsi_dw2vcslib.csh.
```

### Syntax

```
%vc_static_shell> dw_analyze -help
Usage: dw_analyze      # Analyze the DW source files.
       [-log log_file]      (log file)
       -dwroot dw_source_tree (DW source)
       install_directory     (writable installation directory)
```

Notes:

If both `-dwroot` and Tcl variable `hdlin_dwroot` are unset, the command throws an error.

If `-dwmapfile` is not specified, the default value is `vcst_rtdb/reports/dw_map_file`. If `-log` is not specified, the log file will be in `vcst_rtdb/logs/dw_analyze.log`.

### 3.2.7.2 Compiling Designs with DesignWare Libraries

To compile the design using the pre-analyzed libraries, specify the location of the VC LP DW library. This can be done as shown below:

```
%vc_static_shell> set_app_var vsi_dwroot <VSI_DWROOT>
```

When this variable is set, the mapping of the logical to physical libraries needs to be added to the `synopsys_sim.setup` file for elaboration. For example, if the map file was created in the pre-analysis step, it is copied into the setup file, otherwise the application must supply the mapping explicitly. Since DW parts may contain other DW parts you must always supply a complete mapping.

### 3.2.7.3 Use Model

When you run the `read_file` or `analyze` command, the following execution paths are possible:

1. If the `hdlin_dwroot` and `vsi_dwroot` application variables are set, the following steps are executed:
  - ◆ Pre-analysis is done by calling the command `dw_analyze: dw_analyze -dwroot $hdlin_dwroot -dwmapfile mapfile $vsi_dwroot.`
  - ◆ The mapfile is `vcst_rtdb/.internal/design/dw_map_file`.
  - ◆ The log file is `vcst_rtdb/.internal/design/dw_analyze.log`.
  - ◆ Merge the mapfile produced by the script into `synopsys_sim.setup` of VC LP.
  - ◆ Invoke `vlogan/vhdlan` to analyze the design.
2. If the `hdlin_dwroot` application variable is not set, and the `vsi_dwroot` application variable is set, analyze the DW Analysis version file.
  - ◆ If the version is compatible with the current VC LP:
    - ◆ Generate the mapfile according to `vsi_dwroot`
    - ◆ Merge the mapfile into `synopsys_sim.setup` of VC LP
    - ◆ Invoke `vlogan/vhdlan` to analyze the design
  - ◆ If the version is incompatible:
    - ◆ Print an error message
    - ◆ Invoke `vlogan/vhdlan` to do analyze

3. If `hdlin_dwroot` is set and `vsi_dwroot` is not set, use `vcst_rtdb/.internal/design/vsi_dwroot` as `vsi_dwroot`
4. If both variables `hdlin_dwroot` and `vsi_dwroot` are not set:
  - ◆ Invoke `vlogan/vhdlan` to do analyze

### 3.2.8 Improving Performance of Design Reads

You can improve the design read times in VC Static using either (1) or (2) or both (1) and (2) depending on the constraints mentioned below:

1. In some setups, it is possible that you might not have a list of the RTL files that must be compiled in one go. You may be compiling one or a small set of files with each `analyze` or `read_file` command. Most mixed language designs and those compiling to several work libraries, have setups where multiple `analyze` commands are used in their design read setup.

```
% vc_static_shell -file vcst_inp.tcl
```

where `vcst_inp.tcl` has several `analyze` commands such as ...

```
analyze 1.v -work work1
analyze 2.v -work work2
analyze 3.v
...
analyze 100.v
```

Such a file list can slow down the design read time in VC Static. However, under a mode, the tool can automatically combine them into fewer actual analysis operations, thereby improving design read times. To enable this, set the `enable_opt_analyze` application variable before any `analyze/read_file` command.

```
%vc_static_shell> set_app_var enable_opt_analyze true
```

2. If you have machines with multiple cores, VC Static can make use of them to boost the design read times. This feature is called parallel SIMON and is applicable to RTL design reads only.

The option `-j <NUM_PROCESS>` can be used with the `read_file` or `elaborate` commands to specify the number of processes to use for parallel compilation in an RTL flow.

#### Use Model:

```
analyze top.v
elaborate top -j <NUM_PROCESS>
```

```
read_file top.v -top top -j <NUM_PROCESS>
```

Where `NUM_PROCESS` is an integer. Values less than or equal to 1 is ignored and VC LP runs in the sequential mode (single process run) only. If the `read_file` command is used without the `-top` option, or used with the `-netlist` option, then, option `-j` will be ignored.



#### Note

To enable multi-threading feature for `read_file` stage, the VT-VC-BETA license should be checked out.

#### Ideal value of NUM\_PROCESS (value of j):

The value of `j` should be less than the number of cores available on the machine. Any value that is greater than the number of cores available degrades the performance. Through experiment and regression benchmarks data, it is found that the value of 4 or 8 (provided such number of cores is available on machine) is ideal for `j` for most of the benchmarks. You must use a multi-core machine

to run parallel compilations in the RTL flow. If you are not using a multi-core machine, then you must not use the `-j` option, as it degrades the performance.

## 3.3 Checking Successfully Built Design

You can check if a design is built successfully using the `report_cell_classification`, `report_link`, and `report_read` commands:

### 3.3.1 Checking Design Consistency and Completeness

The customer liberty file may contain cells that are modeled incorrectly (or incompletely) with respect to their power behavior. Currently, these malformed models may or may not result in false violations - with no clear indication of what the underlying problem is.

It is highly recommended that you review the libraries and their deviations reported by the tool to fix them so that they do not cause false positives or false negatives during a low power sign-off checking process using VC LP.

For all the attributes which VC LP uses, VC LP provides the `report_cell_classification` command which can be used to check low power library cell for modeling consistency, completeness and report problems, if any.

#### Syntax

```
%vc_static_shell> report_cell_classification -help  
Usage: report_cell_classification      # Reports the design cells found and analyzed  
       [-type <type>]           (Specifies one type to report:  
                                Values: blackbox, diode, macro,  
                                       multirail_cell, pad, physical_cell,  
                                       power_switch, protection, retention,  
                                       standard_cell, tie_cell)  
       [-summary]                (Report Summary Table only)  
       [-exhaustive]             (Generates report for all loaded cells)  
       [-deviation]              (Generates report for only cells that show deviation from  
expected)
```

#### Use Model

```
%vc_static_shell>report_cell_classification
```

This command can be used after the `read_design` command (design loaded). It is OK to use this command before or after the `read_upf` command. The libraries specified by `link_library` and `search_path` contain DB cells. Usually, only some of them are used in the real design. By default, the `report_cell_specification` command checks only the used DB cells.

#### Other Use Models

```
%vc_static_shell>report_cell_classification  
%vc_static_shell>report_cell_classification -summary  
%vc_static_shell>report_cell_classification -exhaustive -summary  
%vc_static_shell>report_cell_classification -exhaustive -deviation  
%vc_static_shell>report_cell_classification -type diode -deviation  
%vc_static_shell>report_cell_classification -type standard_cell -type macro -exhaustive
```

Cells with the same name may exist in multiple libraries. In this case, only the first cell is checked and reported. The order depends on the setting in the `link_library` and `search_path`.

### Example

```
%vc_static_shell> report_cell_classification
```

Summary Table:

Cell Type	Total / Deviations
protection	2 / 1
retention	1
power_switch	1 / 1
diode	2 / 2
physical_cell	0
pad	1
tie_cell	1
macro	2
multirail_cell	2
blackbox	0
standard_cell	2



**Note**  
Meaning of Deviation: Cells that have more or less information in the db with respect to the supported, expected characteristics.

**Figure 3-2 Example of the Report Cell Classification Exhaustive Output**

Cell Name	Cell Type	Deviation	Reason
<hr/>			
DMISAN2	protection		
DMLSUAN12V	protection	true	#std_cell_main_rail > 1, #input > 2, #level_shifter_enable_pin != 1
DMFD1RR	retention		
DMPSWHEAD	power_switch	true	#std_cell_main_rail(internal_power  internal_ground) < 1
FOOTPRINT_ANTEENNA	diode	true	antenna_diode_type!=1 2 0
FOOTPRINT_DIODE	diode	true	antenna_diode_type!=1 2 0
IOCB2EBTNN2LA01	pad		
TIEHI	tie_cell		
MRM1	macro		
DMAOBF	multirail_cell		
DMAOIV	multirail_cell		
DMBF	standard_cell		
DMEO	standard_cell		

#### 3.3.1.1 Checking Two Libraries Models for the Same cell with Different Attributes

When there are two libraries models for the same cell with different attributes, VC LP picks only one cell attributes for elaboration and linking, however, you may expect the other cell with different attributes is picked. As the cell names are same but attributes are different, which cell is chosen for linking can impact electrical checks. Therefore, `report_cell_classification` is improved to report inconsistencies when there are multiple libraries defined for the same cell. However, it is recommended that you provide only one liberty model per cell.

The `report_cell_classification` Tcl command reports deviation for cases when there are two libraries/DBs (with different file names) containing the same cell with different attributes.

Currently, the `report_cell_classification` command reports deviation if one of the following attributes is different:

- ❖ voltage\_map value

- ❖ related\_power\_pin
- ❖ related\_ground\_pin
- ❖ input\_voltage\_range
- ❖ output\_voltage\_range
- ❖ Number of Logic Pins
- ❖ Number of PG Pins
- ❖ Different Pin Names
- ❖ Function attribute, with missing related\_power\_pin and/or related\_ground\_pin

If a lib pin has 'function' attribute, with missing related\_power\_pin and/or related\_ground\_pin or input\_voltage\_range/output\_voltage\_range attributes, the issue will be captured under "report\_cell\_classification" tcl command output and the "LIB\_PINATTR\_MISSING" tag. These will be triggered even when 'is\_unconnected' attribute present at the lib pin.

### Example

Consider the following two cells defined in two libraries. These cells have different attributes in the libraries.

DB1

```
===
cell(MY_CELL) {
.....
pin(A) {
related_power_pin : VDD;
related_ground_pin : VSS;
...
}
.....
}
```

DB2

```
===
cell(MY_CELL) {
.....
pin(A) {
related_power_pin : VDD1;
related_ground_pin : VSS;
...
}
.....
}
```

The following is output of the report\_cell\_classification command:

Cell Name	Cell Type	Deviation	Reason
MY_CELL	standard_cell	true pin A: related_power_pin VDD	DB2/MY_CELL pin A: related_power_pin VDD1

### 3.3.2 Deviations Reported Using the report\_cell\_classification

The following table provides all the deviations reported with the report\_cell\_classification -deviation command.

**Table 3-1 Deviations Reported Using report\_cell\_classification -deviation**

Cell Types	Deviations by report_cell_classification	What's the check	Library Check Tags
<b>1. Common checks for all cells</b>	#pg_pin == 0	This deviation is reported if the cell has no pg pins.	LIB_CELLPIN_COUNT
	#primary_power == 0	This deviation is reported if the cell has pg pins, but missing primary_power pg pin.	LIB_CELLPIN_COUNT
	#primary_ground == 0	This deviation is reported if the cell has pg pins, but missing primary_ground pg pin.	LIB_CELLPIN_COUNT
	#pwell != 0	The deviation is for a specific type of cells, which has backup_power pg pin, but no primary_power pg pin. If a pwell pg pin is detected, it is reported.	LIB_CELLPIN_COUNT

Cell Types	Deviations by report_cell_classification	What's the check	Library Check Tags
1. Common checks for all cells	#nwell != 0	The deviation is for a specific type of cells, which has backup_power pg pin, but no primary_power pg pin. If a nwell pg pin is detected, it is reported.	LIB_CELLPIN_COUNT
	#pg_power >2	This deviation is for isolation/level_shifter/retention cells. If the pg pins number of primary_power or backup_power is more than 2, it is reported. If the cell is with always_on attribute, this check is ignored. It's because that always_on cell expects backup_power pg pin to supply.	LIB_CELLPIN_COUNT
	#pg_ground > 2	This deviation is for isolation/level_shifter/retention cells. If the pg pins number of primary_ground or backup_ground is more than 2, it is reported.	LIB_CELLPIN_COUNT
	related_power_pin(\$port) == null	This deviation is to check if missing related_power_pin specifications on logic pins. It will be no deviation if the pin has attribute is_unconnected: true.	LIB_PINATTR_MISSING
	related_ground_pin(\$port) == null	This deviation is to check if missing related_ground_pin specifications on logic pins. It will be no deviation if the pin has attribute is_unconnected: true.	LIB_PINATTR_MISSING
	#std_cell_main_rail > 1	This deviation is to detect if the attribute std_cell_main_rail is defined on multiple pins.	LIB_CELLPIN_COUNT
	power_down_function(\$port) = null	The attribute power_down_function is expected on output or inout pins. The deviation is to detect if it's missing.	LIB_PINATTR_MISSING
	primary_power(\$port) == output	This deviation is to check the primary_power pg pin with is with direction:output.	LIB_CELLPIN_DIRECTION
	pg_function(\$port) != null	The attribute pg_function is expected for macro or PSW cells only. This deviation is reported if it's detected, but the cell type is not macro or PSW.	LIB_PINATTR_EXTRA
	switch_function(\$port) != null	The attribute switch_function is expected for macro or PSW cells only. This deviation is reported if it's detected, but the cell type is not macro or PSW.	LIB_PINATTR_EXTRA

Cell Types	Deviations by report_cell_classification	What's the check	Library Check Tags
<b>1. Common checks for all cells</b>	isolation_cell_enable_pin(\$port) == true	The attribute isolation_cell_enable_pin is expected for isolation or ELS cells. This deviation is reported if it's detected, but the cell type is not isolation or ELS.	LIB_PINATTR_EXTRA
	isolation_cell_data_pin(\$port) == true	The attribute isolation_cell_data_pin is expected for isolation or ELS cells. This deviation is reported if it's detected, but the cell type is not isolation or ELS.	LIB_PINATTR_EXTRA
	alive_during_partial_power_down(\$port) == true	The attribute alive_during_partial_power_down is expected for isolation or ELS cells. This deviation is reported if it's detected, but the cell type is not isolation or ELS.	LIB_PINATTR_EXTRA
	function attribute not defined	For cells of DLS, LS, ELS, Isolation, power switch coarse, tie and standard cells, the pin attribute function is expected on output and inout logic pins. This deviation is reported if it is missing. This check is not performed on macro, pad, retention, blackbox, diode, power_switch_fine and physical only cells.	LIB_CELLPIN_COUNT
	short_ports: (\$port1, \$port2)	This deviation is reported if cell attribute short_ports is detected.	LIB_CELLPIN_SHORT
	<lib1/cell1> <lib2/cell1> <attr_name>: <value_lib1> <value_lib2> The <attr_name> could be: voltage_map, input_voltage_range, output_voltage_range, "logic pin number", "pg pin number", related_power_pin, and related_ground_pin.	It's to detect if a lib cell is found in different libraries with different values. It checks below attributes: voltage_map, related_power_pin, related_ground_pin, input_voltage_range, output_voltage_range, number of logic pins, number of pg pins	LIB_CELL_MULTIPLE
	antenna_diode_related_power_pins(\$port) != power	The pin listed in attribute antenna_diode_related_power_pins is expected to be a power pg pin. This deviation is reported if it's violated.	LIB_PINATTR_VALUE
	antenna_diode_related_ground_pins(\$port) != ground	The port listed in attribute antenna_diode_related_ground_pins is expected to be a ground pg pin. This deviation is reported if it's violated.	LIB_PINATTR_VALUE

Cell Types	Deviations by report_cell_classification	What's the check	Library Check Tags
<b>2. Checks on Isolation Cell (a cell with cell level attribute is_isolation_cell )</b>	outputinternal pin(<port_name>) in isolation_enable_condition	The pin listed in attribute isolation_enable_condition is expected to be an input pin. This deviation is reported if it's output or inout. This check is for normal isolation cell or macro cell with internal iso.	LIB_CELLPIN_COUNT
	#data_in > 1;	Only one data input is expected for an isolation cell.	LIB_CELLPIN_COUNT
	#data_in < 1;	Only one data input is expected for an isolation cell.	LIB_CELLPIN_COUNT
	#data_out > 1;	At most one output logic pin is expected for an isolation cell.	LIB_CELLPIN_COUNT
	#input > 2;	At most 2 input logic pins are expected for an isolation cell	LIB_CELLPIN_COUNT
	#inout != 0;	No inout logic pin is expected for an isolation cell	LIB_CELLPIN_COUNT
	#std_cell_main_rail < 1	If there are multiple primary_power pg pins, the attribute std_cell_main_rail is expected on one of them. This check is for Iso/ELS/LS/DLS/Macro.	LIB_CELLPIN_COUNT
	#backup_power > 0	This deviation is for a ELS and simple LS cell. If it's with attribute always_on specified, it's reasonable to have a backup_power pg pin. The deviation is to check if it's not always_on, but backup_power pg pin is detected.	LIB_CELLPIN_COUNT
	#level_shifter_enable_pin != 1	The attribute level_shifter_enable_pin is expected for ELS cell.	LIB_CELLPIN_COUNT
	##pg_type(related_power_pin(<pin>): <rpp>) != backup_power	This deviation is detected if the cell has both primary and backup power, and the related_power_pin of the (enable pin or output pin) is the primary power. It should be the backup power. Nor-style isolation or other types of single rail isolation do not need this type of check.	LIB_PINATTR_VALUE

Cell Types	Deviations by report_cell_classification	What's the check	Library Check Tags
<b>3. Checks on Level Shifter Cell (for a cell with cell level attribute is_level_shifter)</b>	#data_in > 1;	Only one data input is expected for a level_shifter cell.	LIB_CELLPIN_COUNT
	#data_in < 1;	Only one data input is expected for a level_shifter cell.	LIB_CELLPIN_COUNT
	#data_out > 1;	At most one output logic pin is expected for a level_shifter cell.	LIB_CELLPIN_COUNT
	#input > 1;	Only one input logic pin is expected for a level_shifter cell.	LIB_CELLPIN_COUNT
	#inout != 0;	No inout logic pin is expected for a level_shifter cell	LIB_CELLPIN_COUNT
	#backup_power > 0	This deviation is for a ELS and simple LS cell. If it's with attribute always_on specified, it's reasonable to have a backup_power pg pin. The deviation is to check if it's not always_on, but backup_power pg pin is detected.	LIB_CELLPIN_COUNT
	#std_cell_main_rail < 1	If there are multiple primary_power pg pins, the attribute std_cell_main_rail is expected on one of them. This check is for Iso/ELS/LS/DLS/Macro.	LIB_CELLPIN_COUNT
	#level_shifter_enable_pin != 1	This check is for ELS. Only one level_shifter_enable_pin is expected.	LIB_CELLPIN_COUNT
<b>4. Checks on Differential Level Shifter Cell (for a cell with cell level attribute is_differential_level_shifter)</b>	#data_in != 2	Only 2 data inputs are expected for a DLS cell.	LIB_CELLPIN_COUNT
	#data_out > 1	At most one output logic pin is expected for a DLS cell.	LIB_CELLPIN_COUNT
	#input > 2 or #input > 3	Only 2 input logic pins are expected for a simple DLS cell, and 3 input logic pins are expected for a enable DLS cell.	LIB_CELLPIN_COUNT
	#inout != 0	No inout logic pin is expected for a DLS cell	LIB_CELLPIN_COUNT
	#backup_power > 0	It's to detect if there is backup_power pg pin found in the cell.	LIB_CELLPIN_COUNT
	#std_cell_main_rail < 1	If there are multiple primary_power pg pins, the attribute std_cell_main_rail is expected on one of them. This check is for Iso/ELS/LS/DLS/Macro.	LIB_CELLPIN_COUNT

Cell Types	Deviations by report_cell_classification	What's the check	Library Check Tags
<b>4. Checks on Differential Level Shifter Cell (for a cell with cell level attribute is_differential_level_shifter)</b>	Complementary Pin without opposite pin	The pin with attribute master_complementary_pin is expected to be with pin_opposite also.	LIB_PINATTR_MISSING
	Complementary Pin also Enable pin	The pin with attribute master_complementary_pin is expected to be an enable pin.	LIB_PINATTR_MISSING
	Master Complementary Pin without opposite pin	The pin with attribute complementary_pin is expected to be with pin_opposite also.	LIB_PINATTR_MISSING
	#master_complementary_pin != 1	At least one pin is expected to be with attribute master_complementary_pin.	LIB_CELLPIN_COUNT
	#complementary_pin != 1	At least one pin is expected to be with attribute complementary_pin.	LIB_CELLPIN_COUNT
	Different RPPs in complementary and master complementary pins	The attribute related_power_pin of pins master_complementary_pin and complementary_pin are expected to be consistent.	LIB_PINATTR_MISSING
	Different RPGs in complementary and master complementary pins	The attribute related_ground_pin of pins master_complementary_pin and complementary_pin are expected to be consistent.	LIB_PINATTR_MISSING
	#inout != 0;	No inout logic pin is expected for a retention cell	LIB_CELLPIN_COUNT
<b>5. Checks on Retention Cell (for a cell with cell level attribute retention_cell)</b>	#clock != 1	One clock logic pin is expected for a retention cell	LIB_CELLPIN_COUNT
	#save != 1;	One save logic pin is expected for a retention cell	LIB_CELLPIN_COUNT
	#restore != 1;	One restore logic pin is expected for a retention cell	LIB_CELLPIN_COUNT
	#backup_power == 0 & & #backup_ground == 0	The backup_power pg pin or backup_ground_pg pin is expected for a retention cell	LIB_CELLPIN_COUNT
	##pg_type(related_power_pin(<pin>): <rpp>) != backup_power	This deviation is detected if the related_power_pin of the (save pin or restore pin) is the primary power. It should be the backup power. There are no other pins of retention cells that can be checked, such as D, clk, reset, test in, and so on.	LIB_PINATTR_VALUE

Cell Types	Deviations by report_cell_classification	What's the check	Library Check Tags
<b>6. Checks on Power Switch Cell (a cell with cell level attribute "switch_cell_type : coarse_grain")</b>	pg_function(\$port) == null	The attribute pg_function is expected on the internal_power or internal-ground pg pin. It's for macro and PSW.	LIB_PINATTR_MISSING
	switch_function(\$port) == null	The attribute switch_function is expected on the internal_power or internal-ground pg pin. It's for macro and PSW.	LIB_PINATTR_MISSING
	switched supply(\$port) != inoutoutput	The check is to report deviation if the switched supply of coarse grain switch has direction as input. The switched supply here means internal_power or internal_ground pg pins. It's for coarse grain switch only.	LIB_CELLPIN_DIRECTION
	logic pin(\$port) with direction internal in switch_function	This check is to detect the switch_function expression contains logic pins(non-PG pins) which are of direction internal. It's for macro and PSW.	LIB_CELLPIN_DIRECTION
	#std_cell_main_rail(internal_power internal_ground) < 1	Each internal_power or internal_ground pg pin is expected to be with attribute std_cell_main_rail. The deviation is reported if std_cell_main_rail is not detected on an internal_power or internal_ground pg pin.,	LIB_CELLPIN_COUNT
	#related_power_pin(<pin>:<rpp>) is switch output	It's a check for the output power. It's recognized as it is the only pg_pin which has a pg_function. This deviation is reported if the related_power_pin of any logic pin is the output power.	LIB_PINATTR_VALUE

Cell Types	Deviations by report_cell_classification	What's the check	Library Check Tags
<b>7. Checks on Macro Cell (a cell with cell level attribute "switch_cell_type : fine_grain")</b>	#std_cell_main_rail < 1	If there are multiple primary_power pg pins, the attribute std_cell_main_rail is expected on one of them. This check is for Iso/ELS/LS/DLS/Macro.	LIB_CELLPIN_COUNT
	outputinternal pin(<port_name>) in isolation_enable_condition	The pin listed in attribute isolation_enable_condition is expected to be an input pin. This deviation is reported if it's of either output or inout. This check is for normal isolation cell or macro cell with internal iso.	LIB_CELLPIN_DIRECTION
	pg_function(\$port) == null	The attribute pg_function is expected on the internal_power or internal-ground pg pin. It's for macro and PSW.	LIB_PINATTR_MISSING
	switch_function(\$port) == null	The attribute switch_function is expected on the internal_power or internal-ground pg pin. It's for macro and PSW.	LIB_PINATTR_MISSING
	logic pin(\$port) with direction internal in switch_function	This check is to detect the switch_function expression contains logic pins(non-PG pins) which are of direction internal. It's for macro and PSW.	LIB_CELLPIN_DIRECTION
<b>8. Checks on Diode cell (a cell with cell level attribute user_function_class : antennaldiode or antenna_diode_type : power  ground   power_and_gro und)</b>	#related_power_pin(<pin >):<rpp> is switch output	This deviation detects the case where the switch enable pin (with attribute switch_pin: true) is a switch_function for a pg pin, but also has a related_power_pin of this pg_pin.	LIB_PINATTR_VALUE
	antenna_diode_type != 1 2 0	It's to detect if the value of attribute antenna_diode_type is none of 0, 1 or 2	LIB_CELL_ATTRIBUTE
	user_function_class != antennaldiode	It's to detect if the value of attribute user_function_class is neither antenna nor diode	LIB_CELL_ATTRIBUTE
<b>physical_only</b>	null		

Cell Types	Deviations by report_cell_classification	What's the check	Library Check Tags
<b>9. Checks on Pad Cell (a cell with cell level attribute is_pad or pad_cell)</b>	#is_pad(logic pin) == 0	It is to check if there is no logic pin with attribute is_pad.	LIB_CELLPIN_COUNT
	#is_pad == 0	It is to check if there is neither logic nor pg pin with attribute is_pad.	LIB_CELLPIN_COUNT
<b>10. Checks on Tie Cell (a cell whose function_id is a0.0 or la0.0)</b>	#output != 1;	One output logic pin is expected for a tie cell	LIB_CELLPIN_COUNT

### 3.3.3 Identifying Black boxes, Empty Modules and Modules Without Definitions

VC LP is equipped to identify empty modules, black boxes, modules that do not have a definition and those that do not qualify the synthesis stage using the `report_link` Tcl command.

The output of the `report_link` command includes module names, which contains simulation definitions and liberty definitions. However, the port name, port width and port direction specified in an instance connection does not match with ports specified in a liberty definition. In this case, the liberty definition is discarded and the simulation definition is used in those instances. When the `-sim_match` option is specified, the output of the `report_link` command includes all modules that have a simulation and liberty definition included.

#### Syntax

```
%vc_static_shell> report_link -help
Usage: report_link      # Report status of the design build using Simon/VNR
       [-sim_match]           (Includes library cells where a simulation model is given)
```

#### Use Model

```
%vc_static_shell> report_link
Module      Instances      Reason
-----      -----
Module1      3            SimPort*, Synthesis
Module2      1            SimDirection*
Module3      17           Empty
Module4      11           AutoBB, Synthesis
Module5      12           SimWidth*, SimDirection*, SimPort*
```

The `report_link` command reports issues for the following reasons:

**Table 3-2 Reasons for report\_link Issues**

S.No	Reasons for report_link	Description
1	Empty	Modules definition is empty, that is no wire or signal in module definition has a driver.

2	Synthesis	Any part of the module is left out due to non-synth constructs or unsupported constructs, even if the majority of the module is synthesized successfully. The entire circuit is black-boxed due to tool support or if the module is detected as simulation memory.
3	UserBB	Module is black-boxed if a set_blackbox command is issued on this module.
4	AutoBB	Module is black-boxed giving synthesis as a reason.
5	SimPort*	Both liberty and simulation definition appear; the simulation definition has been kept for some of its instances because the port name of a port in instance connection does not match with any ports in the liberty definition.
6	SimDirection*	Both liberty and simulation definition appear; the simulation definition has been kept for some of its instances because the port direction of a port on the instance connection does not match the port direction of the corresponding port specified in the liberty definition.
7	SimWidth*	Both liberty and simulation definition appear; the simulation definition has been kept for some of its instances because the port width of a port in instance connection does not match port width of the corresponding port specified in the liberty definition.
8	DirtyData*	Module definition is picked after doing dirty data handling for some of its instances in the design.
9	SimMatch*	Both liberty and simulation definition appear; the liberty definition has been kept because the port list, directions and widths match exactly for some of its instances.
10	Unresolved	There is no definition for the module among any of the Simulation definition or liberty files.



### Note

STAR (\*) marked on report\_link reason indicates this reason is instance specific and you need to look into the warning or error messages to know instance specific details.

#### 3.3.4 Obtaining a Report of the Design Read and UPF Messages

While reading a design, you can get a report of the Design Read and the UPF parsing messages that are available in the violation database using the report\_read command.

##### Syntax

```
%vc_static_shell> report_read -help
Usage: report_read      # Report the messages issued in reading the design
       [-family { design_read_family_list}]
                           (Provides a list of allowable family:
                            Values: all, hdl, netlist, sdc, upf)
       [-list]           (List all messages in simple form)
       [-verbose]        (Displays short description for each message)
```

```

[-no_summary]           (Suppresses summary information)
[-tag_type <builtin | vcst | legacy>]
                      (Specify the tag naming convention:
                       Values: builtin, legacy, vcst)

```

## Use Model

```

read_upf top.upf
UPF Filename = top.upf
Parsing power intent file './top.upf'
[Warning] UPF_OSRSN: Overriding the related power/ground
          ./top.upf:63 ./top.upf:59 Overriding the related power for design port 'A' with
          'top/srsn_vdd1'. Previous related supply specified was 'top/srsn_vdd'.
Parsing complete

%vc_static_shell> report_read -family upf
-----
Tree Summary
-----
Severity      Family        Tag            Count
-----        -----        -----        -----
Warning       UPF          UPF_OSRSN     1
-----        -----        -----        -----
Total          1

%vc_static_shell> report_read -family upf -verbose
-----
Tree Summary
-----
Severity      Family        Tag            Count
-----        -----        -----        -----
Warning       UPF          UPF_OSRSN     1
-----        -----        -----        -----
Total          1
-----
UPF_OSRSN   (1 Warnings)
-----
Tag          : UPF_OSRSN
  MId        : 22
  Message    : Overriding the related power/ground
                ./top.upf:63 ./top.upf:59 Overriding the related power for design port
                'A' with 'top/srsn_vdd1'. Previous related supply specified was 'top/srsn_vdd'.
-----
```

GUI/Filter support is not yet available for this feature; it is planned to be provided in a future release.

## 3.4 Specifying Conditional UPF Command

VC LP provides the `test_conditional_upf` command to enable you to write UPF commands that requires to be put inside certain Tcl conditionals, such as:

- ❖ if {!\$::\_MAP\_PWR\_SWITCH\_DISABLE} {map\_power\_switch ...}
- ❖ if {\$::\_\_WELL\_BIAS} { create\_supply\_set -power -ground -pwell -nwell } else {
 create\_supply\_set -power -ground }
- ❖ if {\$::\_\_<IP>\_PST\_ENABLE || !\$\_\_IP\_PST\_DISABLE\_ALL} { create\_pst ... }

```

❖ if {!$::__IP_SRSN_DISABLE} { set_port_attributes -driver/receiver_supply }
❖ if {!$::__DISABLE_PWR_SWITCH} { create_power_switch ... }
❖ if {!$::__MAP_PWR_SWITCH_DISABLE} { map_power_switch ... }

```

You can specify multiple requirements for these conditionals using the `test_conditional_upf` command. If any specified UPF command does not fit in the required conditional, violations are reported as part of the `report_read` command. For example,

*File1.upf:29: [Error] CONDITIONAL\_MISSING: Command appears without required conditional. Command create\_power\_switch appears without required conditional  
{::DISABLE\_PWR\_SWITCH == 1 }*

As an UPF author, you should ensure that certain commands are protected by certain conditionals.

## Syntax

```

vc_static_shell> test_conditional_upf -help
Usage: test_conditional_upf      # Specify the conditional upf setting.
       -variable <{ {variable_name variable_value}+ }>
                           (Specify the varialbe name and value)
       [-command <command>]   (Specify commands to be checked for conditions:
                           Values: add_domain_elements,
                           add_parameter, add_port_state,
                           add_power_state, add_pst_state,
                           add_state_transition, add_supply_state,
                           apply_power_model, associate_supply_set,
                           begin_power_model, bind_checker,
                           connect_logic_net, connect_supply_net,
                           connect_supply_set,
                           create_composite_domain,
                           create_hdl2upf_vct, create_logic_net,
                           create_logic_port, create_power_domain,
                           create_power_state_group,
                           create_power_switch, create_pst,
                           create_supply_net, create_supply_port,
                           create_supply_set, create_upf2hdl_vct,
                           define_power_model,
                           describe_state_transition,
                           end_power_model, find_objects,
                           load_simstate_behavior, load_upf,
                           load_upf_protected, map_isolation_cell,
                           map_level_shifter_cell,
                           map_power_switch, map_repeater_cell,
                           map_retention_cell, merge_power_domains,
                           name_format, save_upf, set_correlated,
                           set_design_attributes, set_design_top,
                           set_domain_supply_net, set_equivalent,
                           set_isolation, set_isolation_control,
                           set_level_shifter,
                           set_partial_on_translation,
                           set_pin_related_supply,
                           set_port_attributes, set_power_switch,
                           set_repeater, set_retention,
                           set_retention_control,
                           set_retention_elements, set_scope,

```

```

        set_simstate_behavior, set_variation,
        sim_assertion_control,
        sim_corruption_control,
        sim_replay_control, upf_version,
        use_interface_cell)
[-required_supply_set_function <required supply set function>]
    (Specify required supply set function:
     Values: deepnwell, deeppwell, ground,
              nwell, power, pwell)
[-disallowed_supply_set_function <disallowed supply set function>]
    (Specify disallowed supply set function:
     Values: deepnwell, deeppwell, ground,
              nwell, power, pwell)
[-disallowed_port_attribute <disallowed port attribute>]
    (Specify disallowed port attribute:
     Values: clamp_value, driver_supply,
              feedthrough, is_analog, is_isolated,
              literal_supply, pg_type,
              receiver_supply, related_bias_ports,
              related_ground_port, related_power_port,
              sink_off_clamp, source_off_clamp,
              unconnected)
[-variable_logic <variable logic>]
    (Specify the relation of the variables:
     Values: and, or)

```

### 3.4.1 Use Model

The test\_conditional\_upf command should be used before the read\_upf command.

```

set search_path lib_dir
set link_library "demo_std_12v_pg.db"

test_conditional_upf -variable { {::VARA 1} } - command map_power_switch
test_conditional_upf -variable { {::VARB 1} } -disallowed_supply_set_function { power
ground}
test_conditional_upf -variable { {::VARC 1} } -required _supply_set_function { pwell
nwell}
test_conditional_upf -variable { {::VARD 1} } -disallowed_port_attribute driver_supply

read_file top.v -top top -netlist
read_upf top.upf

report_readViolation
check_lp
report_lp

```

## 3.5 Reading Power Intent

After a successful design read, you can provide the required power intent in the form of UPF.

### 3.5.1 The read\_upf Command

The `read_upf` command reads a UPF file and applies it to the design. The UPF is checked for syntax and it is ensured that all the design objects mentioned in the UPF file exist in the design. Some additional semantic checks are also performed.

Any messages from the tool are printed to the screen. In case a large number of messages appear, it may be helpful to use the standard redirect syntax.

#### Syntax

```
vc_static_shell> read_upf -help
Usage: read_upf      # Read in UPF file
       [-supplemental <supf_file_name>]
                           (Supplemental UPF file to read)
       [-strict_check true|false]
                           (Perform exact name search and error check)
       [-quietBbox true|false]
                           (All the warning messages during black-box object
processing would be ignored)
       [-quietAbs]           (All the warning messages during abstracted block
processing would be ignored)
       [-no_design]          (The upf reading will be done in design independent mode)
       [-no_trace_supplies] (The upf reading will not do supply tracing)
       [-j <num threads>]   (Number of threads to be used in parallel mode)
       upf_filename ...     (UPF file to be read in)
```

#### Example:

```
%vc_static_shell>read_upf myfile.upf
%vc_static_shell>redirect myupf.log {read_upf myfile.upf}
```

At this stage, if there are any syntax errors reported by the tool on the UPF files, you can use the `remove_upf` feature to remove the loaded UPF, correct the reported issues and source it back again using the `read_upf` command.

#### Example

By default, `read_upf` runs in serial mode. To enable multi-threading, provide the `-j` option with thread count the `j` value during the `read_upf` stage.

```
read_upf -j 8 test.upf
```

The actual number of threads that will run during execution depends on thread availability. The debug prints can be viewed using the following command:

```
set_app_var enable_debug true
DebugMsg -add "MtPgTracer"
```



#### Note

The VC\_LP\_MT\_NT license would be checked-out during the execution of `read_upf -j` option.

#### Notes:

- ❖ If you are in the golden UPF flow (RTL level UPF being used for netlist/PG netlist designs), the design transformation information should be provided in a supplemental UPF file (using the `read_upf -supplemental` command). For more information on the golden UPF flow, see section "[Golden UPF Flow](#)".
- ❖ VC LP does not allow UPF to be read in without successful design read.

- ❖ The `load_upf` and `read_upf` command can be used interchangeably.
- ❖ UPF must be read in the form of a file. VC LP does not allow UPF Tcl commands such as `create_power_domain` directly on the `vc_static_shell`.

### 3.5.1.1 Support for Handling Macro Modules

The `upf_macro_from_liberty` application variable is introduced for UPF parser to handle macro module. For each module with `is_macro:true` found in the design, the UPF parser does not treat the module as if the attribute `is_hard_macro` has been set in the UPF.

By setting `upf_macro_from_liberty` to false, VC LP returns to previous behavior to handle it as black box.

The application variable `upf_macro_from_liberty` is set to true by default. Therefore, by default, the following parser warnings are reported for macro modules for different scenarios.

- ◆ *[Warning] UPF\_WITHIN\_MACRO\_ACCESS: Macro object accessed from outside Upf commands Trying to access inside a macro scope*
- ◆ *[Warning] UPF\_MACRO MODIFY\_FROM\_OUTSIDE: Macro object modify from outside Upf commands trying to define power intent within the child side of the hard macro from outside scope.*
- ◆ *[Warning] UPF\_STRATEGY\_MACRO\_WRONG\_LOCATION: Strategy defined at wrong location Strategy with location 'self' defined across the hard macro instance boundary is not allowed.*

### 3.5.2 The `remove_upf` Command

The `remove_upf` command enables you to load another UPF for the same design. Hence, you can use combination of `read_upf` and `check_lp -stage upf` with the `remove_upf` to incrementally refine the UPF during UPF development stage, without needing to quit the session to reload the design.

For more information on the `check_lp` command, see section “[Running VC LP Checks](#)”.

#### Syntax

```
%vc_static_shell> remove_upf -help
Usage: remove_upf      # Removes the UPF constraints from the design
```

#### Use Models

The following example shows the typical usage of the `remove_upf` command.

##### `remove_upf` before `check_lp` command :

```
=====
%vc_static_shell> read_file          // Design file read
1
%vc_static_shell> read_upf top.upf    // If any Parsing error/warning reports
at this stage, the existing UPF data can be removed and Modified UPF can be parsed
top.upf:65: [Error] UPF_PDDNE: Power Domain does not exist
Value 'top/PD_CORE_2' passed as argument to -domain option of 'set_isolation' is
not a power domain.
Power Intent parsing failed
Error: 0
%vc_static_shell> remove_upf
1
%vc_static_shell> read_upf top.upf
1
%vc_static_shell> check_lp -stage upf
```

**remove\_upf after check\_lp command:**

```
=====
%vc_static_shell> read_file                                // Design file read
1
%vc_static_shell> read_upf top.upf                         // UPF read
1
%vc_static_shell> check_lp -stage upf                     // User can also remove UPF after
check_lp
1
%vc_static_shell>report_violations -app LP
1
%vc_static_shell> remove_upf
1
%vc_static_shell> read_upf top.upf
1
%vc_static_shell>check_lp -stage upf                     //Now checks done based on Modified UPF
1
```

**remove\_upf in restore session:**

```
=====
%vc_static_shell> read_file                                // Design file read
1
%vc_static_shell> read_upf top.upf                         // UPF read
1
%vc_static_shell> check_lp -stage upf                     // read_upf ,check_lp performed in in
first session
1
%vc_static_shell>save_session                            // save the session
1
%vc_static_shell -restore
%vc_static_shell> remove_upf                           // Now in restore session User can also
remove Upf
1
%vc_static_shell> read_upf top.upf
1
%vc_static_shell>check_lp -stage upf                     //Now checks done based on Modified UPF
```

**3.5.3 The get\_upf\_scope Command**

The `get_upf_scope` command enables you to identify which UPFs are loaded at which scope.

**Syntax**

```
vc_static_shell> get_upf_scope -help
Usage: get_upf_scope      # This command returns upf files or upf scopes
       [-upf <file>]        (Specifies UPF file name)
       [-scope <scope>]       (Specifies UPF power scope name)
```

**Example**

Consider the following hierarchical UPF:

*Top.upf*:

```
load_upf ABC.upf -scope ABC
load_upf BCD.upf -scope BCD
load_upf BCD.upf -scope CDE
```

Table 3-3 provides the commands and the results for the mentioned hierarchical UPF.

**Table 3-3    Tcl Command Results**

get_upf_scope -upf ABC.upf	{"ABC"}
get_upf_scope -upf BCD.upf	{"BCD", "CDE"}
get_upf_scope -scope ABC	{"ABC.upf"}
get_upf_scope -scope BCD	{"BCD.upf"}

### 3.5.4    The report\_violations -app UPF Command

The report\_violations -app UPF command generates the key power intent information after reading the UPF.

#### Syntax

```
%vc_static_shell> report_violations -app UPF -help
Usage: report_violations -app UPF # Prints counts of interesting UPF objects
```

#### Example

```
%vc_static_shell> report_violations -app UPF
Design top : top
Isolation instances : 1
Level shifter instances : 0
Retention instances : 0
Power switch instances : 0
Multirail macro instances : 0
Total instances : 2
Crossovers : 4
Merged power states : 2
```

#### 3.5.4.1    Support for Wild Cards Expansion Report

When you have wild cards specified in the UPF, VC LP enables you to get a report of the UPF file with all the wild cards expanded. The UPF file with the wild cards expanded is saved in the *vcst\_rtdb/reports/upf\_expanded.upf* file. You can directly reuse the expanded UPF file in the design to get the same results instead of the original UPF.



**Note**  
Currently, the expanded UPF file is not recommended to be used as the golden UPF file for a design. The expanded UPF file is provided as a debug methodology.

To save the UPF file with all the wild cards expanded, use the `enable_lp_dump_debug_reports dump_decompile_upf` command.

```
%vc_static_shell> enable_lp_dump_debug_reports dump_decompile_upf
```

#### Example

- ❖ The following is an example UPF command with wild card characters:

```
create_power_domain PD1 -elements {A*}
create_supply_port VDD
```

```

create_supply_port VDD1
create_supply_port VDD2
create_supply_net VDD
create_supply_net VDD1
create_supply_net VDD2
connect_supply_net VDD -ports {V*}

```

The *vcst\_rtdb/reports/upf\_expanded.upf* file contains the following output with the wild card characters expanded:

```

create_power_domain PD1 -elements { A A_1 }
create_supply_port VDD
create_supply_port VDD1
create_supply_port VDD2
create_supply_net VDD
create_supply_net VDD1
create_supply_net VDD2
connect_supply_net VDD -ports { VDD VDD1 VDD2 }

```

- ❖ If a wild card is used in a option that is not supported, the wild card for that option is not expanded and appears as it is in the expanded UPF file.

Consider the following UPF snippet in which the `-instance` is not supported:

```

set_level_shifter LS1 -domain LOW
    -applies_to outputs
    -rule low_to_high
    -location self
    -elements {h1/out* h2/out*}
    -exclude_elements {h1/out2}
    -threshold 0.2
    -source SS_TOP
    -sink SS_LOW
    -instance {h*}

```

The expanded UPF contains the following output:

```

set_level_shifter LS1 -domain LOW
-applies_to outputs
-rule low_to_high
-location self -elements { h1/out1 h1/out2 h1/out3 h2/out1 h2/out2}
-exclude_elements { h1/out2 }
-threshold 0.2
-source SS_TOP
-sink SS_LOW
-instance { h* }

```

- ❖ The Tcl commands specified in the UPF file is not saved in the expanded UPF file. However, if the tcl commands are related to UPF file generation, the resolved UPF is saved in the expanded UPF file.

### Example 1

Consider the following UPF:

```

for {set i 0} {$i < 5} {incr i} {
    set name [format "v%02d" $i]
    create_supply_port $name
    create_supply_net $name
    connect_supply_net $name -ports $name
    for {set j 1} {$j < 6} {incr j} { add_port_state $name -state "$$j 1.$j" }
    lappend supplies $name
}

```

The following details are saved in the expanded UPF:

```
create_supply_port v00
create_supply_net v00
connect_supply_net v00 -ports { v00 }
add_port_state v00 -state { s1 1.1 }
add_port_state v00 -state { s2 1.2 }
add_port_state v00 -state { s3 1.3 }
add_port_state v00 -state { s4 1.4 }
add_port_state v00 -state { s5 1.5 }
....
```

### Example 2

Consider the following UPF:

```
puts "[find_objects . -pattern {ar[*]} -object_type inst ]"
puts "UPF_EXPANSION_CHECK"
```

The following details are saved in the expanded UPF:

```
find_objects . -pattern ar[*] -object_type inst
# ar[0] ar[1] ar[2] ar[3]
```

#### 3.5.4.2 Saving Resolved UPF Files Without Unavailable Design Elements

Use the `lp_dump_resolved_upf` application variable to save resolved UPF files by excluding the unavailable design element references in the UPF. The resolved expanded UPF is saved in `vcst_rtdb/lpdb/debug_reports/upf_expanded_resolved.upf`.

For example, consider the following UPF where `core_unavail` is an unavailable design element. It may be unavailable for any reason such as not being defined, SAM abstraction, black box flow and so on.

```
create_power_domain PD1 -elements
{ core1 core_unavail }
```

When the `set_app_var lp_dump_resolved_upf true`, the `vcst_rtdb/lpdb/debug_reports/upf_expanded_resolved.upf` file would contain the following

```
create_power_domain PD1 -elements
{ core1 }
```

#### 3.5.4.3 Enabling `upf_lrm_option_defaults` Application Variable

Use the `upf_lrm_option_defaults` application variable to define the default values of the some options used in UPF commands.

When `upf_lrm_option_defaults` is set to 3.1 (`set_app_var upf_lrm_option_defaults 3.1`), the default enum values for the following options of `set_isolation`/`set_level_shifter`/`set_repeater` are updated:

- ❖ `applies_to_boundary` both
- ❖ `diff_supply_only` TRUE
- ❖ `applies_to` both

### 3.5.5 Support Boolean Expressions in All UPF Commands

The XOR (^) and not equal to (!=) operators are supported for following commands.

UPF Command and Options	Examples
<pre>create_power_switch     -on_state {state_name input_supply_port     {Boolean_expr}}     -off_state {state_name {Boolean_expr}}     -on_partial_state {state_name     input_supply_port {Boolean_expr}}     -error_state {state_name {Boolean_expr}}     -ack_port {net_name [Boolean_expr]}</pre>	<pre>create_power_switch sw3 \     -domain PD_TOP \     .....     -on_state {s1 vin {(A==0) &amp;&amp; (C!=0)}} \ </pre>
<pre>set_retention     -save_condition {Boolean Expr}     -restore_condition {Boolean Expr}     -retention_condition {Boolean Expr}</pre>	<pre>set_retention ret_PD2 \     -domain PD2 \     .....     -retention_condition { (!retain1        retain2) ^ (retain3)} \</pre>
<pre>add_power_state -supply -logic_expr</pre>	<pre>add_power_state -supply SS1 -state ON {-     logic_expr {(net1 != net2) &amp;&amp; net3} }</pre>

The support for boolean UPF expressions are enhanced to align all the Synopsys tools under a common specifications for boolean expressions support.

### 3.5.6 Support for Escape Characters

The `enable_upf_escape_style` application variable supports the usage of escape characters. By default, the `enable_upf_escape_style` application variable is set to false.

When the `enable_upf_escape_style` application variable is set to true, the escape characters inserted in the design are matched with the netlist and UPF definitions. All escape characters including the special character mentioned in LRM "[.:\*?]" is matched with the netlist and UPF definitions. The special characters such as "\*" is not treated as wild card. However, it is recommended to use curly braces when defining the name in the UPF.

The `enable_upf_escape_style` application variable usage is applicable for Verilog and System Verilog (SV) language design files.

#### Example

When the instance name in the `buf_i1\|a\|b\|c` netlist has escape characters, and the `enable_upf_escape_style` application variable is set to true, VC LP identifies and matches the name with the instance name in netlist.

- ❖ `connect_supply_net VDD -ports { {buf_i1\\a\\b\\c/VDD} }`
- ❖ `connect_supply_net VDD -ports { {\buf_i1/a/b/c /VDD} }`
- ❖ `connect_supply_net VDD -ports { {\buf_i1/a/b/c /VDD} }`
- ❖ `connect_supply_net VDD1 -ports {uSUB/uSUB1/and_i1\\temp/VDD}`

- ❖ connect\_supply\_net VDD1 -ports {uSUB/uSUB1/and\_i1\\*/VDD}
- Here \* is treated as a special character, instead of a wild card character.

### 3.5.7 Support for analyze\_domain\_crossings Command

In the DIUC flow, a new method is introduced to identify the level shifter and isolation requirements between power domains through `analyze_domain_crossings` Tcl command. This command will output whether there is an isolation/level shifter requirement between two domains and the type of level shifter requirement if any.

#### Syntax

```
Usage: analyze_domain_crossings      # Reports ISO/LS requirements between domains
       [-format <format>]          (The format in which the output is expected:
                                      Values: csv, txt, upf)
       [-iso_only]                 (Report ISO requirements only)
       [-ls_only]                  (Report LS requirements only)
       [-source <source>]          (Collection of Source Power Domains to be considered)
       [-sink <sink>]              (Collection of Sink Power Domains to be considered)
       [-supplies]                 (Enables dumping of ISO/LS requirement among supplies)
```

Where:

- ❖ [-format <format>]: Specify the format in which the output is expected (Values: csv, txt, upf)
- ◆ txt: With this option, the output with the requirements is reported in a simple text base. This is the default format.

#### Example

```
analyze_domain_crossings -format txt
Source      Sink      Isolation? LevelShifter?
-----      -----      -----
PD1        top       Yes        LH
top        PD1       No         HL
```

- ◆ upf: With this option, the output with the requirements is reported in form of incomplete upf strategies populated with available information, so that you can use the same by adding other specific information.

#### Example

```
analyze_domain_crossings -format upf
set_isolation -domain PD1 -source PD1 -sink top isolation_supply_set -
location -isolation_sense -isolation_signal
set_level_shifter -domain PD1 -source PD1 -sink top -rule LH -threshold
set_level_shifter -domain top -source top -sink PD1 -rule HL -threshold
```

- ◆ csv: With this option, the command produces a comma separated value output suitable for importing into Excel.

```
analyze_domain_crossings
Source      Sink      Isolation? LevelShifter?
-----      -----      -----
PD1        top       Yes        LH
top        PD1       No         HL
```

- ❖ [-iso\_only]: Specify this option to get a report of the ISO requirements only.
- ❖ [-ls\_only]: Specify this option to get a report of the LS requirements only.
- ❖ [-source <source>]: Specify the collection of Source Power Domains to be considered.

When you provide power domains to -src, -sink switches, they should be given in form of collections

```
analyze_domain_crossings -format txt -source [ get_power_domains PD1 ] -sink [ get_power_domains top ]
```

- ❖ [-sink <sink>]: Specify the collection of sink power domains to be considered.

In case any -source, -sink is not specified in the command, it will output results for all the combinations of all the available domains.

- ❖ [-supplies]: Specify this option to save the ISO/LS requirement among the domains based on power/ground and PST states in text and CSV formats.

The following is an example output when the -supplies option is specified:

Supplies	VDD	VDD1	VDD2	VDD3
VDD	-	LS (LH)	LS (LH)	LS (LH)
VDD1	LS (HL)	-	LS (BOTH)	ISO
VDD2	LS (HL)	LS (BOTH)	-	LS (BOTH)
VDD3	LS (HL)	ISO	LS (BOTH)	-

Supply Net	BC OV	WC OV	Supply States
VDD	0.90V	0.90V	on
VDD1	1.10V	1.10V	on, off
VDD2	1.20V	1.00V	on
VDD3	1.10V	1.10V	on, off

The json dump support is available for DIUC flow. The following json logs are saved in the DIUC flow.

- ❖ policy
- ❖ rootsupply { limited support for upf supplies }
- ❖ upfdatamodel

### Example

- ❖ There is an isolation requirement and a level shifter requirement from domain PD1 to top depending on the supply states of their primary supplies.

Consider the following UPF:

top --> primary supply (VDD1)

PD1 --> primary supply (VDD)

```
create_pst stble -supplies {VDD VDD1 VSS}
add_pst_state PD1off -pst stble -state {OFF ON_0_9A ON}
add_pst_state PD1diff -pst stble -state {ON_0_9 ON_1 ON}
```

Use the analyze\_domain\_crossings tcl command to get a report of the ISO and LS requirements:

Source	Sink	Isolation?	LevelShifter?
PD1	top	Yes	LH
top	PD1	No	HL

### 3.5.8 Support for Plato Parser

Starting with O-2018.09, Plato is the made the default parser in VC LP. There are no differences in QOR or performance compared to pixl2 parser in Plato. Plato is made the default parser to introduce common parsers across Synopsys tools.

### 3.5.8.1 Backward compatibility

You can switch back to pixl2 parser by setting the following application variable to false:

```
set_app_var enable_plato_db_population false
```

### 3.5.8.2 Parser Messages Introduced for Plato Parser

The following parser messages are introduced for the plato parser.

**Table 3-4** **Plato Parser Messages**

Message	Description
MODEL_NOT_FOUND_WARN	When the < model name> specified with command option 'set_design_attributes-models'cannotberesolved
MODEL_NOT_FOUND_ERROR	When the Model specified in command option set_design_attribute-modulecannotbefoundinthedesign.
UPF_ISOLATION_APPLIES_TO_SPECIFIED	Tying to specify '-applies_to' of an Isolation Strategy which already has already has the option specified
UPF_ISOLATION_LOCATION_SPECIFIED	Tying to specify '-location' of an Isolation Strategy which already has already has the option specified
UPF_STRATEGY_SIGNAL_SPECIFIED	Tying to specify '-isolation_signal' of an Isolation Strategy which already has already has the option specified
UPF_SUPPLY_PORT_NO_HICONN	where a supply port is missing connection to a upper level hierarchy
UPF_SUPPLY_PORT_NO_LOWCONN	where a supply port is missing connection to a lower level hierarchy
UPF_GLOBAL_OBJECT_NOT_FOUND_WARN	Where the Global object <object> specified with command option'connect_supply_net-vct'cannot be resolved.
UPF_ISO_HETEROGENEOUS_FANOUTS	When a path based Isolation stategy is specified for heterogeneous fanout
UPF_ATTR_MULTIPLE_WARN	Attribute 'correlated_supply_group' has already been specified in higher level scope
UPF_SSET_NET_PD_DEPENDENT	When a domain dependent supply net is defined as a supply set function

### 3.5.8.3 Messaging Differences Between Plato and pixl2

The following are differences in the parser messages reported in the Plato and Pixl2 parser.

**Table 3-5** **Messaging differences in Plato**

Scenario	Plato behavior	Pixl2 behavior
create_supply_set with hierarchical name	[Error] UPF_INVALID_HIERARCHICAL_REFERENCE	[Error] UPF_INVALID_ID

Scenario	Plato behavior	PixI2 behavior
Erroneous -supply in create_power_domain	[Warning] <i>UPF_OPT_IGNORED</i>	No warning
Supply nets that do not have a port created with the same name	[Warning] <i>UPF_SUPPLY_PORT_IMPLPLICIT_CONNECTION</i>	No warning
Empty element list in create_power_domain	[Error] <i>UPF_OBJECTS_NOT_FOUND</i>	[Error] <i>UPF_EMPTY_OBJ_LIST</i>
Unavailable elements specified in create_power_switch -control_port option	Plato ignores the create_power_switch command at the 1st error. Plato gives <i>UPF_OBJECTS_NOT_FOUND</i> errors for the UPF commands which refer to the supply ports of the ignored psw strategy.	PixI2 evaluates the create_power_swich command further, and errors out for other fault such as referring to the unavailable port in -on_state ( <i>UPF_PSW_EXPR_CONTROL_PORT</i> ).
set_related_supply_net-object_list with an unavailable object	[Warning] <i>UPF_OBJECT_NOT_FOUND_WARN</i>	[Error] <i>UPF_OBJECT_NOT_FOUND_ERROR</i>
enable_state_propagation_in_add_power_state attribute is defined after add_power_state	[Error] <i>UPF_ATTR_STATES_ALREADY_CREATED</i>	Error] <i>UPF_SET_DESIGN_ATTRIBUTE_ILLEGAL</i>
Invalid attribute values in UPF_dont_touch design attribute	[Error] <i>TCL_INV_ATTR_VAL</i> [Warning] <i>UPF_CMD_IGNORED</i> Plato further flags <i>MODEL_NOT_FOUND_WARN</i> for other set_design_attributes with same element	[Error] <i>TCL_INV_ATTR_VAL</i>
Out of range voltage values in add_port_state	[Error] <i>UPF_VOLTAGE_VALUE_OUTOFRANGE</i>	[Error] <i>UPF_SPORT_STATE_INVALID_VALUE</i>
available_supply_sets in report_power_domain	Includes default_isolation and default_retention supplies	Does not include default_isolation and default_retention supplies
Violation debug fields for equivalent net/supply set	Either one of the equivalent supplies may be reported in the debug fields	Supply defined first will be reported in the debug fields
Implicit bias nets of a supply set	No violation	[Warning] <i>UPF_SUPPLY_UNUSED</i>
State State specified multiple times for supply port UPFSupply	No violation	[Warning] <i>UPF_STATE_DUPLICATE</i>
For unavailable objects in value argument in -attribute switch in set_design_attributes	[ERROR] <i>UPF_ATTR_OBJECT_NOT_FOUND</i>	[ERROR] <i>UPF_ATTR_OBJECT_NOT_FOUND</i> [ERROR] <i>UPF_OBJECT_NOT_FOUND_ERROR</i>

Scenario	Plato behavior	PixI2 behavior
Global design attributes defined with -elements switch	[Warning] <i>UPF_ATTR_INVALID_OBJECT</i>	[Error] <i>UPF_ATTR_NOT_FOR_TOP_SCOPE</i>
sim_assertion_control, sim_corruption_control and sim_replay_control commands	[Info] <i>UPF_COMMAND_IGNORED</i> Upf command is ignored. The options of the command are not processed.	[Info] <i>UPF_CMD_SIM_IGNORED</i> The options of the command are processed and flags following violations for different scenarios.  [Error] <i>TCL_OPTION_UNKNOWN</i> for unknown options.  [Error] <i>TCL_OPTION_EXTRA</i> for extra positional options  [Error] <i>TCL_OPTION_MANDATORY</i> for missing mandatory options  [Error] <i>TCL_OPTVAL_INVALIDONEOF</i> for invalid arguments in options  [Warning] <i>UPF_OBJECT_NOT_FOUND_WARN</i> for unavailable objects in options

### 3.5.9 Support for Comparing UPF Data Models

UPF is the key database for any low power verification. Any inconsistency, incompleteness, error in the UPF can lead to incorrect, incomplete, inconsistent low power verification results. VC LP saves its UPF database in JSON format to provide more observability and debug ability. These saved UPF databases can be monitored and compared against any golden/reference database to ensure that the UPF used for low power checks remains correct, consistent, and complete.

In many customers flow, the verification/CAD team need to ensure that the UPFs used for two different flows (for example, verification and synthesis) are consistent to each other. In these flows, the design remains the same.

For such flows, VC LP enables you to compare two different UPF databases, and the resulting difference are presented in terms of VC LP violations.

To compare the UPF databases, use the `compare_upf` Tcl Command.



**Note**  
This feature is available with Beta Support under the VC-LP-ULTRA license. For more information on this feature, contact [mvsupport@synopsys.com](mailto:mvsupport@synopsys.com)

#### 3.5.9.1 Comparing UPF Databases

Use the `compare_upf` command to compare UPFs. The `compare_upf` should be used after reading the design file.

##### Syntax

```
Usage: compare_upf      # compares original and current UPF
       [-component <list of types>]
           (cell types:
            Values: csn, iso, ls, pd, pst, psw, ret,
            spa_srsn, ss)
```

```
-current <current UPF name>
          (current UPF file)
-original <original UPF name>
          (original UPF file)
```

### Example

```
compare_upf -current my_current_upf -original original_upf
```

VC LP performs the following operations when `compare_upf` Tcl command is used:

1. VC LP executes the `read_upf original_upf` command, and dumps the database into JSON format.
2. VC LP executes `remove_upf original_upf`.
3. VC LP executes the `read_upf my_current_upf` command, and dumps the database into JSON format.
4. After performing comparison between above two databases, VC LP reports the violation with the `report_violations -app LP` command.



### Note

For the BBOX scoped UPF, this feature will not work.

#### 3.5.9.2 Use Model

```
set_app_var search_path "./ pgdb /...."
set_app_var link_library " aaa.db aad.db abb.db asdsd.db asds.db "
read_file -format verilog -top top -netlist " DIFF_ISO_LOCATION.v "
compare_upf -component iso -current DIFF_ISO_LOCATION.upf -original golden.upf
report_violations -app LP -verbose -file report_lp.txt
```

Where,

The following is the content of `golden.upf` UPF file:

```
set_isolation ISO_1 -domain PD1 -elements {CORE1/out1} -clamp_value 0 -isolation_signal
isoen -isolation_supply_set TOP.primary -location self
```

The following is the content of `DIFF_ISO_LOCATION.upf` UPF file:

```
set_isolation ISO_1 -domain PD1 -elements {CORE1/out1} -clamp_value 0 -isolation_signal
isoen -isolation_supply_set TOP.primary -location parent
```

### Output

The following violation is reported as there is a difference in the location mismatch (parent and self) in the ISO strategy:

Violation Snippet

```
Tag : DIFF_ISO_LOCATION
Description : Isolation instance [Strategy] location value mismatch between
original and current UPF
Strategy : PD1/ISO_1
DiffType : MISMATCH
NewAttrValue
  AttributeValue : parent
  FileName : DIFF_ISO_LOCATION.upf
  LineNumber : 50
OldAttrValue
  AttributeValue : self
```

```
FileName      : golden.upf
LineNumber   : 46
```

### 3.5.9.3 Violations Introduced

The following violations are introduced for this support. All the violations are reported at the check\_lp - stage UPF, Family: UPFDiff, with severity *warning*.



For more details on these violations, refer to the man pages.

#### Violation tags related to isolation strategy

- ❖ DIFF\_ISO\_CLAMP
- ❖ DIFF\_ISO\_RESOLVED
- ❖ DIFF\_ISO\_GROUND
- ❖ DIFF\_ISO\_MAP
- ❖ DIFF\_ISO\_LOCATION
- ❖ DIFF\_ISO\_NOISO
- ❖ DIFF\_ISO\_EXIST
- ❖ DIFF\_ISO\_POWER
- ❖ DIFF\_ISO\_SINK
- ❖ DIFF\_ISO\_SOURCE
- ❖ DIFF\_ISO\_SENSE
- ❖ DIFF\_ISO\_APPLIES
- ❖ DIFF\_ISO\_SIGNAL

#### Violation tags related to level-shifter strategy

- ❖ DIFF\_LS\_ELEMENT
- ❖ DIFF\_LS\_FORCE
- ❖ DIFF\_LS\_LOCATION
- ❖ DIFF\_LS\_NOSSHIFT
- ❖ DIFF\_LS\_EXIST
- ❖ DIFF\_LS\_RULE
- ❖ DIFF\_LS\_THRESHOLD
- ❖ DIFF\_LS\_MAP

#### Violation tags related to power-switch strategy

- ❖ DIFF\_PSW\_ACK
- ❖ DIFF\_PSW\_CONTROL
- ❖ DIFF\_PSW\_INPUT
- ❖ DIFF\_PSW\_MAP
- ❖ DIFF\_PSW\_EXIST

- ❖ DIFF\_PSW\_OFFSTATE
- ❖ DIFF\_PSW\_ONSTATE
- ❖ DIFF\_PSW\_OUTPUT
- ❖ DIFF\_PSW\_DOMAIN
- ❖ DIFF\_PSW\_ONSTATEEXPR
- ❖ DIFF\_PSW\_ONPARTSTATEEXPR
- ❖ DIFF\_PSW\_ERRSTATE
- ❖ DIFF\_PSW\_ERRSTATEEXPR
- ❖ DIFF\_PSW\_OFFSTATEEXPR
- ❖ DIFF\_PSW\_ACKEXPR

**Violation tags related to retention strategy**

- ❖ DIFF\_RET\_ELEMENT
- ❖ DIFF\_RET\_GROUND
- ❖ DIFF\_RET\_MAP
- ❖ DIFF\_RET\_NORET
- ❖ DIFF\_RET\_EXIST
- ❖ DIFF\_RET\_POWER
- ❖ DIFF\_RET\_PRIMARY
- ❖ DIFF\_RET\_RESTORE
- ❖ DIFF\_RET\_SAVE
- ❖ DIFF\_RET\_DOMAIN
- ❖ DIFF\_RET\_SAVENSE
- ❖ DIFF\_RET\_RESTORENSE

**Violation tags related to SUPPLY\_SET**

- ❖ DIFF\_SET\_EXIST
- ❖ DIFF\_SET\_GROUND
- ❖ DIFF\_SET\_POWER
- ❖ DIFF\_SET\_NWELL
- ❖ DIFF\_SET\_PWELL

**Violations related to SPA\_SRSN**

- ❖ DIFF\_SPA\_EXIST
- ❖ DIFF\_SPA\_DRIVER
- ❖ DIFF\_SPA\_RECEIVER
- ❖ DIFF\_SRSN\_EXIST
- ❖ DIFF\_SRSN\_POWER
- ❖ DIFF\_SRSN\_GROUND

### Violations related to add\_power\_state

- ❖ DIFF\_POWERSTATE\_EXIST
- ❖ DIFF\_POWERSTATE\_SUPPLYEXPR
- ❖ DIFF\_POWERSTATE\_LOGICEXPR
- ❖ DIFF\_POWERGROUP\_EXIST
- ❖ DIFF\_POWERGROUP\_STATE
- ❖ DIFF\_POWERGROUP\_LOGICEXPR

### Violations related to PST

- ❖ DIFF\_PST\_EXIST
- ❖ DIFF\_PST\_STATE
- ❖ DIFF\_PST\_SUPPLY
- ❖ DIFF\_SUPPLY\_STATE
- ❖ DIFF\_SUPPLYSTATE\_ATTR

### Other violation tags

- ❖ DIFF\_DOMAIN\_ELEMENT
- ❖ DIFF\_DOMAIN\_EXIST
- ❖ DIFF\_DOMAIN\_EXCLUDE
- ❖ DIFF\_NET\_EXIST
- ❖ DIFF\_PORT\_EXIST
- ❖ DIFF\_CSN\_MISSING
- ❖ DIFF\_SYSTEM\_STATE

### Limitations

Currently, VC LP is not reporting the DIFF\_LS\_DOMAIN, DIFF\_RET\_DOMAIN and DIFF\_CSN\_MISSING violations.

### 3.5.10 Support for Design Independent UPF Checker Flow

The Design Independent Upf Checker (DIUC) flow enables you to read and resolve UPF independent of the design. In this flow, the following are performed without the involvement of a design:

- ❖ UPF parsing checks
- ❖ UPF semantics checks
- ❖ UPF supply analysis and PST checks
- ❖ UPF LP consistency checks
- ❖ Debugging capability for UPF features in Verdi
- ❖ Verdi support on UPF consistency LP checks



#### Note

In the DIUC flow, the hierarchical design references are always assumed valid. Violations for invalid design references are not reported. The debug commands that refer to design objects are not supported in the DIUC flow.

## Use Model

The DIUC flow can be enabled using the `-no_design` switch in the `read_upf/load_upf` Tcl command.

### Syntax

```
read_upf xxxx.upf -no_design
report_read_violations -verbose -file <FILE>
check_lp -stage upf
report_lp -verbose -file <FILE>
```

### 3.5.10.1 Parser Messages and Violations Reported in the DIUC Flow

[Table 3-6](#) lists the parser level violations that are reported in the DIUC flow.

**Table 3-6 Supported Parser Level Violations in DIUC Flow**

Parser Violations	Parser Violations
TCL_OPT_ATLEAST_ONE_OF	TCL_OPT_ATMOST_ONE_OF
TCL_OPT_ATTR_UNSUPPORTED	TCL_OPT_DEPENDS
TCL_OPT_EMPTY	TCL_OPT_EMPTY_LIST
TCL_OPT_EXACTLY_ONE_OF	TCL_OPT_IMPROPER_ARGS
TCL_OPT_INVALIDONEOF	TCL_OPT_NOT_TOGETHER
TCL_OPT_NYI	TCL_OPT_TOGETHER
TCL_OPT_VAL_UNSUPPORTED	UPF_ATTR_BIAS_SUPPLIES_CREATED
UPF_ATTR_MULTIPLE	UPF_ATTR_OBJECT_NOT_FOUND
UPF_CMD_IGNORED	UPF_EXPR_INTERVAL_FUNCTION
UPF_EXPR_NEGATIVE_NUMBER	UPF_EXPR_SYNTAX_ERROR
UPF_FILE_NOT_FOUND	UPF_FILE_NOT_SPECIFIED
UPF_FILE_PARSING	UPF_GROUP_ALREADY_DEFINED
UPF_GROUP_NO_UPDATE	UPF_GROUP_STATE_OPTION_NOT_ALLOWED
UPF_GROUP_STATE_UPDATE_LEGAL	UPF_GROUP_UPDATE_FIRST_USE
UPF_INVALID_HIERARCHICAL_REFERENCE	UPF_INVALID_ID
UPF_ISOLATION_APPLIES_TO_SPECIFIED	UPF_ISOLATION_LOCATION_SPECIFIED
UPF_ISOLATION_SENSE_SIZE_MISMATCH	UPF_LIST_SYNTAX_ERROR
UPF_LOGIC_PORT_ALREADY_CONNECTED	UPF_LOGIC_PORT_CONNECTION_OVERRIDE
UPF_PARTIAL_ON_UNSUPPORTED_TOOL	UPF_PD_DEFINED
UPF_PD_EXTRA_SUPPLIES_VALUE	UPF_PD_PRIMARY_SET
UPF_PD_STRATEGY_ALREADY_DEFINED	UPF_PD_STRATEGY_NOT_DEFINED

Parser Violations	Parser Violations
UPF_PD_UPDATE_NOT_DEFINED	UPF_PST_DUPLICATE_STATE
UPF_PST_DUPLICATE_SUPPLY	UPF_PST_STATE_DEFINED
UPF_PST_STATE_INVALID_VALUES	UPF_PST_STATE_NOT_FOUND
UPF_PSW_EXPR_CONTROL_PORT	UPF_PSW_INPUT_OUTPUT_SAME_CONNECTI ON
UPF_PSW_PORT_NOT_FOUND	UPF_PSW_STATE_DEFINED
UPF_EQUIVALENT_FUNC_ONLY_STATES_CONFLICT	UPF_SET_EQUIVALENT_PSW_SUPPLY_PORTS
UPF_SET_EQUIVALENT_SUPPLY_DOMAIN_DEPENDENT	UPF_SET_EQUIVALENT_SUPPLY_FUNC_MISM ATCH
UPF_SET_EQUIVALENT_SUPPLY_TYPE_MISMATCH	UPF_SNNet_ALREADY_DEFINED
UPF_SNNet_INCORRECT_REUSE	UPF_SNNet_REUSE_RESOLUTION
UPF_SNf_ALREADY_ASSOCIATED	UPF_SPORT_STATE_INVALID_VALUE
UPF_SPORT_STATE_OFF	UPF_SSET_ALREADY_ASSOCIATED
UPF_SSET_ALREADY_DEFINED	UPF_SSET_ASSOCIATION_LOOP
UPF_SSET_FUNCTION_ASSOCIATED	UPF_SSET_FUNCTION_MISMATCH
UPF_SSET_FUNCTION_NOT_FOUND	UPF_SSET_INCORRECT_UPDATE
UPF_SSET_INVALID_ASSOCIATION	UPF_SSET_INVALID_SELF_ASSOCIATION
UPF_SSET_STATE_INVALID_UPDATE	UPF_SSET_STATE_NO_UPDATE
UPF_SSET_STATE_NORMAL_UPDATE	UPF_SSET_STATE_UPDATE
UPF_SSET_STATE_UPDATE_LEGAL	UPF_SSET_STATE_VALUES_INVALID
UPF_SSET_UPDATE_FIRST_USE	UPF_STATE_NOT_DEFINED_ERROR
UPF_STRATEGY_SIGNAL_SPECIFIED	UPF_SUPPLY_PORT_ALREADY_CONNECTED
UPF_SUPPLY_PORT_IMPLICIT_CONNECTION	UPF_SUPPLY_PORT_NO_LOWCONN
UPF_SUPPLY_PORT_UNCONNECTED	UPF_SUPPLY_STATE_ALIAS
UPF_SUPPLY_STATE_TYPE_MISMATCH	UPF_SUPPLY_STATE_VALUE_MISMATCH
UPF_SUPPLY_STATE_VALUES_INVALID	UPF_SV_KEYWORD_AS_ID
UPF_TIME_INVALID_UNIT	UPF_UPF_KEYWORD_AS_ID
UPF_VALUE_NOT_SPECIFIED	UPF_VCT_NOT_FOUND
TCL_OPTION_NOVALUE	TCL_OPTVAL_LESSVALUES
UPF_OBJECT_NOT_FOUND	TCL_OPVAL_LISTLENGTH

Parser Violations	Parser Violations
TCL_OPTION_UNKNOWN	TCL_COMMAND_UNKNOWN
TCL_OPTION_MANDATORY	UPF_GROUP_STATE_NO_UPDATE
UPF_ATTR_TOO_FEW_PORTS	UPF_SUPPLY_PORT_NO_HIGHCONN

Table 3-7 lists the VC LP checks that are supported in the DIUC flow.

Table 3-7 Supported VC LP Tags in DIUC Flow

Tag Name	Tag Name	Tag Name
COMPAT_FUNCTION_DEEP	COMPAT_FUNCTION_USER	COMPAT_LOCATION_AUTOMATIC
COMPAT_LS_SUPPLYSET	COMPAT_PST_TRANSIENT	COMPAT_SUPPLYNET_SCOPE
COMPAT_SUPPLYSET_SCOPE	DIFF_PST_EXIST	DIFF_PST_STATE
DIFF_PST_SUPPLY	DIFF_SET_GROUND	DIFF_SET_EXIST
DIFF_SET_NWELL	DIFF_SET_POWER	DIFF_SET_PWELL
DIFF_SUPPLY_STATE	DIFF_SUPPLYSTATE_ATTR	ISO_CONTROL_MISSING
ISO_SENSE_DEFAULT	ISO_SENSE_MISSING	ISO_SUPPLY_MISSING
ISO_SUPPLY_UNAVAIL	LS_SUPPLY_UNAVAIL	PST_GROUND_NONZERO
PST_STATE_INVALID	PST_STATE_MULTIPLE	PST_STATE_ZERO
PST_SUPPLY_MULTIPLE	PST_VOLTAGE_DROPPED	PST_VOLTAGE_UNUSED
PSW_ACK_MISSING	PSW_CONTROL_STATE	PSW_CONTROL_VOLTAGE
PSW_IN_EXTRA	PSW_OUT_EXTRA	PSW_OUTPUT_STATE
PSW_STRATEGY_REDUND	PSW_SUPPLY_SCOPE	PSW_SUPPLY_STATE
PSW_SUPPLY_UNAVAIL	RET_CONTROL_MISSING	RET_STRATEGY_BACKUP
RET_STRATEGY_MISSING	RET_STRATEGY_NORETAIN	RET_STRATEGY_NOSAVRES
RET_SUPPLY_MISMATCH	RET_SUPPLY_MISSING	RET_SUPPLY_UNAVAIL
SUPPLY_UNAVAIL_UNDETERMINABLE	UPF_BIAS_MISSING	UPF_COMMAND_NYI
UPF_CSN_LOGIC	UPF_CSN_UNAVAIL	UPF_DOMAIN_EQUIVALENT
UPF_ISO_CONTROL	UPF_NET_MULTIPLE	UPF_OPTION_EMPTYLIST
UPF_NOSHIFT_PORT	UPF_OBJECT_UNDEF	UPF_PRIMARY_MISSING
UPF_OPTION_NYI	UPF_PORT_IMPLICIT	UPF_PST_USER
UPF_PRIMARY_UNAVAIL	UPF_PST_MISSING	UPF_STATE_DUPLICATE
UPF_REPEATERS_UNAVAIL	UPF_SOURCESINK_PORT	UPF_SUPPLY_MULTISTATES

Tag Name	Tag Name	Tag Name
UPF_SUPPLY_BOTH	UPF_SUPPLY_MISSING	UPF_SUPPLY_OFF
UPF_SUPPLY_NEGATIVE	UPF_SUPPLY_NOSTATE	UPF_SUPPLY_UNDRIVEN
UPF_SUPPLY_RESOLUTION	UPF_SUPPLYSTATE_MISSING	UPF_TABLE_DROPPED
UPF_SUPPLY_UNUSED		

### 3.5.10.2 Limitations

- ❖ In the DIUC flow, the old save restore support is not available. The checkpoint restart support is available.

## 3.5.11 Support for Smart Database Linking

VC LP supports the smart database linking feature. By default, when more than one cell definition with same name is used in the design, VC Static links the first match found, and continues with the design read. This results in unwanted violations, that is, LS\_INST\_RANGEI, LS\_INST\_RANGEO, PG\_VOLTMAP\_INCONSISTENT.

When more than one cell definition with the same name, but different voltage map values is used, the smart db linking feature can be enabled to link correct cell definition to the correct instance.

To enable this feature, use the following application variable and query commands:

- ❖ The lp\_smart\_relink application variable {Values: None (default), voltage\_range, all}
- ❖ The set\_voltage Command
- ❖ The report\_relink\_lp Command

### 3.5.11.1 The lp\_smart\_relink Application Variable

When the lp\_smart\_relink application variable is set, VC LP finds the matching cell for an instance based on the voltage values of connected supply nets, and the voltage values mapped to the PG pins of cells in the library. If a match is found, it will relink to correct instance. If no match is found, or there is an ambiguity, linking will not happen.

The lp\_smart\_relink application variable can take the following values:

- ❖ none (default): When set to none, no relinking is performed, and the default behavior continues.
- ❖ voltage\_range: When set to voltage\_range, VC LP link the cells with voltage\_range, input\_voltage\_range and output\_voltage\_range attribute, usually the LS cells.
- ❖ all: When set to all, VC LP links all cells considering their voltage\_map values.

### 3.5.11.2 The set\_voltage Command

For relinking, the voltage values of connected supplies are collected from the PST states. Using the set\_voltage command, you can set values for the supplies when choosing the library cell. The lp\_smart\_relink application variable honors the new value, and links the cells based on it.

#### Syntax

```
%vc_static_shell> set_voltage -help
Usage: set_voltage      # Set smart relink voltage for supply nets
       -object_list <nets>      (List of supply net names)
```

```
<float>           (Link voltage)
```

Using the `set_voltage` command does not change the value of supply voltages in PST tables, and does not impact other violation checks. It helps in relinking only.

### 3.5.11.3 The report\_relink\_lp Command

Use the `report_relink_lp` command to get a summary of relink instances. This command can be used to debug relink data. You can view for a cell how many instances match, do not match, and are ambiguous. The output of the `report_relink_lp` command depends on the value of `lp_smart_relink` application variable.

```
%vc_static_shell> report_relink_lp -help
Usage: report_relink_lp      # Report details of smart (voltage aware) instance relinking
       [-instance <instance>] (Show extended details for instance)
```

The following is an example output of the `report_relink` command.

```
%vc_static_shell> report_relink
Matched No match Ambiguous Cell name
1          1      o      DFF
1          1      0      LSLH
```

When the `-instance` option is specified in the `report_relink_lp` command, it helps to debug from the perspective of an instance. For a library to match, all pins should match. In the following example output, you can notice that for the instance `u1` library `smartlink1` can be linked.

```
vc_static_shell> report_relink_lp -instance u1
Library   Pin     VLib     Supply    Vsupply    PMatch    LMatch
smartlink 1      VDD      1.2       v124       1.2 1.3Y    Y
smartlink 1      VDDI     1.2       v124       1.2 1.3Y    Y
smartlink 2      VDD      1.4       v124       1.2 1.3N    N
smartlink 2      VDDI     1.4       v124       1.2 1.3N    N
```

Where:

{`PMatch`: Pin match, `LMatch` : Library match, `Vsupply` : PST values/ `set_voltage` , `VLib` : `volt_map` values}.

### 3.5.11.4 Limitations

Difference between cell definition should be only in voltage values {`volt_map`, input/output voltage range}, The other differences are not honored. The cell name should be the same.

## 3.6 Analyzing Compiler Messages

You can capture the screen log of a VC Static run and ensure all the warnings and errors are captured.

```
%vc_static_shell -o screen.log -file vcst.tcl
```

## 3.7 Running VC LP Checks

For performing checks, based on the stage of the design it is important to understand the way in which VC LP checks are organized. There are three separate stages of the design:

1. Design / UPF creation: The UPF is being created and the design does not yet contain inserted low power objects such as level shifters or isolation gates.
2. Post-synthesis: The design contains inserted low power objects, but connection of all the power and ground nets has not yet been performed.

### 3. Post-route: The design contains all low power objects and power/ground connections.

In the first stage, it clearly does not make sense to perform checks related to insertion of level shifters and isolation gates, or power and ground connection checks. The goal of checking at this stage is to predict any problems which will be encountered during later processing. The following are several examples of problems that are detected during this stage:

- ❖ Relevant objects such as hierarchical instances referred to in the UPF do not exist in the design. UPF requires definition of supply nets, which will not exist in an early stage design; so only relevant objects are checked.
- ❖ Inconsistency between parts of the UPF, such as a crossing between two domains where the driver is off, but the receiver is on and there is no isolation strategy defined.

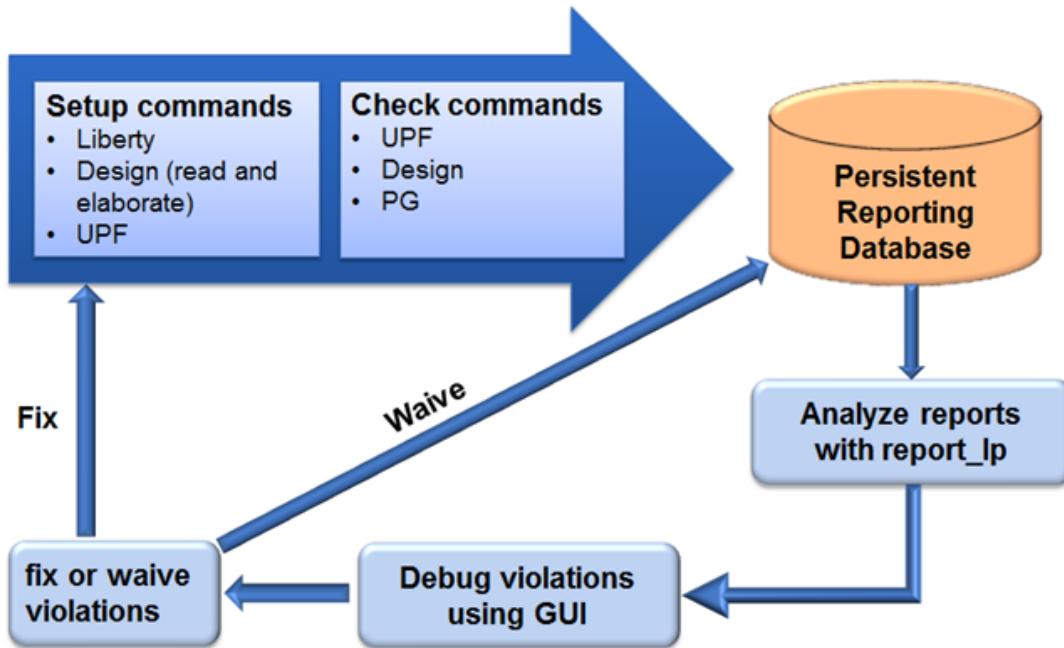
In the second or post-synthesis stage, all of the previous checks should be performed. Although the UPF may be stable at this point, design changes may cause new UPF inconsistencies to appear. The correct solution for these problems may be to change the UPF. In addition, the electrical correctness of the design can be verified. The following are several examples of problems which are detected during this stage:

- ❖ A level shifter is needed at a crossing, but there is no level shifter in the design.
- ❖ An isolation gate is present at a crossing where it is needed; but the isolation control signal routed to the gate does not match what is specified in the UPF.

In the third or post-route stage, all of the previous checks should be performed. Inconsistencies may have appeared that are best fixed by changing the UPF or the synthesis results. In addition, at this stage all of the power and ground connections can be checked for electrical correctness and to make sure that they are consistent with the UPF. The following are several examples of problems which are detected during this stage of the design:

- ❖ A level shifter is present at a crossing, but the supply connected to its input power pin does not match the supply connected to the gate driving the level shifter.
- ❖ A macro has a power pin connected to a power supply net, but the specific supply connected does not match the supply specified in the UPF.

**Figure 3-3 VC LP Design Flow**



In terms of VC LP commands, the `check_lp -stage upf`, `check_lp -stage {upf design}` `check_lp -stage {upf design pg}` commands correspond to these three stages.

It is important to run both `check_lp -stage upf` and `check_lp -stage design` commands at the post-synthesis stage. Skipping the `check_lp -stage upf` command will not miss any errors, but it may be hard to discover that a certain problem can be best fixed by changing the UPF. For the same reason, it is important to run the `check_lp -stage upf`, `check_lp -stage design` and `check_lp -stage pg` commands at the post-route stage.

### 3.7.1 VC LP Database

#### 3.7.1.1 Generating Crossovers

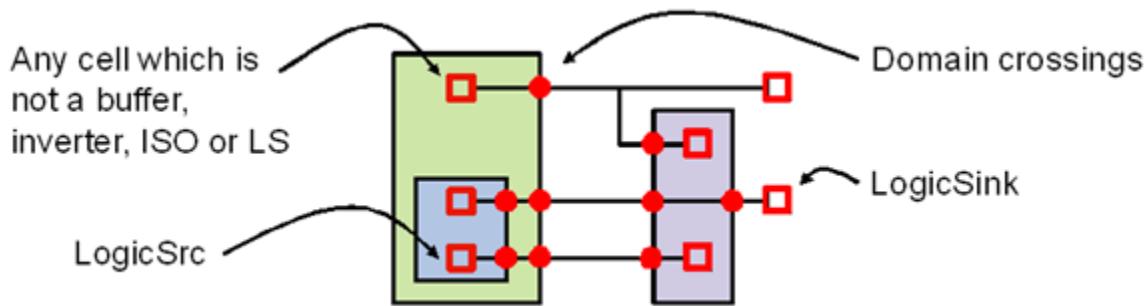
When you run the `check_lp` command, VC LP generates the low power crossover database. However, you can also generate the crossover database directly using the `generate_crossovers` command.

The crossover (xover) in VC LP is a design path that is crossing voltage and/or power domain boundaries. Each crossover path may include several design nodes in between that may be hierarchical ports/net/pins, cells (such as buffers, inverters) and low power special cells (such as ISO, LS and ELS cells).

VC LP creates crossover in the database following the below rules:

- ❖ Identifying all ports with domain crossings
- ❖ Identifying the logic sources, that is, VC LP traces fan-in to the first cell which is not a buffer, inverter, IOS or LS.
- ❖ Identifying the logic sink, that is, VC LP traces fan-out from each logic source to the first cell which is not a buffer, inverter, ISO, ELS, LS.

Each Crossover is a path that goes from a LogicSrc to a LogicSink. For the example in [Figure 3-4](#), VC LP generates four crossover path in the database.

**Figure 3-4 Creating Crossovers Example**

**Note**  
Exception Crossovers (CSN/SRSN/PG/infer\_domain) are additionally generated.

### Support for enable\_lp\_dump\_debug\_reports dump\_crossover\_through\_info Command

When the `enable_lp_dump_debug_reports dump_crossover_through_info` command is set, the `xoverThroughInfo.csv` report file is saved inside `vcst_rtdb/lpdb/debug_reports` folder during the crossover generation.

The csv file contains *Xover #, Source, Sink, PD\_Source, PD\_Sink, PD\_Through\_#, Boundaries* for each crossover path.

The *PD\_Through\_#* field reports all the domains which the crossover path goes through except for the source domain and the sink domain. The command decides the max shown *PD\_Through\_#* numbers.

#### Example

The following is an example of the *xoverThroughInfo.csv* report file:

```
Xover #, Source, Sink, PD_Source, PD_Sink, PD_Through_0, PD_Through_1, PD_Through_2,
PD_Through_3, Boundaries
Xover #1 i_bloc_IP_SS1/AND_IP_SS1_4/Z, i_bloc_IP_SS2/i_bloc_IP_SS3/AND_IP3_2/B, PD_SS1,
PD_SS3, TOP, PD_SS2, EMPTY, EMPTY, i_bloc_IP_SS1/out3, i_bloc_IP_SS2/in5,
i_bloc_IP_SS2/i_bloc_IP_SS3/in4
```

#### 3.7.1.2 Generating Inter Tile/Module Based Crossovers

By default, VC LP perform the crossover generation from a set of starting nodes. These starting nodes are boundary ports of power domains specified in the design. VC LP performs a transitive fanin traversal from each of the boundary ports until it finds a node which can be a root. This is called the crossover root, from which VC LP performs a fanout traversal to build the crossover tree. VC LP names these crossover roots as Logic Source.

Under an application variable, VC LP enables you to generate crossover paths from specific starting nodes. You can specify the starting nodes from which crossovers must be generated, and VC LP generates the crossover only from the specified starting nodes and does not generate the crossovers within the specified instance. VC LP also performs all the crossover based checks only on the crossovers which are generated from the specified starting nodes. The overall performance of the `check_lp` run is improved as the crossovers entirely contained within a module specified with the application variable are ignored for analysis.

#### Use Model

Use the following application variable to specify the instances within which VC LP does not need to build crossovers.

```
%vc_static_shell>set_app_var ignore_intra_instance_xover {instance1 instance2}
```

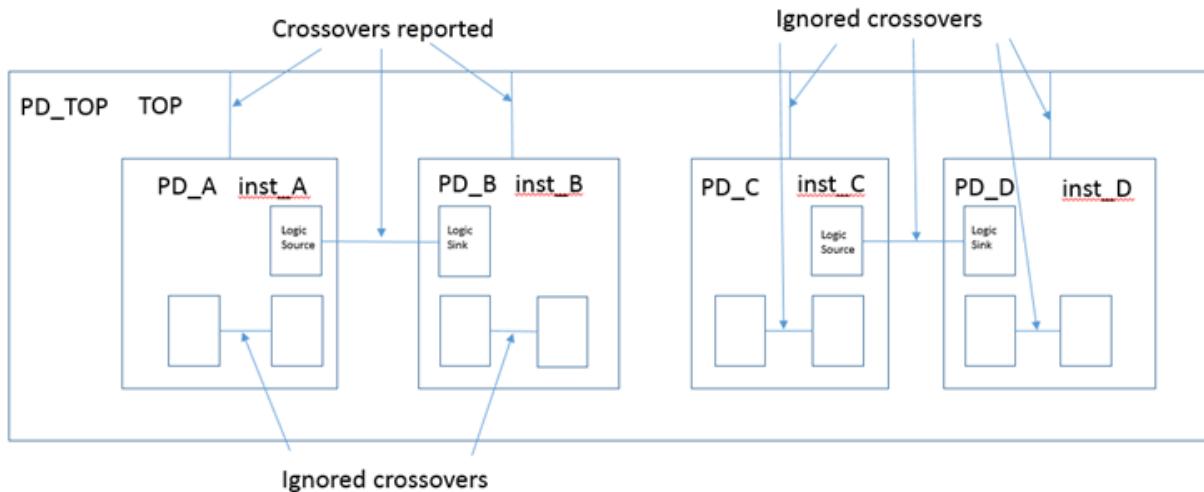
where tile1 and tile2 are the instances from which the crossover generation must start.

- ❖ The instance list must be separated by a space.
- ❖ The instance name must be a full hierarchical name \*without\* top.
- ❖ The instance name must not contain wild card characters.

### Example

For the example design as shown in the following figure, if the following application variable is set, VC LP reports the crossovers from inst\_A to inst\_B, and not the crossovers within inst\_A and inst\_B.

```
%vc_static_shell>set_app_var ignore_intra_instance_xover {inst_A inst_B}
```



```
%vc_static_shell> report_crossover [get_crossovers -from_signal inst_A/dff1/Q -to_signal inst_B/dff1/A]
```

```
Crossover          : "inst_A/dff1/Q inst_A/out1 inst_B/in1 inst_B/dff1/A"
Source Power Domain : PD_A
Dest Power Domain   : PD_B
Domain Boundary     : inst_A/out1, inst_B/in1
Source Signal       : inst_A/dff1/Q
Through Signal      : inst_A/out1
Through Signal      : inst_B/in1
Dest Signal         : inst_B/dff1/A
Strategy Info       :
Device Info         :
```

---

### 3.7.1.3 Resolving Supplies in Crossover

Supply connection for a given crossover node can come from the following sources

- ❖ Physical connectivity in the netlist (PG)
- ❖ connect\_supply\_net in UPF (CSN)

- ❖ set\_related\_supply\_net in UPF (SRSN)
- ❖ Domain primary supplies in UPF (DomainSupply)
- ❖ Strategy supplies in UPF (StrategySupply)
- ❖ set\_port\_attributes (driver\_supply, receiver\_supply and repeater\_supply) in UPF and so on

VC LP has a predictable way for supply resolution for any given crossover node that have more than one source of supply information. The thumb rule that is followed for lib cells that have more than one supply connection is

*SRSN> PG > CSN > StrategySupply > DomainSupply*

What it means is, for a lib cell if the CSN is missing, VC LP implicitly falls back to the policy supply (for special cells, such as ISO, LS, ELS) and to the domain supply for standard cells. In the absence of strategy associated for special cells, VC LP falls back to the domain supply.

The resolved supply is what will be used for strategy resolution (for example, resolving set\_isolation - source -sink applicability) for PST based checks.

VC LP supports mixed supply modes. For example, an IP block can be PG netlist and top level can be non-PG netlist with supply connections in UPF. So, tool still handles part PG + Part UPF connections well. Whatever supply was chosen and the supply method used is clearly mentioned in the violation reports. For example, SourceInfo, SinkInfo, InstanceInfo debug fields have the full details for the same.

```
-----
ISO_INST_SOURCE (1 error/0 waived)
-----
Tag : ISO_INST_SOURCE
Description : Nor-style isolation instance [Instance] ([Cell]) present,
             but strategy supply on when source supply off
Violation : LP:1
Instance : iso1
Cell : ISOLNORX1_RVT
Strategy : PD_TOP/pd_top_iso_in
StrategyNode : IN
Source
  PinName : IN
SegmentSourceDomain : PD_TOP
SourceInfo
  PowerNet
    NetName : VVDD
    NetType : Design/UPF
  PowerMethod : FROM_UPF_DRIVER_SUPPLY
  GroundNet
    NetName : VSS
    NetType : Design/UPF
  GroundMethod : FROM_UPF_DRIVER_SUPPLY
InstanceInfo
  PowerNet
    NetName : VDD
    NetType : Design/UPF
  PowerMethod : FROM_PG_NETLIST
  GroundNet
    NetName : VSS
    NetType : Design/UPF
  GroundMethod : FROM_PG_NETLIST
```

```
States
State      : top_pst/vvdd_off
```

**NOTE**

The following notes also apply to supply net resolution:

- ❖ DUT top level ports have precedence for exception supplies (SRSN or SPA). If the absence of exception supply connections in UPF, VC LP by default relates them to the domain supply of the top domain
- ❖ Typically hierarchical ports do not have any supplies associated. For example, SRSN on a hierarchical port is ignored for any electrical checks perspective, but can be used for UPF consistency checks.
- ❖ The power and ground precedence orders are computed separately. For example, suppose a standard cell has `connect_supply_net` specified only for power. Then, for ground, resolved net computation will fall back to the primary supply of the parent domain.
- ❖ If a physical net is connected to a supply pin, this does not guarantee it as resolved supply. Only physical nets which have a corresponding supply net declared in UPF and used in the power state table are considered as supply nets.
- ❖ For UPF `connect_supply_net` and strategy supplies, the supply net must be present in the power state table to become an effective resolved supply.
- ❖ The default resolved supply will be the parent domain's primary supply. However, if these supplies are not present in the UPF power state table, an error is produced and the design will not be processed.

**3.7.1.4 Support for `lp_analyze_without_voltages` Application Variables**

Use the `lp_analyze_without_voltages` application variable for performing crossover checks in designs at the RTL stage without the UPF voltage information.

When the `lp_analyze_without_voltages` application variable is set to true, limited crossover checks are performed. Crossover checks that involve purely ON and OFF scenarios are supported, but any checks that require voltage values from UPF as reference are not performed.

When the `lp_analyze_without_voltages` application variable is set to true, voltage settings are not allowed in design. Only the supply states with name ON or OFF are supported. VC LP would assume function power and ground are both FULL\_ON for supply state ON and function power or ground are OFF for supply state OFF.

```
The following UPF is supported under set_app_var lp_analyze_without_voltages true
create_power_domain TOP -elements { . }
create_power_domain PD1 -elements { CORE1 }
create_power_domain PD2 -elements { CORE2 }

#power states definitions without voltages
add_power_state -supply TOP.primary -state { ON }
add_power_state -supply PD1.primary -state { ON -simstate NORMAL } -state OFF
add_power_state -supply PD2.primary -state ON { -simstate NORMAL }

add_power_state -domain TOP -state { ON -logic_expr {TOP.primary == ON} }
add_power_state -domain PD1 -state { ON -logic_expr {PD1.primary == ON} } -state { OFF
-logic_expr {PD1.primary == OFF} }
add_power_state -domain PD2 -state { ON -logic_expr {PD2.primary == ON} }
```

```

create_power_state_group TOP_G
add_power_state -group TOP_G -state ALL_ON {-logic_expr {TOP == ON && PD1 == ON && PD2 == ON}}
add_power_state -group TOP_G -state PD1_OFF {-logic_expr {TOP == ON && PD1 == OFF && PD2 == ON}} -update

```

The following parser error is reported when voltage values are defined for the supply states and `lp_analyze_without_voltages` is set to true

*top.upf:6: [Error] UPF\_VOLTAGE\_DISALLOWED: Voltages are not allowed  
Voltages are not allowed when variable lp\_analyze\_without\_voltages is true.*

### Limitation

Predefined power states ("ON" and "OFF") are not supported for this feature.

#### 3.7.1.4.1 Handling SRSN as Liberty Override

VC LP treats SRSN specified at lib cell pins as the override to all other supplies at a given node by default. Hence the supply net resolution precedence rules apply by default.



If you do not want to treat SRSN specified at lib cell pins as the override to all other supplies at a given node, then set the variable `enable_dc_srsn = false`.

VC LP uses SRSN at the lib cell pin as the ultimate resolved supply for all its analysis of the overriding PG connectivity, RPP+CSN (`related_supply_pin+connect_supply_net`), strategy supplies or domain supplies if one or more of these are available at a given node. The eventual resolved SRSN supply is used for protection analysis, device to strategy association and all other checks.

#### 3.7.1.5 Special Cells to UPF Strategy Association

##### 3.7.1.5.1 Power switches

Power switch strategy to cell association in VC LP happens based on any one of 3 directives, two of which are user directives and the third and default one is a tool heuristics based.

- ❖ Instance naming convention used in the PG netlist when it is written out by ICC
- ❖ Enable this feature using the `use_icc_switch_names` application variable, and hence is a user directive
- ❖ Strategy to cell matching based on patterns specified in the `bind_switch_strategy`, hence this is also an user directive.
- ❖ Heuristic based: VC LP gives weightage to a match among various attributes of the strategy and cell (output supply, input supply, control pin and acknowledge pin in that order) and based on weightage VC LP associates the cell to the right strategy. This is the default mode and heuristics based.

For complex designs having multiple power domains and power switch strategies and several thousands of power switch cells, one or more of the above directives may have been used for strategy for matching cells.

If you find that a power switch strategy is incorrectly associated with a cell, and thereby leading to false violations in VC LP, the following debug aid can help understand what directive was used by tool for such association.

VC LP can dump a new debug file containing details of the power switch association results. You must set the following command in the script

```
%vc_static_shell> enable_lp_dump_debug_reports dump_crossover
```

And look into file vcst\_rtdb/.internal/lp/powerSwitchAssociation.rpt.

Here is an example file snippet:

```
-----> Pin: a/VDD policy: pa match: input_supply VDDIA
-----> Pin: a/VDDSW policy: pa match: output_supply VDDOA
---> Instance: a policy: pa reason: by_weight
=====
---> Instance: snps_pa_snps policy: pa reason: iccname
=====
-----> Pin: snps_pb_snp/VDD policy: pb match: input_supply VDDIB
-----> Pin: snps_pb_snp/VDDSW policy: pa match: output_supply VDDOA
---> Instance: snps_pb_snp policy: pa reason: by_weight
```

The === separates one power switch instance from the next. Each associated instance has one summary line with "--->" and one of three reasons/directives:

1. iccname: It means that user directive "set\_app\_var use\_icc\_switch\_names true" (not the default) has been used and the power switch instance is associated to strategy based on this directive
2. bind: It means that user directive "bind\_switch\_policy" has been used and the pattern matched a strategy instance name.
3. by\_weight: It means a match was made by matching at least one power switch instance pin against a strategy attributes, and the strategy with the highest weight was selected.

In this case there will be at least one line like "-----> Pin" immediately before, where either the input\_supply, output\_supply or control pin matches.

If you study the third instance in the above example carefully, you will notice it matches one policy for input\_supply, but a different policy for output\_supply. Since output\_supply has higher weight, the instance is associated with the output\_supply strategy. In this case, there will be a PSW\_INPUT\_CONN error.

## NOTES

- ❖ The enable\_lp\_dump\_debug\_reports dump\_crossover command dumps a lot of debug information apart from above file. It can slowdown the run and the vcst\_rtdb size may increase depending upon the design complexity and size.
- ❖ The above file is only a debugging aid and should not be used for performance analysis and so on. you can contact support\_center@synopsys.com if you still need further help with your design analysis.

### 3.7.1.5.2 Power State Table

For static verification tool such as VC Static, power state table is a golden reference. The power state table captures the relationship of various supplies in the design which represent various power modes. Typically users specify the port states on supplies either through add\_port\_state or add\_power\_state commands. Later, create\_pst is used to define the power state table.

#### PST merging

For flat UPF, typically only one PST is defined at the top level UPF. Hierarchical UPF methodology generally has PSTs being defined at the child level as well. In such cases, too can auto derive a merged PST (called System PST) by doing an intersection of all PSTs. The merging can happen for PSTs in the same scope as well.

#### Debugging System (merged) PST

Top designs may have a large number of power state tables with complex interactions. All LP-aware tools merge the power state tables into a "system" power state table. Merging may cause unexpected interactions between supplies, and make certain power states invalid. VC LP provides unique messages and tools to understand these relationships.

This section describes different commands available to debug power state tables in VC LP.

Following is the methodology suggested for pst debugging:

- ❖ Read the design (`analyze`, `elaborate`, `read_file` and so on)
- ❖ Read the UPF (`read_upf`)
- ❖ Check the UPF specific to pst (`check_lp -stage upf ;Vpst ; report_violations -app LP`)

In this step, watch out for violation tags such as

- ◆ **PST\_VOLTAGE\_DROPPED:** During the merging of power state tables, one or more power states were defined for the parent instances that were not defined in a child instance. Using a power state at the parent level that does not exist at the child level causes the child instance to operate under conditions for which it was not designed, which may cause the design to function improperly or not function at all. Since the child instance has no definition for the power state, as state was explicitly dropped from the merged power state table, to avoid operating the child instance in an illegal state.
- ◆ **PST\_VOLTAGE\_UNUSED:** During the merging of power state tables, one or more power states were defined for a child instances that were not defined for the parent instance. Since the parent instance has no definition for that power state, that state will not be included in the merged power state table.

This situation is common. For example, you may be instantiating a sub-block that has been designed to operate in a variety of power states, but for this particular application, only some of those states are actually needed. Having one or more child power states not being used at the parent level will not make the design fail.

- ❖ Generate the system pst (`report_system_pst`)

Use this command to print the system power states which are present in a design after merging the individual power state tables.

For designs with a large number of independent supplies, the full table may be huge. Use the `-summary` option for an unfamiliar large design first, to prevent the potential huge runtime of printing millions of states. For more options, please type `man report_system_pst` in VC LP.

Example UPF:

```

add_port_state VDD1 -state "H1 1.2"
add_port_state VDD2 -state "H2 1.2" -state "Z2 off"
create_pst top_pst -supplies "VDD1 VDD2 VSS"
add_pst_state t0 -pst top_pst -state "H1 H2 ZV"
add_pst_state t1 -pst top_pst -state "H1 Z2 ZV"

set_scope ua
add_port_state VDD3 -state "H3 1.2" -state "L3 1.1"
add_port_state VDD4 -state "H4 1.2" -state "Z4 off"
create_pst ua_pst -supplies "VDD3 VDD4 VSS"
add_pst_state t0 -pst ua_pst -state "H3 H4 ZV"
add_pst_state t1 -pst ua_pst -state "L3 Z4 ZV"

set_scope /

```

```
connect_supply_net VDD1 -ports ua/VDD3
```

### Example command lines and results:

```
%vc_static_shell> report_system_pst -summary
Supplies: 3 Port states: 6 Potential system states: 8 Actual system states: 2
%vc_static_shell> report_system_pst
VDD1,VDD2,ua/VDD4
1.2,*,1.2

%vc_static_shell> report_system_pst -loop
VDD1,VDD2,ua/VDD4
1.2,OFF,1.2
1.2,1.2,1.2

%vc_static_shell> set s [get_supply_nets VDD*]
{"VDD1", "VDD2" }

%vc_static_shell> report_system_pst -supplies $s
VDD1,VDD2
1.2,*
```

- ❖ Analyze the invalid power states (analyze\_invalid\_power\_state)

In a design with many PST, the system power state table may be huge (30 supplies => 1 billion states). Many power states may be invalid on purpose. Therefore, VC LP does not display the invalid states by default (there is no warning or report\_violations -app LP violation for this).

To see all the invalid states:

```
report_pst_state -only_invalid
PST State : s1_01
Full Name : pst1/s1_01
vdda| vddb|
pst1/s1_01: OFF| HV|
```

Let us see couple of examples for invalidation.

Example 1: In the following case, as s1\_01 state is not available in any other pst with the same supplies, it gets invalidated during pst merging.

```
create_pst pst1 -supplies {VA VB}
add_pst_state s1_00 -pst pst1 -state {OFF OFF}
add_pst_state s1_01 -pst pst1 -state {OFF HV} „3 Invalid state
add_pst_state s1_10 -pst pst1 -state {HV OFF}
add_pst_state s1_11 -pst pst1 -state {HV HV}
create_pst pst2 -supplies {VA VB}, „3Killer
add_pst_state s2_00 -pst pst2 -state {OFF OFF}
add_pst_state s2_10 -pst pst2 -state {HV OFF}
add_pst_state s2_11 -pst pst2 -state {HV HV}
```

Example 2: Let us take little complex case with several pst tables.

```
create_pst pst1 -supplies {VA VB}
add_pst_state s1_00 -pst pst1 -state {OFF OFF}
add_pst_state s1_01 -pst pst1 -state {OFF HV}
add_pst_state s1_10 -pst pst1 -state {HV OFF}
add_pst_state s1_11 -pst pst1 -state {HV HV}

create_pst pst2 -supplies {VA VB VC}
add_pst_state s2_000 -pst pst2 -state {OFF OFF OFF}
add_pst_state s2_011 -pst pst2 -state {OFF HV HV}
add_pst_state s2_101 -pst pst2 -state {HV OFF HV}
```

```

add_pst_state s2_111 -pst pst2 -state {HV HV HV}

create_pst pst3 -supplies {VA VB VD}
add_pst_state s3_000 -pst pst3 -state {OFF OFF OFF}
add_pst_state s3_010 -pst pst3 -state {OFF HV OFF}
add_pst_state s3_101 -pst pst3 -state {HV OFF HV}
add_pst_state s3_111 -pst pst3 -state {HV HV HV}

create_pst pst4 -supplies {VC VE}
add_pst_state s4_00 -pst pst4 -state {OFF OFF}
add_pst_state -pst s4_11 -state {HV HV}
create_pst pst5 -supplies {VD VE}
add_pst_state -pst s5_00 -supplies {OFF OFF}
add_pst_state -pst s5_11 -supplies {HV HV}

```

❖ Finding the reason for invalidation (analyze\_invalid\_power\_state)

In the previous example, “analyze\_invalid\_power\_state” can help understand the reason for invalidation of s1\_01 as shown below:

```

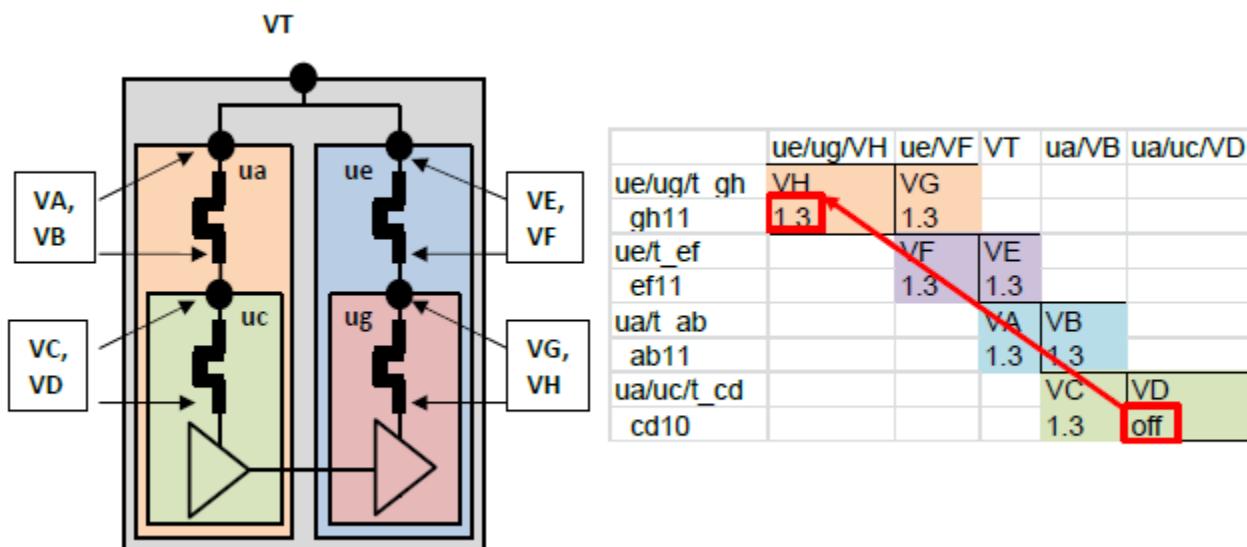
analyze_invalid_power_state -state pst1/s1_01
State is invalid due to combination of following PSTs:
pst2 pst3 pst4 pst5
Supplies: VA| VB| VC| VD| VE|
Invalid: OFF| HV| | | |
Valid: OFF| OFF| OFF| OFF| OFF|
Valid: HV| OFF| HV| HV| HV|
Valid: HV| HV| HV| HV| HV|

```

❖ Finding unexpected path between supplies (analyze\_pst\_chain)

Very frequently, you may see cases that isolation strategy is missing from one supply to another supply. Especially, this happens when two integrated into a design and individual IP owners may not have thought about this interaction. When you see violation tags such ISO\_STRAT\_MISSING, user can analyze the pst chain using “analyze\_pst\_chain”.

In the following example circuit, four blocks each with a local supply. VC LP generates missing isolation strategy from VD to VH. Reason is not obvious to any single block owner.



Use the `analyze_pst_chain` command to find any chain of supplies where source VD is off and sink VH is on for example circuit as shown below:

Long hierarchical names are abbreviated for the table

	S1	S2	S3	S4	S5		Supply names :
T1:	VH	VG					S1. ue/ug/psw/VDDSW
gh11:	1.3	1.3					S2. ue/psw/VDDSW
T2:		VF	VE				S3. VT
ef11:		1.3	1.3				S4. ua/psw/VDDSW
T3:			VA	VB			S5. ua/uc/psw/VDDSW
ab11:			1.3	1.3			
T4:				VC	VD		
cd10:				1.3	OFF		

	Table names :		
T1:	ue/ug/t_gh		
T2:	ue/t_ef		
T3:	ua/t_ab		
T4:	ua/uc/t_cd		

Merging power state tables for TOP design is a complex procedure. During this process, several voltage values can be dropped, several states can be invalidated and unexpected interactions can be exposed. Designers can use VC LP and follow the debug methodology proposed in this article to analyze system power state tables.

#### ❖ Adding All Off State in PST for Accuracy

Consider the case where an IP designer did not write an "all off" state for the IP; but in the SOC, there is a power state where the IP supplies are all off.

VC LP reported the `PST_VOLTAGE_DROPPED` (Error) violation, as the OFF state is not present in child instance.

IP Level PST		TOP Level PST		
VB1	VB2	VB1	VB2	VT
on	on	on	on	on
on	off	on	off	on
off	on	off	on	on
		off	off	on

VC LP adds always 'OFF' state in PST. This support can be enabled using the `enable_pst_add_all_off` application variable. By default, this application variable is set `false`.

When `enable_pst_add_all_off` is set to true, VC LP generates OFF state for each power net present in PST and the `PST_VOLTAGE_DROPPED` violation is not reported.

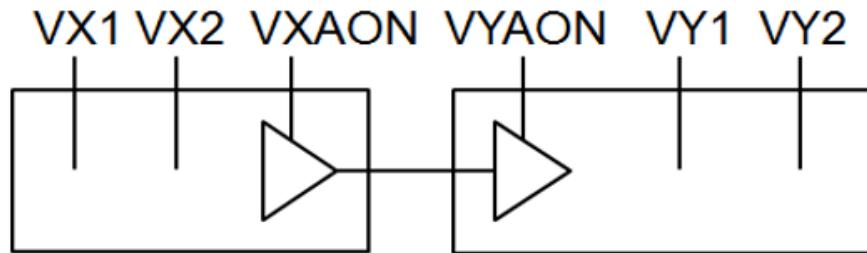
Enhance IP Level PST		Enhance TOP Level PST
VB1	VB2	VT
on	on	on
on	off	on
off	on	on
off	off	on
off	off	off

The Off states in the PST is generated:

- ◆ For the power nets which is present in PST.
- ◆ Unless a design explicitly uses a switched ground, the extra off port state is not generated for any ground supply.

### Example

Some violations are reported due to internal generated OFF state. Consider the example in the following figure.



In this example, VXAON and VY1 are always ON. Assume that, VXAON is present in the PST and VY1 is not present in the PST. Under this application variable, VC LP generates OFF state for VXAON and VC LP reports `ISO_INST_MISSING` violation with following internal generated state `pst_name/_SNPS_GENERATED_STATE_ALL_OFF`.

When any violation which is generated due to internal generated OFF state, VC LP reports `pst_name/_SNPS_GENERATED_STATE_ALL_OFF` state in the violation report.

### The `analyze_add_all_off` Tcl Command

Use the `analyze_add_all_off` Tcl command to get a report of the PST names and supply net names where the `off` state is added.

#### Syntax

```
%vc_static_shell> analyze_add_all_off -help
Usage: analyze_add_all_off      # analyzes where all-off PST would be added
       [-upf]                  (Write output in UPF syntax)
```

The following is an example output when the `analyze_add_all_off` command is executed.

```
Feature enabled: true
Invalid state count: 1
Tables where an off-state will be added:
```

```
CORE1/pst1
CORE2/pst1
pst1
Supplies where an off-state will be added:
VDD
VDD1
VDD2
```

**Limitation:**

- ◆ OFF state is not generated for create\_power\_state\_group.
- ◆ It may create OFF state for the ground nets as well.
- ◆ OFF state is not generated when add\_port\_state and add\_power\_state are defined in the same PST.
- ◆ This support is not available in Verdi.

### 3.7.2 The check\_lp Command

The `check_lp` command makes the use model for performing low-power checks simpler and intuitive. It also takes care of new checks and new grouping of checks.

You can use the preset configurations provided for the checks or configure them based on your methodology. This lets you enable/disable certain checks for different stages based on your methodology's requirement.

For design checking, execute the following commands:

```
%vc_static_shell> check_lp -stage upf
%vc_static_shell> check_lp -stage design
%vc_static_shell> check_lp -stage pg
```

Skip the `check_lp -stage pg` command if the design is not a PG netlist (netlist with power/ground connections). Check\* commands perform low-power checks and store the violations detected in the VC LP database. To see a summary of the violations, execute the following command:

```
vc_static_shell> report_violations -app LP -verbose -file report.txt
```

**Syntax**

```
vc_static_shell> check_lp -help
Usage: check_lp      # This performs all stages of low power verification
       -stage <list>          (List of process stages to be executed:
                                Values: Debug, all, design, library, pg,
                                upf)
       [-force]                (Perform checks even when setup checks fail)
       [-family <list>]        (List of check families to be performed:
                                Values: all, analog, aob, bias,
                                cellcheck, compatibility,
                                designconsistency, designfeedthrough,
                                designfork, diode, functional,
                                hierconsistency, isolation,
                                levelshifter, powerground,
                                powerstatetable, powerswitch, retention,
                                signalcorruption, upfconsistency,
                                upfdiff)
       [-include_info]         (Also display messages with severity INFO)
```

```

[-goal <goal name>]      (Before checking, execute stored configure_lp_tag
commands)
[-j <num threads>]       (Number of threads to be used in parallel mode)

```

## Use Model 1

```

%vc_static_shell>check_lp -stage design -family {isolation}
%vc_static_shell>check_lp -stage design -family {isolation signalcorruption}
%vc_static_shell>check_lp -stage design

```

The following are the `family_options` applicable to the design stage:

- ❖ `-isolation`: Only isolation related checks are performed, such as, checks if an ISO cell is required when it is missing, and vice-versa.
- ❖ `-levelshifter`: Only level\_shifter related checks are performed, such as, checks if a LevelShifter is missing when it is required and vice-versa.
- ❖ `-retention`: Only retention related checks are performed, such as, checks if a retention register is present in the design but a corresponding strategy is not found in the UPF. When the `lp_retention_report_root` application variable is set to true, VC LP reports the root supplies for retention tags.
- ❖ `-signalcorruption`: Only signal corruption related checks are performed.
- ❖ `-analog`: Only analog checks are performed.

## Use Model 2

```

%vc_static_shell>check_lp -stage upf -family upfconsistency
%vc_static_shell>check_lp -stage upf -family isolation
%vc_static_shell>check_lp -stage upf -family {powerswitch powerstatetable}

```

The following are the `family_options` applicable to the UPF stage:

- ❖ `-isolation`: Runs all checks related to isolation allowed for the selected stage(s).
- ❖ `-upfconsistency`: Only UPF consistency related checks are performed. These checks also include checking for presence of `connect_supply_net` to level-shifter power pins and backup pin of dual rail buffers.
- ❖ `-levelshifter`: Runs all checks related to level-shifter allowed for the selected stage(s).
- ❖ `-retention`: Runs all checks related to retention allowed for the selected stage(s).
- ❖ `-powerswitch`: Runs all checks related to power switch allowed for the selected stage(s).
- ❖ `-powerstatetable`: Runs all checks related to PST allowed for the selected stage(s).
- ❖ `-signalcorruption`: Runs all checks related to signal corruption (also known as architecture checks) allowed for the selected stage(s).

The following are the `family_options` applicable to the PG stage:

- ❖ `powerground`: Only checks the power network in the design to ensure that it is consistent with respect to the UPF specification. For example, if the UPF `connect_supply_net` connection does not match with the actual PG connection in the design.

## Use Model 3

```

%vc_static_shell>check_lp -stage pg -family diode
%vc_static_shell>check_lp -stage pg -family bias
%vc_static_shell>check_lp -stage pg

```

- ❖ **-force:** If setup checks find any error violations, by default, other stages are not run. If you choose the **-force** option, other stages chosen will also be run.
- ❖ **-include\_info:** Displays message with severity INFO.

## Use Model 4

By default, `check_lp` runs in serial mode. To enable multi-threading in `check_lp` stage, use the `-j <number_of_threads>` option. %vc\_static\_shell> `check_lp -stage (design/upf/pg) -j <number_of_threads>`



### Note

Multi-Threading support has been added for macro connectivity checks and PG\_VOLTMAP\_INCONSISTENT checks.

If the MT license is not available run continues with serial flow with following warning.

*[Warning] LIC\_CHECKOUT\_FAIL\_WARN: Unable to checkout license for VCLP Multi-Threading Feature. Run will continue in serial mode.*



### Note

To enable multi-threading feature for `check_lp` and `infer_source` stage, the VC-LP-MT-NT license should be checked out.

For example, `check_lp -stage all -j 10` requires one VC-LP-MT-NT and one VC-STATIC-LP licenses.

## Syntax Errors

If there are any syntax errors reported by the tool on the UPF files, use the `remove_upf` command to remove the checked UPF, correct the reported issues and source the corrected UPF using the `read_upf` command. Use the `check_lp -stage upf` command to recheck the UPF database. For more details on how to use the `remove_upf` command, see section “[The remove\\_upf Command](#)”.

The `-stage` option is mandatory for the `check_lp` command. If you do not provide this option, the following error is reported:

```
%vc_static_shell> check_lp
```

*[ERROR]: -stage option is required. To check an RTL design, use -stage upf.*

*To check a netlist design, use -stage {upf design}. To check a PG routed design, use -stage {upf design pg}.*

If you give a non-overlapping choice of stage and family, the following error is reported:

```
%vc_static_shell > check_lp -stage upf -family diode
```

*[ERROR]: -stage and -family options do not overlap. There are no checks with both the given stages and given families. Please check your options.*

If you provide duplicate options, the following warning message is reported:

```
%vc_static_shell> check_lp -stage upf -stage design
```

*Warning: duplicate option '-stage' overrides previous value. (CMD-018)*



### Note

The following warning message is reported when the runtime of the design is high because of the high count of multidriver nets. This message is reported only when the multidriver nets count is greater than 100k.

*([Warning] DESIGN\_RUNTIME\_HIGH: Possible high runtime design. Reason: 'High count of multidriver nets')*

### 3.7.2.1 Analyzing the Convergence Metric of a Design

With intent to give more statistical data on the LP design, VC LP introduces the `report_lp_statistics` command. This command in the future is proposed to replace \*\_OK messages reported by VC LP. It gives statistical data on LP objects in the design and their pass percentage before or post LP check analysis is performed. It accurately lists the total, correct count, and percent correct for each LP object. This data serves as a converge metric to ensure that tool has covered for all LP objects of the design and they are now categorized into objects that have violations (check failures) or objects that do not have violations (traditionally reported as \*\_OK messages).

#### Syntax

```
%vc_static_shell> report_lp_statistics -help
Usage: report_lp_statistics      # reports counts and percentages for LP objects
       [-verbose]                (Display the names of all correct objects (may be large))
       [-stage <list>]            (Include statistics for these stages:
                                    Values: all, design, pg, upf)
```

#### Use Model

The `report_lp_statistics` command takes no options except `-verbose`. It can be run at any time after reading the UPF, on a design at any stage (RTL, netlist, PG). However, some lines of the output are only useful after running checks, and some lines are only useful on netlist or PG designs.

When this command is run before any checks are executed, the following output is shown:

```
%vc_static_shell> report_lp_statistics
      Type : Total  Correct Percent
Isolation strategies : 1      N/A    N/A
Level shifter strategies : 0      N/A    N/A
Power switch strategies : 0      N/A    N/A
Retention strategies : 0      N/A    N/A
Crossover boundaries : 5      N/A    N/A
Isolation instances : 1      N/A    N/A
Level shifter instances : 0      N/A    N/A
Power switch instances : 0      N/A    N/A
Retention instances : 0      N/A    N/A
Multirail macro instances : 0      N/A    N/A
Total instances : 1      N/A    N/A
Supply pins : 4      N/A    N/A
Bias pins : 0      N/A    N/A
N/A: corresponding checks not yet run
```

The *Correct* and *Percent* columns are filled in when the checks are run. After you run the `check_lp -stage upf` command, these columns are filled in for the strategy and boundary node lines. After `check_lp -stage design` command is run, these columns are filled in for the instance lines. After `check_lp -stage pg` command is run, these columns are filled in for the pin lines. The following is an example of the same design after all the checks have been run:

```
%vc_static_shell> report_lp_statistics
      Type : Total  Correct Percent
Isolation strategies : 1      1      100%
Level shifter strategies : 0      0      100%
Power switch strategies : 0      0      100%
Retention strategies : 0      0      100%
Crossover boundaries : 5      5      100%
Isolation instances : 1      0      0%
Level shifter instances : 0      0      100%
```

Power switch instances	:	0	0	100%
Retention instances	:	0	0	100%
Multirail macro instances	:	0	0	100%
Total instances	:	1	0	0%
Supply pins	:	4	0	0%
Always-on instances	:	18	2	11%
 Bias pins	:	0	0	100%

When the `report_lp_statistics` command is run with the `-verbose` option, it prints the names of all of the OK objects. For the previous example, you can see the following output:

```
%vc_static_shell> report_lp_statistics -verbose
Correct isolation strategies
    CORE1/PD1/iso1

    Correct level shifter strategies

    Correct power switch strategies

    Correct retention strategies

    Correct crossover boundaries
    CORE1/in2
    CORE1/iso_i1/B
    CORE1/iso_block
    CORE1/in1
    CORE1/out1

    ...
    ...
    ...

    ...
```

### 3.7.3 Merging Reports of Multiple Parallel check\_lp Runs

You can run multiple VC LP checks and merge the reports from different runs to the current report. Use the `merge_database` command to read a report database on the disk from a previous run, and merge it into the report database for the current run.

You can use the `merge_database` command:

- ❖ For parallel runs, where multiple simultaneous executions of VC Static perform a different set of checks.
- ❖ To combine the violations from different check commands into a single database, which helps in reporting and querying violations.
- ❖ To integrate check engines which are not part of VC Static, but which can write database files using the proper schema. This allows results of multiple check engines to be displayed together using `view_activity` in the Verdi GUI.

#### Syntax

```
%vc_static_shell> merge_database -help
# Merges report database from a saved run into the current run
```

```
[ -session <session name> ]
          (Name of saved session to merge messages from)
```

The `session name` refers to the session saved from a previously completed run with `save_session`.

### Example

Consider the following two run scripts.

`run1.tcl`

```
##load design and upf##
check_lp -stage pg
save_session -session run1
report_violations -app LP # note "a"
```

`run2.tcl`

```
##load design and upf##
check_lp -stage design
report_violations -app LP # note "b"
merge_database -session run1
report_violations -app LP # note "c"
```

At `note "a"`, some number of violations are reported and at `note "b"`, new violations are reported. When you run the `merge_database` command before `note "c"`, the `report_violations -app LP` command reports a union of both sets of violations. All the duplicate violations appear only once.

The following is the output from `run1`, before merging:

```
-----
Tree Summary
-----
Severity Stage      Tag           Count
----- -----
error    PG          PG_DOMAIN_CONN   1
error    PG          PG_PIN_UNCONN    66
error    PG          PG_STRATEGY_CONN 2
----- -----
Total                            69
```

The following is the output from `run2`, after merging.

```
-----
Tree Summary
-----
Severity Stage      Tag           Count
----- -----
error    Design     ISO_CONTROL_CONN 1
error    Design     ISO_CONTROL_INVERT 1
error    Design     ISO_BUFINV_STATE 6
error    Design     ISO_INPUT_STATE   5
error    Design     ISO_OUTPUT_STATE 2
warning  Design     ISO_DATA_BLOCKED 1
warning  Design     ISO_DATA_UNCONN 4
warning  Design     ISO_INST_NOCROSS 5
warning  Design     ISO_INST_REDUND 1
warning  Design     ISO_LOCATION_WRONG 1
warning  Design     ISO_STRATEGY_UNUSED 8
----- -----
```

```
Total 35
Here is the output from run2, after merging.
-----
Tree Summary
-----
Severity Stage Tag Count
-----
error Design ISO_CONTROL_CONN 1
error Design ISO_CONTROL_INVERT 1
error Design ISO_BUFINV_STATE 6
error Design ISO_INPUT_STATE 5
error Design ISO_OUTPUT_STATE 2
error PG PG_DOMAIN_CONN 1
error PG PG_PIN_UNCONN 66
error PG PG_STRATEGY_CONN 2
warning Design ISO_DATA_BLOCKED 1
warning Design ISO_DATA_UNCONN 4
warning Design ISO_INST_NOCROSS 5
warning Design ISO_INST_REDUND 1
warning Design ISO_LOCATION_WRONG 1
warning Design ISO_STRATEGY_UNUSED 8
-----
Total 104
```

### 3.7.3.1 Limitations

- ❖ The `merge_database` command does not retain waived violations from the saved database. The merged database will lose any waivers applied for violations in the individual saved database.
- ❖ The `merge_database` command does not support merging of VC Formal report databases. This command gives an error when the `fml_mode_on` application variable is set to true.

### 3.7.3.2 Reusing the Results from Previous Run

VC LP provides the `diff_database` command to read the violations from the disk of a previous run in the current run. For each violation which is common to both runs, the command either deletes the violation from the current run, or adds a waiver for the violation in the current run. The result is a new, smaller database where the current run contains only new violations, not found in the previous run.

#### Syntax

```
%vc_static_shell> diff_database -help
Usage: diff_database      # Subtracts violations in report database of a saved run from
the current run
      [-raw <input_filename>]
                           (Raw message file to subtract messages)
      [-session <session name>]
                           (Name of old style saved session to subtract messages)
      [-checkpoint <session name>]
                           (Name of checkpoint session to subtract messages)
      [-remove]
                           (Remove messages found in both databases)
      [-waive]
                           (Waive messages found in both databases)
      [-app <app>]
                           (Application name, default LP)
```



#### Note

Either of the `-remove` or `-waive` option are mandatory, to tell what operation should be performed on the violations in both databases.

Where:

- ❖ raw <input\_filename>: Specify the message file which contains the violations of a previous run that must be removed in the current run.
- ❖ session <session name>: Specify the name of the saved session which contains the violations of a previous run that must be removed in the current run.
- ❖ checkpoint <session name>: Specify the name of the checkpoint session which contains the violations of a previous run that must be removed in the current run.
- ❖ remove: Specify this option to remove the duplicate violations in the new report.
- ❖ waive: Specify this option to create a waiver named *duplicate*. This waiver is created listing each violation matching the previous run, using the violation ID and not the signature, so that waiver should not be stored for use in any future runs.
- ❖ app <app>: Specify the application name to perform the `diff_database` operation. If this option is not specified, VC LP is considered.

### 3.7.3.2.1 Use Models

The previous database can be referred in the following methods.

- ❖ Database saved using the `save_db` command can be compared using the `-raw` switch.

```
check_lp -stage { upf design } -force
report_violations -app LP -file run1.rpt
  save_db
    file copy -force vcst_rtdb/.internal/svi/report.db run1.db (saved db file should be
      copied to the run location)
  check_lp -stage design -force
  report_violations -app LP -file run2.rpt
  diff_database -raw run1.db -remove
```

- ❖ Database saved using check pointing can be compared using the `-checkpoint` switch.

```
# Run 1. Create database with pg violations
  check_lp -stage design -force
  report_violations -app LP -file run1.rpt
  save_db
  checkpoint_session -session check1
# Run 2. Create database with design violations and merge
  check_lp -stage all -force
  report_violations -app LP -file run2.rpt ;# fake out verbose
  diff_database -checkpoint check1 -waive
```

- ❖ Database saved using `save_session` can be compared using the `-session` switch.

```
# Run 1. Create database with pg violations
  check_lp -stage design -force
  report_violations -app LP -file run1.rpt
  save_session -session run1
# Run 2. Create database with design violations and merge
  check_lp -stage upf -force
  report_violations -app LP -file run2.rpt
  diff_database -session run1
```

### 3.7.3.2.2 Example

The following is an example of a database in the previous run:

Severity	Stage	Tag	Count
-----	-----	-----	-----

error	UPF	ISO_STRATEGY_CTRL_UNCONN	3
error	UPF	ISO_SUPPLY_UNAVAIL	2
error	UPF	UPF_BIAS_MISSING	6
error	UPF	UPF_STRATEGY_RESOLVE	9
error	Design	ISO_INST_MISSING	8
warning	UPF	ISO_MAP_MISSING	3
warning	UPF	ISO_STRATEGY_MULTIPLE	2
warning	UPF	ISO_STRATNODE_UNDRIVEN	4
warning	UPF	PSW_SUPPLY_UNAVAIL	2
warning	UPF	UPF_PORT_UNCONSTRAINED	1
warning	UPF	UPF_POWER_DIFF	36
warning	Design	DESIGN_POWER_DIFF	36
warning	Design	ISO_STRATEGY_UNUSED	11
<hr/>			<hr/>
Total			123

The following is an example of a new database in the current run:

Severity	Stage	Tag	Count
error	UPF	ISO_STRATEGY_CTRL_UNCONN	3
error	UPF	ISO_SUPPLY_UNAVAIL	2
error	UPF	UPF_BIAS_MISSING	6
error	UPF	UPF_STRATEGY_RESOLVE	9
error	PG	PSW_ACK_UNDRIVEN	1
error	PG	PSW_STRATEGY_UNUSED	1
warning	UPF	ISO_MAP_MISSING	3
warning	UPF	ISO_STRATEGY_MULTIPLE	2
warning	UPF	ISO_STRATNODE_UNDRIVEN	4
warning	UPF	PSW_SUPPLY_UNAVAIL	2
warning	UPF	UPF_PORT_UNCONSTRAINED	1
warning	UPF	UPF_POWER_DIFF	36
warning	PG	PG_SUPPLY_NONET	5
warning	PG	PG_SUPPLY_NOPORT	4
<hr/>			<hr/>
Total			79

The following is the output when you use `diff_database -waive` command:

Severity	Stage	Tag	Count	Waived
error	UPF	ISO_STRATEGY_CTRL_UNCONN	0	3
error	UPF	ISO_SUPPLY_UNAVAIL	0	2
error	UPF	UPF_BIAS_MISSING	0	6
error	UPF	UPF_STRATEGY_RESOLVE	0	9
error	PG	PSW_ACK_UNDRIVEN	1	0
error	PG	PSW_STRATEGY_UNUSED	1	0
warning	UPF	ISO_MAP_MISSING	0	3
warning	UPF	ISO_STRATEGY_MULTIPLE	0	2
warning	UPF	ISO_STRATNODE_UNDRIVEN	0	4
warning	UPF	PSW_SUPPLY_UNAVAIL	0	2
warning	UPF	UPF_PORT_UNCONSTRAINED	0	1
warning	UPF	UPF_POWER_DIFF	0	36
warning	PG	PG_SUPPLY_NONET	5	0
warning	PG	PG_SUPPLY_NOPORT	4	0
<hr/>			<hr/>	<hr/>

Total		11	68
The following is the output when you use <code>diff_database -remove</code> command:			
error	PG	PSW_ACK_UNDRIVEN	1
error	PG	PSW_STRATEGY_UNUSED	1
warning	PG	PG_SUPPLY_NONET	5
warning	PG	PG_SUPPLY_NOPORT	4
<hr/>			
Total		11	

### 3.7.4 Reducing Compile Time in Subsequent check\_lp Runs

VC LP compiles the design and UPF before performing LP sign-off checks. Generally, time taken to compile design is much longer than the time taken to compile the UPF as the design data is usually much more than UPF data. However, not everything in the design contribute to the LP sign-off checks.

VC LP can identify these modules (or instances) which are not relevant for the UPF and which do not have any feed-through crossovers in it. These modules and instances can be saved in a Tcl file. In the subsequent runs, when this Tcl file is sourced, VC LP automatically black boxes such modules (or instances), and does not compile them, thus reducing the compile time and improving the overall turnaround time.

#### Steps to Reduce Compile Time

To reduce the compile time of the subsequent `check_lp` runs:

- ❖ In the first run of `check_lp`, you must compile the design and UPF, and set the `enable_auto_bbox_identification` application variable and the `identify_lp_blackboxes` command.

*#The following application variable enables automatic black boxing of modules/instances which are not relevant for the UPF.*

```
set_app_var enable_auto_bbox_identification true
read_file {design file} -top top -netlist
load_upf <upf file name>
# The following command writes all the modules/instances which are not relevant for the UPF in the Tcl file,
# which must be used in the subsequent runs.
identify_lp_blackboxes -file out_bbox.tcl
```

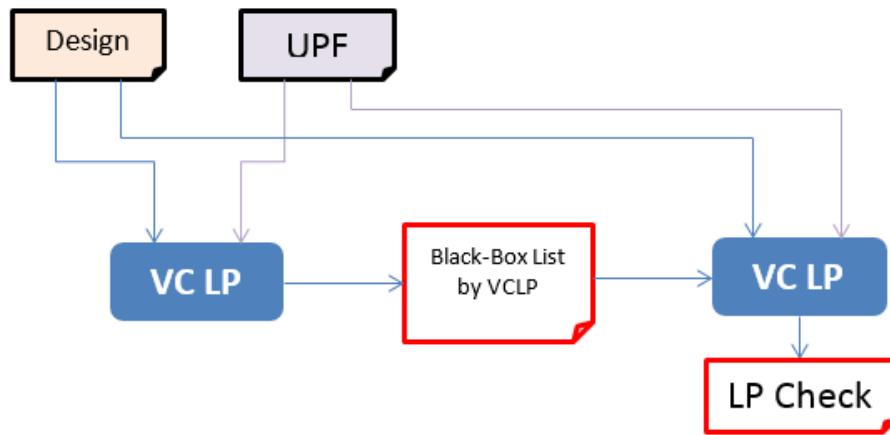
- ❖ In the subsequent runs, you must compile design, UPF and Tcl file generated by VC LP.

*#In the subsequent runs, source the Tcl file generated in the first run.*

```
source out_bbox.tcl
read_file {design file} -top top -netlist
load_upf {upf file}
```

#### Note:

- ❖ If the module or instance has CSN/SRSN or any relation with the UPF, it is not marked as a blackbox.
- ❖ If the module or instance having LS, ISO or ELS cell, it is not marked as a black box.
- ❖ If the module of the instances to be black boxed are uniquely instantiated, that is, that instance is the single instance of the module. In such cases, the module itself is black boxed.

**Figure 3-5 Steps for Reducing Compile Time**

### 3.7.4.1 Limitation

- ❖ For PG\_NETLIST design, make sure corresponding CSN/SRSN is present in the UPF for PG connections. In the absence of any UPF constraints, this feature might not work properly.

## 3.8 Support for Library Checks

VC LP supports a flow where you can provide both the IP RTL and IP .db, and perform all the low power checks based on the IP RTL.

By default, if both the RTL module definition and liberty definition is found, VC LP performs checks on the library. When the `prefer_lib_over_rtl` application variable is set to false, a full RTL design is expected, without DB. If the RTL definition is not found, then the module is black boxed, and the following message is reported:

*Warning-[SM\_URMI] Unresolved module instance in design.*

VC LP performs the following two checks in a single run with the `lib_instance_list` application variable, when the `prefer_lib_over_rtl` application variable is set to false.

- ❖ Full checks (check\_lp) on RTL designs (TOP RTL and IP RTL) with UPF (TOP upf and IP UPF).
- ❖ Some UPF consistency liberty checks on the given IP .db with IP UPF

### 3.8.1 Use Model

The `prefer_lib_over_rtl` application variable must also be set to false, as VC LP reads in both the RTL and liberty definitions. The RTL definition is taken to build the design, and the database is also read in.

Use the `lib_instance_list` application variable before `load_upf` and `read_upf` to specify the list of elements for which liberty checks must be performed.

```
% set_app_var prefer_lib_over_rtl false
## Then the common flow to load design/upf and do check
% set_app_var search_path <the_lib_paths>
% set_app_var link_library <the_libs>
% read_file <the_design_file> -top <top>
% set_app_var lib_instance_list "name_or_object_list"
% read_upf <the_upf_file>
```

```
% check_lp
```

### 3.8.2 Tags Impacted in this Flow

The *RtIVsLiberty* debug field is reported for the following tags when the `lib_instance_list` application variable is specified with the `prefer_lib_over_rtl` application variable set to false. In the common ETM flow checks, the *RtIVsLiberty* debug field is not reported for these violations.

- ❖ ISO\_MACRO\_INCONSISTENT
- ❖ UPF\_PORT\_OVERRIDE
- ❖ UPF\_PORT\_MISSING
- ❖ UPF\_PORT\_DIRECTION
- ❖ UPF\_PORT\_NET



#### Note

For more information on these tags, refer to the man page in the `vc_static_shell`.

### 3.8.3 New Tags Introduced

The following new tags are introduced when the liberty checks are performed in RTL.

- ❖ UPF\_PORT\_EXTRA
- This violation is reported when a supply port is defined/specifies in the UPF, but is not defined as a pg\_pin liberty.
- ❖ UPF\_OBJECT\_UNDEFINDED

This violation is reported when UPF objects are not defined in the database.

## 3.9 Checking Reports

Use the `report_violations -app LP` command to obtain a report of all the messages generated after completing the low power analysis on the design.

### Syntax

```
vc_static_shell>report_violations -app LP -help
Usage: report_violations -app LP # Reports low power check information
      [-no_summary]          (Suppresses summary information)
      [-list]                (List all messages in simple form)
      [-verbose]              (List all messages in detail form)
      [-limit <count>]        (Limit the number of output records per tag)
      [-include_waived]       (Include waived messages in the report)
      [-include_compressed]   (Include compressed messages in the report)
      [-only_waived]          (Report on waived messages)
      [-all_tags]              (Include all tested tags)
      [-tag <tag>]            (Select violations based on tag)
      [-waived <list>]         (Select violations based on waiver name)
      [-id <tag>]              (Select violations based on IDs)
      [-stage <stage>]         (Select violations based on stage:
                                Values: Debug, all, design, diff,
                                library, pg, upf)
      [-severity <list>]        (Select violations based on severity:
                                Values: all, error, info, warning)
      [-filter <expression>]    (Select violations based on expression)
```

```

[-regexp]           (Indicates filter expression type to be regular expression
(default glob-style))
[-nocase]          (Filter expressions ignore case when matching string
values)
[-file <filename>] (Write the results to the designated file)
[-append]          (Append results to the designated file)
[-compressions]    (Include all violations in the same compression set(s).)
[-skip_full_path_for_waiver_file]
                   (Displays only base name for waiver file)
[-add_markers]     (Display BEGIN_LP,END_LP markers in verbose mode)
[-display_severity] (Dump severity info for each record in verbose mode)

```

## Use Model

The following is an example output of the `report_violations -app LP` command.

```

-----
Management Summary
-----
Stage Family Errors Warnings Infos
----- -----
UPF Isolation 3 0 0
UPF LevelShifter 5 0 0
Design Isolation 6 0 0
Design LevelShifter 5 0 0
Design SignalCorruption 3 2 0
----- -----
Total 22 2 0
-----
Tree Summary
-----
Severity Stage Tag Count
----- -----
error UPF ISO_CONTROL_STATE 1
error UPF ISO_STRATEGY_MISSING 2
error UPF LS_STRATEGY_MISSING 5
error Design CORR_CONTROL_STATE 3
error Design ISO_CONTROL_INVERT 1
error Design ISO_ENABLE_UNCONN 1
error Design ISO_INST_NOPOL 2
error Design ISO_SINK_STATE 2

```

## 3.10 Saving and Restoring Sessions Using `save_session` and `-restore`

VC LP enables you to save a compiled design and then restore it whenever you want to. This enables you to save considerable runtime when multiple sessions are used to analyze a design. For example, if you save and exit one session, you can display schematic paths after reloading the design database. Save/restore allows a session to be restarted without incurring the burden of reloading the design. This results in a significant performance boost on subsequent design reloads.

Another benefit comes from the ability to fork a session for exploring several different scenarios. In this model, you can load your design, save the session, make copies of the session, and then for each variation, restore a copy and continue with your tests. VC LP does not automatically save the session when you quit.

When you quit from the `vc_static_shell`, the current session results is not automatically saved. If you wish to save the session setup and run data, use `save_session` command.

```
%vc_static_shell> save_session
```

The information from run is then saved under *vcst\_rtdb* in the current working directory. The following message is printed on the screen.

```
[Info] SAVING_SESSION: Saving session (vcst_rtdb) .
[Info] SAVE_I_FACET: 'FV' saved.
[Info] SAVE_I_FACET: 'design' saved.
```

The restore option works in conjunction with the *-session* command line option. When you use the *-session* command line option, the restore options apply to this session rather than the default *vcst\_rtdb* session directory. Essentially, this creates unique multiple sessions and reloads them whenever you want.

### 3.10.1 Saving a Session Automatically

The auto save feature (save when quit) is disabled by default. You can set *save\_session\_default* to true to enable this feature. Otherwise, you can always use *save\_session* to save the session to a given directory at any given stage of the run.

#### Use Model

```
%vc_static_shell -restore
```

This invocation restores both the design and the violation databases.

```
%vc_static_shell -no_restore
```

This invocation deletes an existing *vcst\_rtdb* directory and creates a new one for a new session.

The restore option works in conjunction with the *-session* command line option. When you use the *-session* command line option, the restore options apply to this session rather than the default *vcst\_rtdb* session directory. Essentially, this will help you create unique multiple sessions and reload them whenever you want. For instance, the use model for a particular session would appear as:

```
%vc_static_shell -restore -session <session_name>
```

For RTL designs (*analyze/elaborate*) the save is default. The design is saved as a natural consequence of the current design flow. For netlist-based designs (*read\_file*), you have the option to create a session copy for subsequent restores or not. The *save\_netlist\_copy* application variable enables you to save your netlist design. This is time consuming, especially for large designs. You can set this Tcl variable to false any time prior to the *read\_file* command if you do not want to save a copy of your netlist design.

#### Without *-session <session\_name>* option

VC LP creates *vcst\_rtdb* directory if *-session* option is not given.

```
%vc_static_shell
%vc_static_shell>set search_path "."
%vc_static_shell>set link_library "my_lib.db"
%vc_static_shell>read_file -format verilog -top top -netlist "design.v"
%vc_static_shell>read_upf "temp.upf"
%vc_static_shell>check_lp -stage upf
%vc_static_shell>check_lp -stage design
%vc_static_shell>check_lp -stage pg
%vc_static_shell>report_violations -app LP -verbose -file report_lp.txt
%vc_static_shell>quit
```

```
[Info] DESIGN_SAVED: Design has been successfully saved.
```

```
%vc_static_shell -restore
```

```
[Info] RESTORING_SESSION: Restoring session (vcst_rtdb).  
1  
%vc_static_shell>
```

After restoring the VC LP session, you can use `view_activity`, `view_schematic`, `report_violations -app LP`, `waive_lp`, `compress_lp` and design related query commands.

#### With `-session <session_name>` option

VC LP creates `<session_name>.rtdb`.

```
%vc_static_shell -session temp  
%vc_static_shell>set search_path "."  
%vc_static_shell>set link_library "temp.db"  
%vc_static_shell> read_file -format verilog -top top -netlist "design.v"  
%vc_static_shell>read_upf "temp.upf"  
%vc_static_shell>check_lp -stage upf/design/pg  
%vc_static_shell>report_violations -app LP -verbose -file report_lp.txt  
%vc_static_shell>quit  
[Info] DESIGN_SAVED: Design has been successfully saved.  
%vc_static_shell -restore -session temp
```

```
[Info] RESTORING_SESSION: Restoring session (temp_rtdb).  
%vc_static_shell>
```

After restoring temporary session, you can use `view_activity`, `view_schematic`, `report_violations -app LP`, `waive_lp`, `compress_lp` and design related query commands.



**Note**  
The crossover and PG tracing data is saved additionally when you perform save and restore. User does not need to execute "generate crossover" additionally when after restoring the session.

### 3.10.2 Deleting Files After Saving a Session Automatically

After a save session, you can enable the VC Static tool to automatically delete files in the folder to save disk space. To automatically delete the files, set the `cleanup_on_exit` application variable to true.

```
set_app_var cleanup_on_exit true
```

This application variable is hidden, and by default, it is set to false.

The following files are deleted from `vcst_rtdb` the folder when you set the `cleanup_on_exit` application variable to true (the list may vary depending on the files present in the tcl script, and can change with each release):

- ❖ messages
- ❖ tests
- ❖ traces
- ❖ read\_only
- ❖ logs/db\_read.log
- ❖ logs/db\_link.log
- ❖ logs/db\_new\_ports.log
- ❖ logs/vcst\_command.log
- ❖ reports/check\_architecture\_coverage.txt

- ❖ logs/covMerge.rpt

### 3.10.3 Deleting Files After Saving a Session Manually

You can also manually delete all the files in the vcst\_rtdb folder. However, you should ensure that you do not delete the vcst\_rtdb/.archive file after the save session, as the vcst\_rtdb/.archive/vsi.tar file is used for the restore session. The vcst\_rtdb/.archive/vsi.tar file is the only file that is used for the restore session.

## 3.11 Saving and Restoring Sessions Using CR Technology

You can save a session at some point for the purpose of restarting the session later from that point. This enables the restore to be much faster than the original execution time it took to reach that point before saving. VC LP supports the save and restore using the process Checkpoint/Restart (CR) technology to improve its performance. The process checkpoint/restart (CR) is a technology which creates a snapshot (checkpoint) of the process image on disk and is later able to reload the image to the memory and restart the process at the same execution point. The CR technology is a very fast save and restore solution, as it dumps and reads the bulk data of the process to and from disk without any kind of transformation or computation.

### 3.11.1 Use Model

The following example illustrates how you can save and restore a session based on CR technology in VC Static.

To save the session at `read_upf` and `check_lp -stage upf`:

```
analyze -format sverilog " top.sv "
elaborate top

read_upf top.upf
checkpoint_session -session SessionReadUPF

check_lp -stage upf
checkpoint_session -session SessionReadUPF1

report_violations -app LP -verbose -file report.log
quit
```

To restart the session from `SessionReadUPF` checkpoint:

```
%vc_static_shell> restart_session -session SessionReadUPF
```

### 3.11.2 New commands Introduced

#### 3.11.2.1 The `checkpoint_session` Command

Use the `checkpoint_session` command to save a session. This command saves the session in the `<sessionName>_cpdb/checkpoints` directory.

##### Syntax

```
vc_static_shell> checkpoint_session -help
Usage: checkpoint_session      # Dumps the process image of a session
      [-session session_name]
                           (checkpoint this session under a specified name)
```

You can add multiple checkpoints during a single run at different stages, but these checkpoints must have different sessionName specified to avoid over writing of the files. If two `checkpoint_session` commands

are run one after another with the same session\_name, then the output of the second command will overwrite the first command output.

### Use Model

```
vc_static_shell>checkpoint_session [-session <name>]
```

The -session <sessionName> argument is mandatory. If this argument is not given, the following error is displayed.

```
vc_static_shell> checkpoint_session
```

[Error] SAVE\_SESSION\_MISSING\_ARG: Save session (vcst\_cpdb) failed because required argument -session <sessionName> not specified.

Error: 0

Use error\_info for more info. (CMD-013)

### 3.11.2.2 The restart\_session Command

Use the restart\_session command to restart the session from the process where it was stopped in the checkpoint\_session command.

The restart\_session command restarts the session from the <sessionName>\_cpdb/checkpoints directory.

The session name of the session from which restart\_session command is run cannot be the same as the checkpointerd <sessionName>. Basically no restart\_session should be run from the checkpointerd sessionName\_cpdb.

### Syntax

```
vc_static_shell> restart_session -help
Usage: restart_session      # Restarts process from a dumped image
       [-session session_name]
                           (restart the process from the image of the given name)
       [-file file_name]   (execute commands from this file once restart iscomplete)
```

### Use Model

```
%vc_static_shell>restart_session -session <sessionName> -file <file_name>
```

The -session <sessionName> argument is mandatory. If this argument is not given, the following error is displayed.

```
vc_static_shell> restart_session
```

[Error] RESTORE\_SESSION\_MISSING\_ARG: Restore session (vcst\_cpdb) failed because required argument -session <sessionName> not specified.

Error: 0

Use error\_info for more info. (CMD-013)

### 3.11.2.2.1 Executing Restart flow in batch mode

The command for running vc\_static\_shell:

```
vc_static_shell -file xyz.tcl
```

where xyz.tcl contains the following:

```
restart_session -session abc -file <execute_these_after_restart>.tcl
```

### 3.11.2.2.2 Limitation

- ❖ Checkpoint and restart feature is supported in the `no_ui` mode with the following limitations:
  - ◆ The time taken for `checkpoint_session` is more as it includes the times taken by LZ4 compression, dumping of process image, and tarring of the `vcst_cpdb`.
  - ◆ The RAM and SWAP of the machine on which the `checkpoint_session` and `restart_session` is running on must be greater than twice the peak memory size of the SVI for the test case.

### 3.11.3 New Error Messages Introduced

The following new error messages are introduced as part of this support.

#### 3.11.3.1 RESTORE\_SESSION\_FAILED\_BUILD\_MISMATCH

Restart can only successfully happen if the invoking binary is exactly identical to the check pointing binary. In cases when these two are not exactly identical, the `RESTORE_SESSION_FAILED_BUILD_MISMATCH` error is reported. This can happen when checkpoint build and restart build are different in following criteria:

- ❖ Opt and debug build
- ❖ 32 and 64 bit

#### 3.11.3.2 RESTORE\_SESSION\_FAILED\_RUN\_ARGUMENT\_MISMATCH

Restart session cannot happen successfully if the following run-time option is different between the checkpoint run and the restart run.

- ❖ `ui` and `no_ui` run

### 3.11.4 GUI support

The checkpoint/restart flow is supported for the GUI mode. You can use `view_activity`, `start_gui`, `verdi` after restart and continue with debug.

## 3.12 Support for Auto-Saving and Auto-Restarting Sessions

You can auto-save a checkpoint for the session while exiting the run and restart the session from the same point in next run.

- ❖ **To auto save a session:**

Set the following application variable in the Tcl file:

```
set_app_var checkpoint_on_exit true
```

When you set the `checkpoint_on_exit` application variable to true, and exit the VC Static session at any point, the session information gets saved in the following session directory:

```
./cp/auto_cpdb/
```

The `auto_cpdb` directory is always in sync with session directory. The `checkpoint_on_exit` application variable is disabled by default.

- ❖ **To restart the session:**

- ◆ In the Verdi first mode

```
$vc_static_shell -verdi -f <script.tcl>
```

- ◆ In the VC Static first mode:

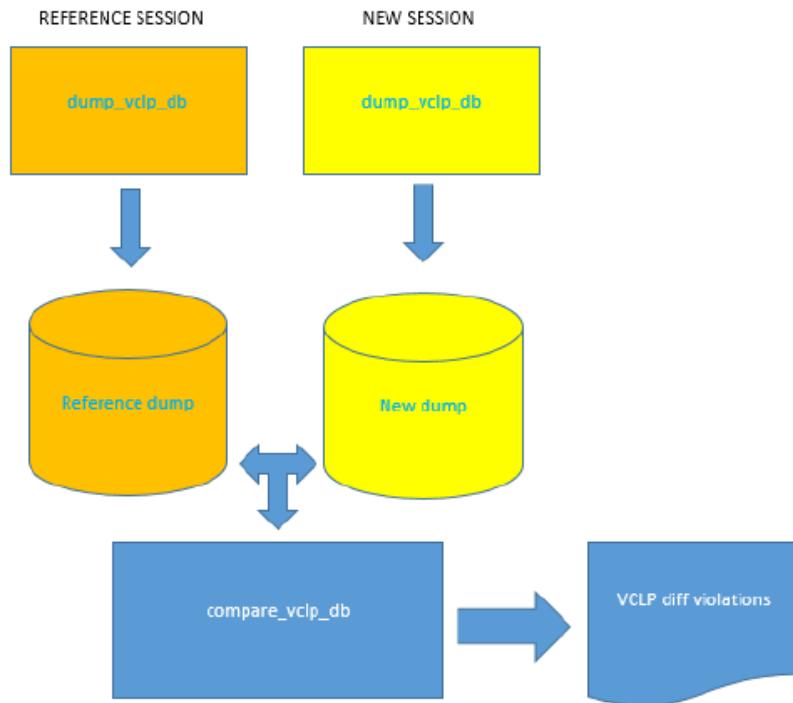
```
$vc_static_shell -f script.tcl
```

## 3.13 Support for Comparing Results Of Two VC LP Sessions

VC LP enables you to find the difference between two combinations of design and UPF across two sessions.

[Figure 3-6](#) provides the use model for comparing the output from two sessions. In the [Figure 3-6](#), the reference dump refers to the results saved from the reference session. The new dump refers to the results from the new session with which you want to compare the reference session results.

**Figure 3-6 Comparing Results of Two VC LP Sessions**



### 3.13.1 Use Model

Perform the following steps to get the difference between two VC LP sessions:

- ❖ Save the reference session's data using the `dump_vcip_db` command.
- ❖ Save the new session's data using `dump_vcip_db` in a different location.
- ❖ In the same session or new session, use the `compare_vcip_db` command to get the differences between the two sessions as VC LP violations.
- ❖ Use the `report_violations -app LP` command to get the DIFF\* violations across two sessions.

### 3.13.2 The `dump_vcip_db` Command

Use the `dump_vcip_db` command to save the session's data.

#### Syntax

```
%vc_static_shell> dump_lp_db -help
```

```

Usage: dump_lp_db    # dump different components(crossover, strategies, root supply, upf
datamodel) of lowpower in json format
      [-component <list of types>]
          (cell types:
           Values: policy, rootsupply,
           rootsupplypin, upfdatamodel, xover,
           xover_refined)
      [-types <list of types>]
          (cell types:
           Values: iso, ls, psw, rep, ret)
      [-dumpheader]           (Dump date and product version along with json dump)

```

### 3.13.3 The compare\_vclp\_db Command

Use the `compare_vclp_db` command to get the differences between two sessions.

#### Syntax

```

%vc_static_shell> compare_vclp_db -help
Usage: compare_vclp_db    # compares vclp_db from two different sessions
      -current <current dump folder name>
          (current VCLP db dump folder)
      -original <original dump folder name>
          (original VCLP db dump folder)

```

You cannot use the `compare_vclp_db` command in the same session where the `compare_upf` command is used. This is because the `compare_vclp_db` and `compare_upf` can produce the same category of violations.



**Note**  
The `compare_vclp_db` command required the VC-STATIC-LP-DEBUG license.

### 3.13.4 The lp\_upf\_diff\_ignore\_patterns Application Variable

You can use the `lp_upf_diff_ignore_patterns` application variable to filter the results of the `compare_vclp_db` command.

In the `lp_upf_diff_ignore_patterns` application variable, you can provide a list of wild card patterns for the violations that must be ignored while comparing the results of the two sessions.

#### Example

Consider that an instance/module existing in both the sessions is black-boxed in only one of the sessions, then this can lead to many violations in the output of the comparison results. You can anticipate such differences which can act as noise before performing the comparison. To avoid such unnecessary differences and noise in the output, you can give wild card patterns to be ignored while comparing the results in the `lp_upf_diff_ignore_patterns` application variable.

### 3.13.5 Limitations Comparing Results Of Two VC LP Sessions

The following are the limitations with the comparing results of two VC LP sessions

- ❖ This feature not supported when the designs are same and UPFs are different.
- ❖ The `compare_vclp_db` command cannot be used in the same session where `compare_upf` is used. This is because this can result in violation collision, since `compare_vclp_db` and `compare_upf` can produce the same category of violations.
- ❖ This feature is not supported in the black box flow.

## 3.14 Handling Encryption

The VC Static Platform lets you use third-party IP and RTL modules encrypted for IP protection. It accepts modules encrypted with encryption schemes such as:

- ❖ 128-bit advanced encryption standard (AES)
- ❖ IEEE Verilog standard 1364-2005 encryption (supports IEEE P 1735)
- ❖ GenIP
- ❖ DWIP encryption
- ❖ Synenc encryption

Support for encrypted modules, enables analysis (structural, formal) of designs that contain encrypted IP, without disclosing the IP content to the user. In order to protect the IP, the following restrictions apply in VC Static:

- ❖ Does not show the code of encrypted modules in violations, Tcl commands, GUI, tool messages.
- ❖ Does not dump waveforms for encrypted modules.
- ❖ Tcl commands do not find nets, cells or instances inside protected modules unless exact names of the encrypted modules are specified.
- ❖ Coverage is not supported inside protected modules.
- ❖ AEPs are not generated for protected modules.
- ❖ All design traversal query commands stop at the encrypted module boundary.
- ❖ If an encrypted property is falsified, only the property name is printed in the error log, but not the implementation of the property.

### 3.14.1 Naming Encrypted Modules

#### 3.14.1.1 Name Output

The following is the name generation scheme for an encrypted design element:

- ```
<hierarchy_up_to_boundary_of_encryption>/<mangled_name_of_enc_hier>/<mangled_name_of_design_element>
```
- ❖ For example, if the full hierarchy name is a/b/c/d/e/<design\_element>, and the hierarchy from instance c is encrypted, then the name generated is a/b/c/<encrypted\_xxxx>/<encrypted\_yyyy>.
  - ❖ For a property, the name is printed in plain-text to let you know which property failed. Hence the naming convention for hierarchical property name is as follows:  

```
<plain_text_full_hierarchy_till_enc_hierarchy>/<mangled_name_of_enc_hier>/<plain_text_property_name>
```

    - ◆ For example, if a property name is a/b/c/d/e/P1 and the hierarchy from the instance c is encrypted, then the name generated for the property is:  
a/b/c/<mangled\_name\_of\_enc\_hier>/P1.

In order to point to the right place in the design in an encrypted zone, the source location (<filename>, <linenum>) is appended to the mangled name. This does not compromise any intellectual property. You can look into the file only when you get the unencrypted files. However, you can point to the IP provider with that information.

### 3.14.1.2 Name Input

You can use exact (non-encrypted) names of encrypted design objects to refer to encrypted design elements in following places:

- ❖ Tcl query commands
- ❖ GUI query for design element using name
- ❖ UPF reference to design element
- ❖ SDC reference to design elements
- ❖ Property bind to design elements

Wildcard reference to encrypted design elements is not supported and results in *design object not found* warning messages.

If the design hierarchy is `a/b/c/d/e/<design_element>` and if only `c` is encrypted, any design element reference inside `a/b/c/*` is treated as encrypted.

### 3.14.1.3 Support for Specifying Threshold Counts for \*\_OUTPUT\_UNCONN Violations

You can specify a threshold count for the following tags to avoid noise.

- ❖ ISO\_OUTPUT\_UNCONN
- ❖ LS\_OUTPUT\_UNCONN
- ❖ RET\_OUTPUT\_UNCONN
- ❖ AOB\_OUTPUT\_UNCONN

You can specify separate thresholds for each of these violations. These violations are reported only if the violation count exceeds the specified threshold value. Use the following command to specify the thresholds for each of these violations.

#### Syntax

```
lpViolationThreshold -tag <tag> -limit <limit>
```

Where:

- ❖ `-tag <tag>`: Specify any of the ISO/LS/RET/AOB\_OUTPUT\_UNCONN indicating which violation is required to be limited by a threshold value.
- ❖ `-limit`: Specify an integer indicating the threshold value.

The command is required to be used before the functional checks relating to UNCONN violations are performed, that is, before `check_lp -stage design` is performed. The command does not perform any database operation on the violation database, but prevents the violation from being even created during the checks.

- ❖ If an invalid or an unsupported tag is specified in the `-tag` option, the following error is reported, and VC LP ignores the command and proceeds further.

```
ID      : TAG_INVALID_FOR_THRESHOLD,
CODE_ID : TAG_INVALID_FOR_THRESHOLD,
Severity : Error,
Primary : "Tag %s specified by the user is not supported to be applied with a
threshold limit using pViolationThreshold command",
Secondary : "Supported tags are ISO_OUTPUT_UNCONN, LS_OUTPUT_UNCONN, RET_OUTPUT_UNCONN,
AOB_OUTPUT_UNCONN"
```

- ❖ The default value of `limit` will be 0, meaning that the current behavior will be preserved. If the given limit is an unsupported like a negative number, an error message is reported.

```
ID      : LIMIT_INVALID_FOR_THRESHOLD
CODE_ID : LIMIT_INVALID_FOR_THRESHOLD
Severity: Error
Primary : Limit value -2 specified by the user is invalid for lpViolationThreshold
command
Secondary : Please provide 0 or a positive integer as the limit value for thresholding a
tag
```

- ❖ Applying thresholds to tags might make violations of certain kind disappear from the `reportViolations -app LP` output. The following message is reported issued whenever a violation of a kind is suppressed due to the application of a threshold.

```
ID      : TAG_THRESHOLDED,
CODE_ID : TAG_THRESHOLDED,
Severity : Warning,
Primary : "Tag %s has been thresholded with a limit of %d and will be missing on the
violation reports generated by the tool",
Secondary : "Please avoid using the command lpViolationThreshold or set back the
threshold limit to 0 if you wish to prevent this from happening"
```

### 3.14.1.4 Specifying Threshold Counts for \*\_OUTPUT\_UNCONN Violations

You can specify a threshold count for the following tags to avoid noise.

- ❖ ISO\_OUTPUT\_UNCONN
- ❖ LS\_OUTPUT\_UNCONN
- ❖ RET\_OUTPUT\_UNCONN
- ❖ AOB\_OUTPUT\_UNCONN

You can specify separate thresholds for each of these violations. These violations are reported only if the violation count exceeds the specified threshold value. Use the following command to specify the thresholds for each of these violations.

#### Syntax

```
lpViolationThreshold -tag <tag> -limit <limit>
```

Where:

- ❖ `-tag <tag>`: Specify any of the ISO/LS/RET/AOB\_OUTPUT\_UNCONN indicating which violation is required to be limited by a threshold value.
- ❖ `-limit`: Specify an integer indicating the threshold value.

The command is required to be used before the functional checks relating to UNCONN violations are performed, that is, before `check_lp -stage design` is performed. The command does not perform any database operation on the violation database, but prevents the violation from being even created during the checks.

- ❖ If an invalid or an unsupported tag is specified in the `-tag` option, the following error is reported, and VC LP ignores the command and proceeds further.

```
ID      : TAG_INVALID_FOR_THRESHOLD,
CODE_ID : TAG_INVALID_FOR_THRESHOLD,
Severity : Error,
Primary : "Tag %s specified by the user is not supported to be applied with a
threshold limit using pViolationThreshold command",
```

Secondary : "Supported tags are ISO\_OUTPUT\_UNCONN, LS\_OUTPUT\_UNCONN, RET\_OUTPUT\_UNCONN, AOB\_OUTPUT\_UNCONN"

- ❖ The default value of `limit` will be 0, meaning that the current behavior will be preserved. If the given limit is an unsupported like a negative number, an error message is reported.

ID : LIMIT\_INVALID\_FOR\_THRESHOLD

CODE\_ID : LIMIT\_INVALID\_FOR\_THRESHOLD

Severity: Error

Primary : Limit value -2 specified by the user is invalid for `lpViolationThreshold` command

Secondary : Please provide 0 or a positive integer as the limit value for thresholding a tag

- ❖ Applying thresholds to tags might make violations of certain kind disappear from the `reportViolations -app LP` output. The following message is reported issued whenever a violation of a kind is suppressed due to the application of a threshold.

ID : TAG\_THRESHOLDED,

CODE\_ID : TAG\_THRESHOLDED,

Severity : Warning,

Primary : "Tag %s has been thresholded with a limit of %d and will be missing on the violation reports generated by the tool",

Secondary : "Please avoid using the command `lpViolationThreshold` or set back the threshold limit to 0 if you wish to prevent this from happening"

### 3.14.2 Tcl Query Command Support

The name matching Tcl query commands return encrypted objects only if the given name is an exact name of encrypted design object.

#### Example 1:

Suppose cell A/B is encrypted and A/C is not.

- ❖ `get_cells A/*` returns only A/C
- ❖ `get_cells A/B` returns A/B

#### Example 2:

Suppose instance hierarchy is TOP -> inst\_A -> inst\_B -> inst\_C -> inst\_D (modules A, B are encrypted).

```
get_ports inst_A(inst_B(inst_C(inst_D(<exact_name_of_signal> returns
inst_A/encrypted_xxxx/encrypted_yyyy
```

Connectivity related Tcl query commands adhere to the following rules:

- ❖ If the input signal is encrypted, the command does nothing and returns nothing.
- ❖ If the traverse gets to an encrypted signal, it stops and does not pass through the encrypted hierarchy and returns the path up until the boundary of the encryption.
- ❖ If the return object is encrypted, the command returns mangled names for encrypted design objects.

For example, consider a path A->B->C->D in the design and C is encrypted.

- ❖ `get_trace_paths -from A` returns path A->B
- ❖ `get_trace_paths -from C` returns no path
- ❖ `get_trace_paths -from B -to C` returns no path
- ❖ `get_trace_paths -from A -to D` returns no path

### 3.14.3 LP Specific Checks

LP checks are done on encrypted modules.

### 3.14.4 Encryption Support in the GUI

This section describes how the VC Static platform supports encryption in the GUI.

#### 3.14.4.1 Design Traversal / Schematic

- ❖ Elements within the encrypted module are not displayed; the rest of the elements are displayed as usual.
- ❖ Property pane shows cell type as *encrypted* and cell name as *encrypted\_xxxx* for encrypted modules.
- ❖ Expand path on signal connected encrypted object stops at the encryption boundary.
- ❖ Cross-probing to source window on encrypted objects does not display the source code.

#### 3.14.4.2 GUI Search Control Options

##### 3.14.4.2.1 Trace

If exact names of encrypted design objects are given the schematic shows the paths.

##### 3.14.4.2.2 Find

If GUI queries for design elements is within the encrypted part of a design, that design element is retrieved only if it's name matches exactly. Otherwise it is not retrieved.

If an element from an encrypted zone is displayed because its name matches exactly, you will not be allowed to expand further to include more elements from an encrypted zone.

#### 3.14.4.3 Source View

Source Viewer does not show any part of an encrypted module.

#### 3.14.4.4 Hierarchy View

- ❖ The hierarchy browser does not show any design hierarchy in a encrypted region of a design.
- ❖ Cross-probing to source window on mangled names of encrypted objects does not display the source code.
- ❖ Cross-probing to schematic on mangled names of encrypted hierarchy does not display the schematic.

### 3.14.5 Design Dumping

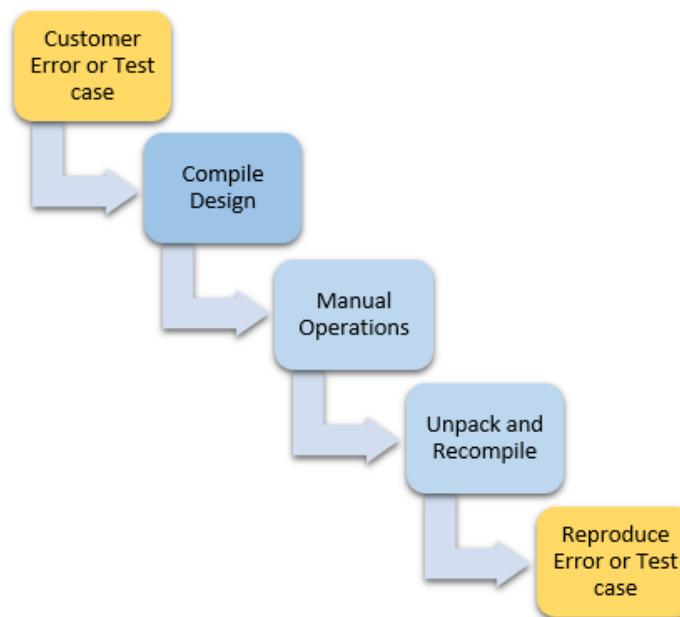
Tool generated files in the database directory (`vcst_rtdb`) do not display encrypted design objects.

### 3.14.6 Black Boxing

In case, the analysis on encrypted IP is not required, you can mark encrypted modules as black boxes, in which case, those parts of the design are not built.

## 3.15 Extracting Test Cases Automatically

You can automatically extract test cases in VC LP using the Automatic Test Extraction (ATE) flow. The ATE flow is a seamless flow which can be used to clone a customer designs and bring them up in-house, so that they can be used to debug issues, benchmark and so on.

**Figure 3-7 ATE Flow**

The test case capture capability (also known as ATE (Automatic Test-case Extraction)) provides a simple and convenient method of extracting, bundling and delivering a test case to the Synopsys support engineers for debugging or testing a case. By using this capability, you are freed from worrying about tasks like which files to include or exclude from the test case, how to bundle the files, how to ensure that the problem is reproducible.

To clone a design:

1. Set the SNPS\_VCS\_ATE\_COLLECT environment variable to the <capture\_directory\_path>.
2. Run the design with LP.
3. Go to the cloned design directory.

```
cd <clone-dir>
```

4. Run the source ate\_collect.csh script.

An ATE directory is created in the <clone\_dir>. All the TCL files that are included the directory where the collect was run (including its subdirectories) is collected in the ATE directory.

5. Copy if there are any other required files like the reports, logs, and so on, into the <clone-dir>, package the ATE directory <clone\_dir> into a tar file, and send the compressed directory to Synopsys.

The Synopsys support engineer, at the Synopsys site, after obtaining the ATE archive, can recompile the ATE directory.

To recompile the cloned design:

1. Unpacks the ATE archive to a new directory <ate\_directory>.
2. Go to the <ate\_directory>.
3. Sets the
4. SNPS\_VCS\_ATE\_BUILD environment variable to point to the <ate\_directory>.
5. Run the design with VC LP normally using the same VC LP setup file.

### 3.15.1 Limitations

1. There may be issues when the extracted design contains environment variables, Tcl variables in the VC LP setup or UPF.
2. There may be some unnecessary errors reported while running ATE flow (This does not impact the results).
3. Empty directories can be seen in the cloned directory.
4. There may be issues while extracting SDC files which are specified with the `read_sdc` command.
5. The cloned design is not usable with `synopsys_sim.setup`, this file must be removed before running the cloned design.
6. There may be issues in ATE flow when Tcl files are sourced in UPF.

## 3.16 In-design integration of ICC2 and VC LP

You can perform incremental VC LP analysis from within the IC Compiler II environment with a quick turnaround time. The IC Compiler II (ICC2) tool can use the VC LP tool to identify UPF and related violations early in the design cycle. If there are no violations, you can proceed with implementation. If violations exist, you can use ICC2 commands to fix them.



#### Note

This method supports full flat verification in the VC LP tool. To perform top-only verification, you must manually edit the template Tcl file.

### 3.16.1 Use Model

The general procedure is as follows:

1. Create a template Tcl file and specify it as follows:  

```
icc2_shell> setenv ICC2_VCLP_TEMPLATE $template_file
```
2. Open a design block.
3. Configure the VC LP tool environment by using the `set_vclp_options` command. Examine the settings by using the `report_vclp_options` command.
4. Use the `check_vclp_design` command to analyze the block.
5. Examine the VC LP report file.
6. If no violations are found, proceed with implementation.
7. If violations exist, debug them by checking the ICC2 layout view or the VC LP schematic view. You can also execute a VC LP command by using the `run_vclp_cmd` command.

For example, you can open the VC LP user interface with the following command

```
icc2_shell> run_vclp_cmd "gui_start"
```

Use the `vclp_zoom_highlight` command to highlight a selected instance.

8. Fix violations by using supported ECO commands and check the design again.
9. Exit the VC LP environment by using the `vclp_stop` command.

When you execute the `check_vclp_design` command, the ICC2 tool writes the UPF and Verilog files and submits them as input to the VC LP tool. If you are using the golden UPF flow, the VC LP tool must also run in the golden UPF mode. In this case, use the `-golden_upf_files` option with

the `check_vc1p_design` command. The argument for this option is a list of golden UPF file paths (full paths) enclosed in double quotation marks. You can optionally use the `-write_verilog_options` option with the `check_vc1p_design` command to specify any of the options available with the `write_verilog` command. Similarly, you can use the `-save_upf_options` option to specify any of the options available with the `save_upf` command. You must use both of these options together. If specified, these options take precedence over the default behavior. The default behavior of the `-write_verilog_options` option is equivalent to the following command:

```
write_verilog -exclude {leaf_module_declarations corner_cells \ cover_cells  
end_cap_cells filler_cells pad_spacer_cells \ physical_only_cells} -hierarchy  
all
```

The default behavior of the `-save_upf_options` option is equivalent to the following command

```
save_upf -exclude {corner_cells cover_cells end_cap_cells filler_cells  
\pad_spacer_cells physical_only_cells}
```

 **Note**

Currently this feature triggers a full run of VC LP when you execute the `check_vc1p_design` command. This flow will be enhanced to do incremental VC LP analysis in future with very fast turnaround.



# 4 Debugging VC LP Reports

This chapter is organized into the following sections:

- ❖ “[Understanding VC LP Violation Database](#)”
- ❖ “[Smart Grouping of Violations](#)”
- ❖ “[Debugging using Tcl](#)”
- ❖ “[Tcl Query Commands](#)”
- ❖ “[Debugging Using GUI](#)”

## 4.1 Understanding VC LP Violation Database

### 4.1.1 Message IDs and Tags

Each VC LP message has a short name, which consists of three words separated by underscores. The first word represents the electrical family, such as an isolation or level shifter. The second word represents the object which has the problem, such as a strategy, instance or net. The third word represents the problem, such as incorrect connection, missing, redundant. Once you are familiar with the vocabulary of the message names, it should be easy to get an exact understanding of the meaning of the message.

There are eight families: CORR (signal corruption), ISO (isolation gates), LS (level shifters), PG (power and ground connections), PST (power state tables), PSW (power switches) and RET (retention registers).

There are many possible design objects represented by the second word of each tag; examples include INST (a design instance), STATE (a power state in the UPF), or PORT (a port in either or both of the design and UPF).

There are several problem categories represented by the third word of each tag; examples include CONN (signal connected incorrectly), INVERT (signal polarity incorrect), MISSING (required object not found), RANGE (voltage range on level shifter), STATE (a power state where an electrical problem happens), UNUSED (object defined but not used).

With this background, consider the message name ISO\_INST\_MISSING. This means that an isolation gate instance is missing. Therefore, there is a crossing where an isolation gate is required, but it is not found in the design.

#### 4.1.1.1 Message Field Types

For any individual message, a number of fields are printed. Some of these fields are used in the *dynamic help* but a number of additional fields are printed. The information here should make understanding the violation easier.

The following table gives the name and a short description of every field in the VC LP messages.

**Table 4-1 Message Fields with Description**

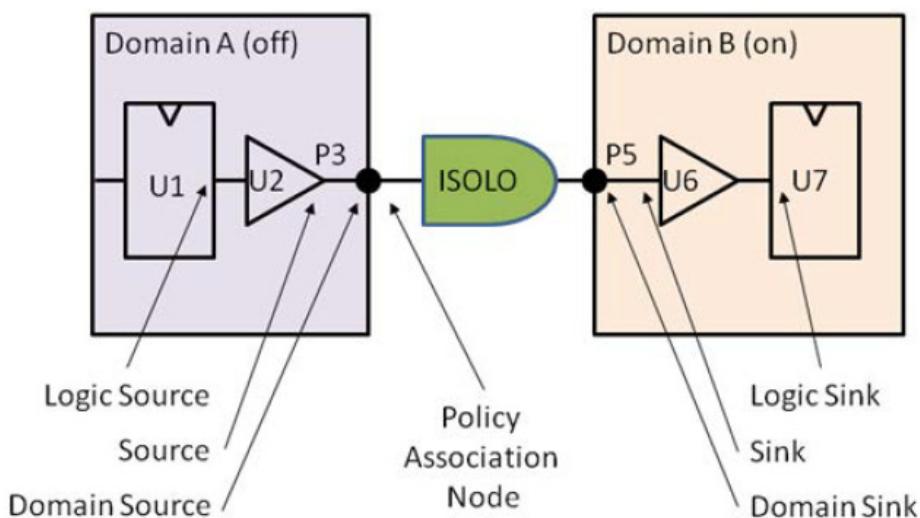
| <b>Field name</b>  | <b>Field description</b>                   |
|--------------------|--------------------------------------------|
| BackupGroundMethod | Backup ground net resolution method        |
| BackupGroundNet    | Backup ground net                          |
| BackupGroundPin    | Backup ground pin                          |
| BackupPowerMethod  | Backup power net resolution method         |
| BackupPowerNet     | Backup power net                           |
| BackupPowerPin     | Backup power pin                           |
| Cell               | Cell name of instance                      |
| CellMaxVoltage     | Cell maximum voltage                       |
| CellMinVoltage     | Cell minimum voltage                       |
| CellPin            | Pin name of cell                           |
| CellRuleValue      | Cell rule value                            |
| DesignClampValue   | Design clamp value                         |
| DesignLocation     | Design location                            |
| DesignNet          | Design net                                 |
| DesignNet          | Design net                                 |
| DesignSupply       | Design supply net                          |
| DestGndNet         | Destination Ground Net                     |
| DestPwrNet         | Destination Power Net                      |
| Domain             | Name of power domain                       |
| DomainSink         | Sink domain crossing pin                   |
| DomainSource       | Source domain crossing pin                 |
| Instance           | Name of instance                           |
| InstanceCount      | Number of instances affected by corruption |
| Instances          | List of instances                          |
| IsolInstList       | List of isolation instances                |
| LogicSink          | Logic sink pin                             |
| LogicSource        | Logic source pin                           |

|                     |                                      |
|---------------------|--------------------------------------|
| MisSenseDesign      | Design signal sense                  |
| MisSenseUpf         | UPF signal sense                     |
| strategies          | List of strategies                   |
| Strategy            | UPF Strategy                         |
| StrategyNode        | Strategy association node            |
| PrimaryGroundMethod | Primary ground net resolution method |
| PrimaryGroundNet    | Primary ground net                   |
| PrimaryGroundPin    | Primary ground pin                   |
| PrimaryPowerMethod  | Primary power net resolution method  |
| PrimaryPowerNet     | Primary power net                    |
| PrimaryPowerPin     | Primary power pin                    |
| Sink                | Sink pin                             |
| SinkGround          | Sink ground net                      |
| SinkGroundMethod    | Sink ground net resolution method    |
| SinkPower           | Sink power net                       |
| SinkPowerMethod     | Sink power net resolution method     |
| Source              | Source pin                           |
| SourceGndNet        | Source Ground Net                    |
| SourceGround        | Source ground net                    |
| SourceGroundMethod  | Source ground net resolution method  |
| SourcePower         | Source power net                     |
| SourcePowerMethod   | Source power net resolution method   |
| SourcePwrNet        | Source Power Net                     |
| SourceSignalList    | List of signal sources               |
| State               | UPF power state                      |
| States              | List of UPF power states             |
| SupplyPort          | UPF supply port                      |
| TieType             | Tie (low,high) type                  |
| UPFBackupPower      | UPF backup power net                 |
| UPFCell             | UPF map cell name                    |

|                  |                               |
|------------------|-------------------------------|
| UPFClampValue    | UPF clamp value               |
| UPFDomainPower   | UPF domain power net          |
| UPFLocation      | UPF location value            |
| UPFMaxVoltage    | UPF maximum voltage           |
| UPFMinVoltage    | UPF minimum voltage           |
| UPFNet           | UPF net                       |
| UPFPrimaryGround | UPF primary ground net        |
| UPFPrimaryPower  | UPF primary power net         |
| UPFRuleValue     | UPF rule value                |
| UPFSupply        | UPF supply net                |
| ViolationSink    | Sink pin of violating net     |
| ViolationSource  | Source pin of violating net   |
| ViolationSources | List of violation source pins |

There are seven key terms which relate to points along a crossover segment. The following diagram shows the relationship between these points.

**Figure 4-1 A Typical Cross-Over**



#### 4.1.1.2 Message Reporting and Iteration

In a mature design, few or zero violations are expected. Before that point, VC LP may find many violations. This section describes an approach for efficiently investigating the violations to find the real design problems.

After running the appropriate check commands as described in the section “[Reading and Building Design](#)”, you should use the `report_violations -app LP` command to display and investigate the violations. By

default, `report_violations -app LP` writes a summary of the violation counts only. Use the `-verbose` option to see the details of all the messages. Sometimes due to incorrect setup, the tool may find a huge number of violations. To protect against accidentally generating a huge result file, by default, only the first 100 messages of each tag are printed. In case you need to see a larger number of messages, you can add the `-limit` flag to the `report_violations -app LP` command. It may also be helpful to add the `-tag` flag to the command line to limit printing to a single tag or list of tags.

[Table 4-2](#) shows an example of the summary printed by `report_violations -app LP`.

**Table 4-2 Summary of Violations**

| Severity | Stage  | Tag                  | Count |
|----------|--------|----------------------|-------|
| Error    | UPF    | ISO_STRATEGY_MISSING | 10    |
| Error    | Design | ISO_INST_MISSING     | 100   |
| Error    | PG     | LS_INGND_CONN        | 50    |
| Error    | PG     | LS_INGND_CONN        | 2000  |
| Warning  | Design | RET_SAVE_CONN        | 1     |
| Warning  | Design | RET_SAVE_CONN        | 10    |

You should review the UPF messages first. Often, a problem with the UPF may also result in a number of design or PG messages. For example, if an isolation strategy is missing from the UPF on a crossing where isolation is needed, then an `ISO_STRATEGY_MISSING` UPF message will appear. For electrical correctness, an `ISO_INST_MISSING` design message will appear for each crossing. Solving the `ISO_STRATEGY_MISSING` message, and then re-synthesizing, will solve the `ISO_INST_MISSING` problem as well.

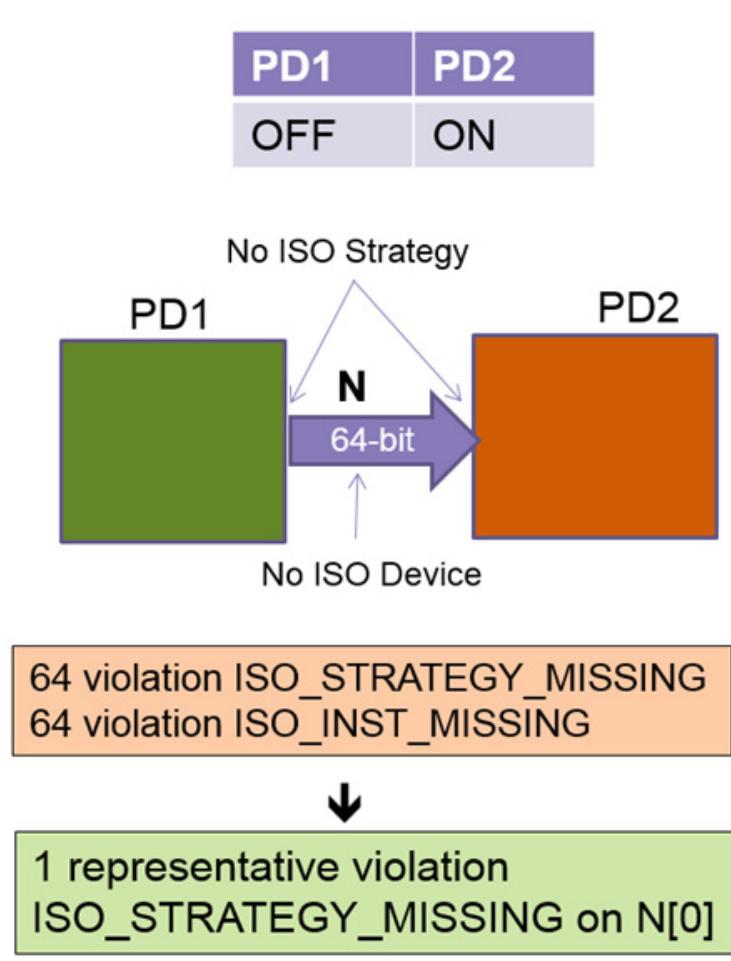
Once the UPF messages are fixed, it may be worthwhile to re-run the tool to update the message counts. For a large design, this may not be practical. The second step is to review the design messages. For a post-route design, the power and ground connectivity messages should be reviewed first.

The next two sections provides more details on the message organization and the `report_violations -app LP` command to investigate the messages efficiently.

#### 4.1.2 Compressing Violations

At times, you come across violations only to find out later that the tool flagged them due to a similar or related causes. In such cases, fixing one issue typically resolves all other corresponding duplicate issues. The compression feature is introduced to help you group messages so that you can focus on the unique problems by analyzing representative violations of grouped messages.

The idea is to define meaningful rules for creating these groups. This capability sounds similar to filters and waivers, but it is not the same. For a filter, you must think up a pattern to be filtered while grouping rules automatically find the right patterns. A better way to think of it, is that grouping rules, each create a number of groups, and each group could have been one cleverly written filter. But all such groups are found automatically. For waivers, the difference is that a group which is waived is assumed to not be an error; however, with grouping, all of the grouped items are real violations.

**Figure 4-2 Violation Compression Example****Figure 4-3 Compression Violation Example Output**

|              |   |                                                                   |
|--------------|---|-------------------------------------------------------------------|
| Tag          | : | ISO_STRATEGY_REDUND                                               |
| Description  | : | Isolation not required, but isolation strategy [Strategy] defined |
| Violation    | : | LP:11                                                             |
| Strategy     | : | LEON3_misc/leon3_misc_iso_out                                     |
| StrategyNode | : | u_m/u_m_pd/ahbs0_0_HRDATA_2                                       |
| Source       | : | u_m/u_m_pd/U9979/z                                                |
| Sink         | : | u_m/u_m_pd/ahbs0_0_HRDATA_2                                       |
| DomainSource | : | u_m/u_m_pd/ahbs0_0_HRDATA_2                                       |
| DomainSink   | : | u_m/u_m_pd/ahbs0_0_HRDATA_2                                       |
| SourceInfo   |   |                                                                   |
| PowerNet     |   |                                                                   |
| NetName      | : | VDDS                                                              |
| NetType      | : | Design/UPF                                                        |
| PowerMethod  | : | FROM_PG_NETLIST                                                   |
| GroundNet    |   |                                                                   |
| NetName      | : | VSS                                                               |
| NetType      | : | Design/UPF                                                        |
| GroundMethod | : | FROM_PG_NETLIST                                                   |
| SinkInfo     |   |                                                                   |
| PowerNet     |   |                                                                   |
| NetName      | : | VDDS                                                              |
| NetType      | : | Design/UPF                                                        |
| PowerMethod  | : | FROM_UPF_POWER_DOMAIN                                             |
| GroundNet    |   |                                                                   |
| NetName      | : | VSS                                                               |
| NetType      | : | Design/UPF                                                        |
| GroundMethod | : | FROM_UPF_POWER_DOMAIN                                             |
| Compression  |   |                                                                   |
| Type         | : | STRATEGY_PORTS                                                    |
| Count        | : | 211                                                               |

There is a "set" of 212 violations and this is the representative one

#### 4.1.2.1 Grouping Types and Rules

The following are the grouping rules, each grouping rule is described in the following section. These grouping rules operate globally on every message type:

##### ❖ FANOUT

It often happens that an instance in one domain has multiple fanouts in another (or multiple) domains. Rather than giving messages for each fanout, the grouping feature enables you to combine messages with the same tag, same LogicSource, and same SinkSupply. If an instance has three fanouts in one domain A and four in another domain B, then the two groups are created. One group contains the domain A violations, with one picked arbitrarily as the representative. The other group contains domain B violations.

##### ❖ BUS\_INST

In many messages, you see the following instances:

u0\_2/abc\_pqr\_STAGE2\_0\_UPF\_LS  
u0\_2/abc\_pqr\_STAGE2\_1\_UPF\_LS

This is a typical pattern of bus bits. A related pattern is two dimensional, such as a\_0\_1\_UPF\_LS. For this pattern, you can group the violations with the same tag and the same instance field. If possible, the message with index 0 should be kept and others should be marked as grouped.

##### ❖ BUS\_PORT

In addition to buses for instances, buses may also be used for ports. The pattern uses square brackets, such as a[0] or a[0][1]. The grouping feature applies this on violations of the same tag, with the pattern applied to the StrategyNode field.

##### ❖ STRATEGY\_INST

Combines two messages with the same source and sink, where one is a UPF level message such as isolation strategy missing and the other is a design level message such as an isolation instance missing. If you fix the strategy message and resynthesize, then the design message is solved.

## ❖ STRATEGY\_PORTS

Combines strategy messages on different ports, where the tag is the same, and the source and sink supplies are the same. This happens generally where a single strategy is missing, and it affects a number of ports with the same domain crossing.

## ❖ STRATEGY\_UNUSED

Combines two messages with the same source and sink, where one is a UPF level message such as isolation strategy unused and the other is a design level message such as isolation instance redundant. If you fix the strategy message and resynthesize, the design message is solved.

## ❖ ISORET\_CONN

Combines messages where a control signal is connected incorrectly, and the intended net and actual net are the same.

## ❖ INTERNAL\_CELL

Internal isolation not needed messages on the same cell type. For a memory, it is common to have internal isolation which may not be used. This would result in one message for each pin of each instance, which can be grouped to show one message for each unique cell type.

## ❖ CROSSOVER\_NOSTATE

Combines UPF\_CROSSOVER\_NOSTATE messages where the UPF supply is the same

## ❖ RET\_NO\_STRATEGY

Combines messages where a retention register is found in a domain, but there is no retention strategy for the domain.

## ❖ MAP\_WRONG

Combines messages where a strategy has a map command, but the design instance does not match, and the cell type is the same.

## ❖ PSW\_CONN

Combines messages where a control or ack signal is connected incorrectly for a power switch cell, and the intended net and actual net are not same.

## ❖ PSW\_WRONG

Combines messages where a power switch instance, has the wrong function of its control signal, and the cell type is the same.

## ❖ ISOLATION

Combines messages where isolation has a constant error and the strategy is the same.

## ❖ CORR\_SOURCE

Combines signal corruption messages where the corrupting instance list and sink are the same.

## ❖ PG\_DOMAIN

Combines messages where one UPF domain's supply net must be connected to a number of instance supply pins, but the same wrong net is found instead.

## ❖ PG\_STRATEGY

Combines messages where one UPF strategy's supply net must be connected to a number of instance supply pins, but the same wrong net is found instead.

## ❖ PG\_CSN

Combines messages where one UPF connect\_supply\_net should be connected to a number of instance supply pins, but the same wrong net is found instead.

## Syntax

```
%vc_static_shell> compress_lp -help
vc_static_shell> compress_lp -help
Usage: compress_lp      # Compresses low power violations covered by a representative
violation
      [-enable <list>]          (enable compression type:
                                    Values: BUS_INST, BUS_PORT, BUS_SINK,
                                    CORR_SOURCE, CROSSOVER_NOSTATE, FANOUT,
                                    INSTANCE, INTERNAL_CELL, ISOLATION,
                                    ISORET_CONN, MAP_WRONG, PG_CSN,
                                    PG_DOMAIN, PG_STRATEGY, PSW_CONN,
                                    PSW_WRONG, RET_NO_STRATEGY,
                                    STRATEGY_INST, STRATEGY_PORTS,
                                    STRATEGY_UNUSED)
      [-disable <list>]          (disable compression type:
                                    Values: BUS_INST, BUS_PORT, BUS_SINK,
                                    CORR_SOURCE, CROSSOVER_NOSTATE, FANOUT,
                                    INSTANCE, INTERNAL_CELL, ISOLATION,
                                    ISORET_CONN, MAP_WRONG, PG_CSN,
                                    PG_DOMAIN, PG_STRATEGY, PSW_CONN,
                                    PSW_WRONG, RET_NO_STRATEGY,
                                    STRATEGY_INST, STRATEGY_PORTS,
                                    STRATEGY_UNUSED)
      [-enable_all]              (enables all compression types)
      [-disable_all]             (disables all compression types)
```

## Use Model

Messages will be compressed when you run compress\_lp. If you wish to enable/disable compression types, use following command which can be either used interactively or through a Tcl script:

```
%vc_static_shell> compress_lp
      [-enable_all]      (enables all compression types)
      [-disable_all]     (disables all compression types)
      [-enable <list>]    (enable compression type)
      [-disable <list>]   (disable compression type)
```

Here, <list> parameters takes the values BUS, FANOUT. For example, the following command disables the FANOUT compression type:

```
%vc_static_shell> compress_lp -disable FANOUT
```

If you want to see messages that are suppressed, you must specify the following additional option to the report\_violations -app LP command to see all messages, including the compressed messages:

```
%vc_static_shell>report_violations -app LP -include_compressed
```

### Example

Consider the following example. When you specify the `report_violations -app LP` command, the tool displays the violations without compression.

```
%vc_static_shell> report_violations -app LP -list
```

The following are the two violations.

```
-----
Management Summary
-----
Stage Family Errors Warnings Infos
-----
Design Isolation 2 0 0
-----
Total 2 0 0
-----
Tree Summary
-----
Severity Stage Tag Count
-----
error Design ISO_CONTROL_CONN 2
-----
Total 2
-----
ISO_CONTROL_CONN (2 errors/0 waived)
-----
1. Strategy top/d_a/s_a isolation control signal iso does not match isolation instance i_0_ connection pin ISO
2. Strategy top/d_a/s_a isolation control signal iso does not match isolation instance i_1_ connection pin ISO
```

Using the same example as above, use compression in this case:

```
%vc_static_shell> compress_lp -enable_all
```

When you specify the `report_violations -app LP` command, the tool displays the following violations after compression:

```
%vc_static_shell> report_violations -app LP -list
```

```
-----
Management Summary
-----
```



| Stage  | Family    | Errors | Warnings | Infos | Compressed |
|--------|-----------|--------|----------|-------|------------|
| Design | Isolation | 1      | 0        | 0     | 1          |
| Total  |           | 1      | 0        | 0     | 1          |

---

#### Tree Summary

---

| Severity | Stage  | Tag              | Count | Compressed |
|----------|--------|------------------|-------|------------|
| error    | Design | ISO_CONTROL_CONN | 1     | 1          |
| Total    |        |                  | 1     | 1          |

---

ISO\_CONTROL\_CONN (1 error/0 waived/1 compressed)

---

Strategy top/d\_a/s\_a isolation control signal iso does not match isolation instance i\_0 connection pin ISO.

Now, consider the message that is compressed and the reason for the compression:

```
%vc_static_shell> report_violations -app LP -verbose -include_compressed
```

The messages may contain the following fields that tell you the technique used for compressing a message. The tool also indicates the representative message number of the group.

Compression

|        |   |      |
|--------|---|------|
| Type   | : | BUS  |
| Master | : | LP:1 |

#### 4.1.2.2 Enhancement in Optimization for the ISO\_INST\_MISSING Tag

Currently, the sub-segment optimization for ISO\_INST\_MISSING tag is too aggressive in certain cases. For a given logic source node, the current caching algorithm takes into consideration the related supplies of the logic sink alone to decide if the violation must be cached, and prevents from being issued again.

The following tags are relaxed by considering the domain source, domain sink, and domain boundary list debug fields using the subsegment\_optimization\_type application variable. The application variables can take values: *default*, *domain*, or *boundary*.

- ❖ ISO\_INST\_MISSING
- ❖ ISO\_INSTMISSING\_NOISO
- ❖ ISO\_STRATEGY\_MISSING
- ❖ LS\_STRATEGY\_MISSING

The subsegment\_optimization\_type application variable can be set to default, domain, or boundary. The subsegment\_optimization\_type application variable works only when the existing enable\_common\_subsegment\_optimization application variable is set to TRUE.

The behavior of the `subsegment_optimization_type` application variable depends on the setting of the existing `enable_domain_boundary_list` application variable.

**Table 4-3 Behavior of subsegment\_optimization\_type and enable\_domain\_boundary\_list**

| <code>subsegment_optimization_type</code> | <code>enable_domain_boundary_list</code> | Behavior                                                                                                                               |
|-------------------------------------------|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| default                                   |                                          | This will be the default value of the app-var and will preserve the current behavior.                                                  |
| domain                                    | TRUE                                     | Cannot proceed as Domain Source and Domain Sink fields will not be available. The type = default is considered with a warning message. |
|                                           | FALSE                                    | Domain Source and Domain Sink fields is considered for the caching and comparison of the violation.                                    |
| boundary                                  | TRUE                                     | Similarity of all elements listed in the Domain Boundary List is considered for caching and comparison of the violation.               |
|                                           | FALSE                                    | Cannot proceed as Domain Boundary List field will not be available. The type = default is considered with a warning message.           |

The following warning message is reported when the application variables set inconsistently:

ID: IIM\_SUBSEG\_OPT\_VARIABLE\_INCONSISTENT

Severity: Warning

Primary: Inconsistent application variable setting.

Secondary: Value "domain" for `subsegment_optimization_type` application variable should be used with `enable_domain_boundary_list` application variable set to false and the value "boundary" should be used with `enable_domain_boundary_list` application variable set to true. Optimization will be performed in "default" value mode due to inconsistency.

## 4.2 Getting the List of Prevented Tags

A tag is prevented from running, if:

- ❖ The tag is disabled by default.
- ❖ You have disabled the tag in the run.
- ❖ Either the tag requires an application variable which is not set, or the tag requires a license which is not available.

Use the `get_prevented_tags` and the `report_prevented_tags` command to get a report of such tags which were prevented from running. These commands is useful to ensure that none of the tags specifically enabled by the user are prevented from running because the tag requires an application variable or a license.

### The `get_prevented_tags` Command

The `get_prevented_tags` returns a Tcl list about the tags that were prevented from running. The Tcl list contains the tag name, prevention type, and name of application variable or license.

## Syntax

```
vc_static_shell> get_prevented_tags
```

## Example

```
get_prevented_tags
{ { tag1 license licensename } { tag2 app_var appvarname } }
```

## The report\_prevented\_tags Command

The report\_prevented\_tags returns a list of the tags that were prevented from running.

## Syntax

```
vc_static_shell> report_prevented_tags
```

## Example

| report_prevented_tags | Tag  | Type    | Name        |
|-----------------------|------|---------|-------------|
|                       | ---  | ---     | ---         |
|                       | tag1 | license | licensename |
|                       | tag2 | app_var | appvarname  |

## 4.3 Smart Grouping of Violations

Design complexities are continuing to grow exponentially, and along with the design complexities, low power complexities are also increasing. More and more sign-off, functional checks are getting added to fulfill the design requirements. Bad user setup or constraints, and dirty designs can lead to the large volume of violations.

Managing large report volumes for sign-off through manual debug is very challenging. Manual approach using Tcl, GUI, reports, design/UPF knowledge is slow and error prone. Manual debug requires tool expertise, domain knowledge, and so on. It is hard to group the similar set of violations, and identify the root cause from the reported VC LP violations.

VC LP helps reducing the debug time by creating and reporting smart groups/clusters and point out the exact root cause so that debug time can be saved.

VC LP considers all the possible cases where smart grouping, root causes can be created and reported to the user in a very clear format.

For example,

- ❖ Constant VSS driven paths can cause ISO/LS\_\*\_REDUND, ISO\_\*\_CONSTANT, PG\_DATA\_SUPPLY, and so on.
- ❖ Missing supply states can cause ISO/LS\_\*\_REDUND, UPF\_CROSSOVER\_NOSTATE, and so on.
- ❖ Incorrect supply connections in PGNetlist can cause ISO/LS\_\*\_REDUND, ISO/LS\_\*\_MISSING, and so on.
- ❖ Incorrect/missing setup related issues can cause unnecessary violations. For example, not setting assign\_port\_from\_padsupply application variable can cause unnecessary ISO/LS\_\*\_MISSING violations on paths from TOP <-> PAD.
- ❖ UPF command based solutions.
- ❖ Paths from TOP port to ANALOG pins can report ANALOG\_NET\_INCORRECT, and solution is to add UPF set\_port\_attributes -is\_analog command to define TOP port as analog port.

- ❖ Unconnected input/output isolation cells and enable of isolation cell tied to constants can be clubbed together as spare cells, and so on.

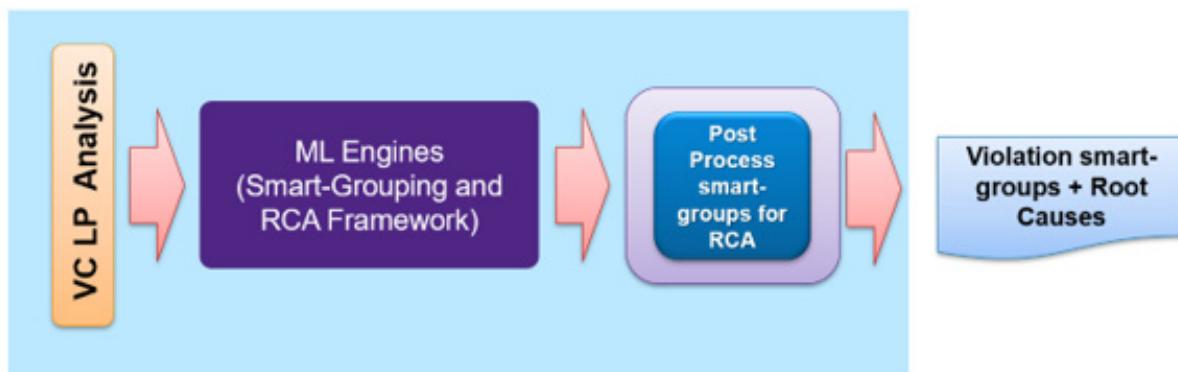
As shown in [Figure 4-4](#), VC LP categorizes existing violations into smart clusters capturing what is CauseViolation and what is the EffectViolation. CauseViolation and EffectViolation are reported in form of smart clusters. These smart clusters are reported to the user for advanced debug.



### Note

The Smart Grouping of Violations feature is enabled in the multi-threaded flow.

**Figure 4-4 Smart Grouping of Violations**

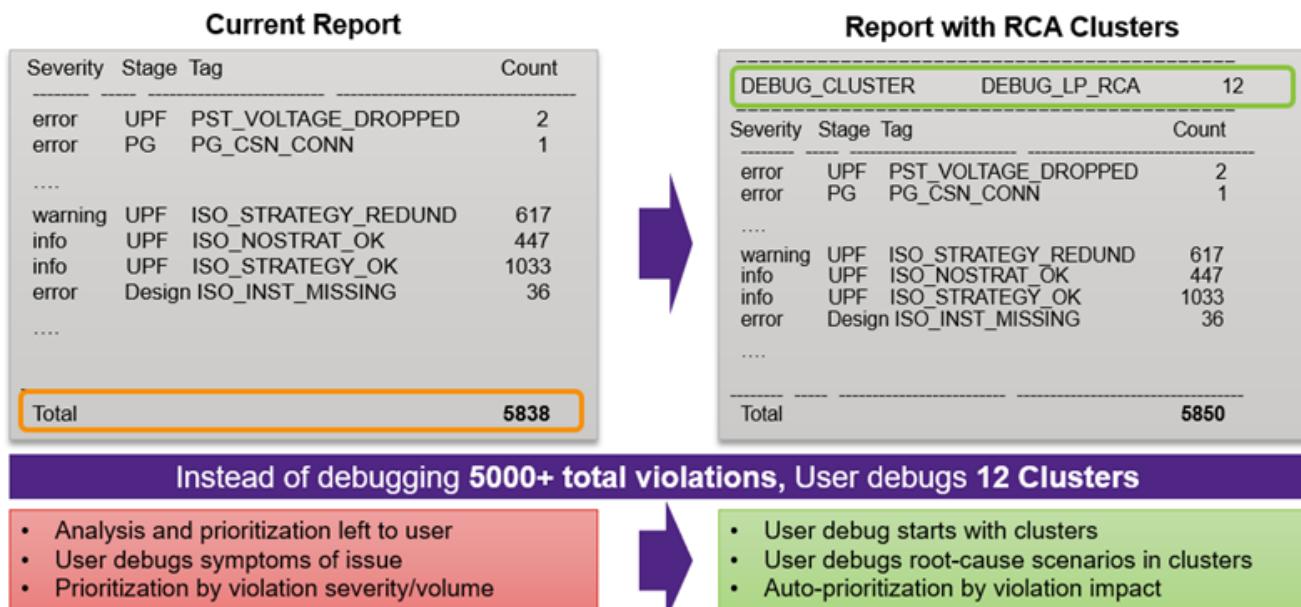


## 4.3.1 Scenarios of Smart Grouping Violations

### 4.3.1.1 PST Merging and PG consistency Issues Effecting Electrical Checks

In the example in [Figure 4-5](#), the following violations are reported:

- ❖ PST\_VOLTAGE\_DROPPED
- ❖ PG\_CSN\_CONN
- ❖ ISO\_STRATEGY\_REDUND
- ❖ ISO\_INST\_MISSING

**Figure 4-5 PST Merging and PG consistency Issues Effecting Electrical Checks**

Due to PST merging issues in the design, the PST\_VOLTAGE\_DROPPED violation is reported and as voltage is dropped, this can cause ISO\_STRATEGY\_REDUND. Here, the PST\_VOLTAGE\_DROPPED is the root cause violation for ISO\_STRATEGY\_REDUND.

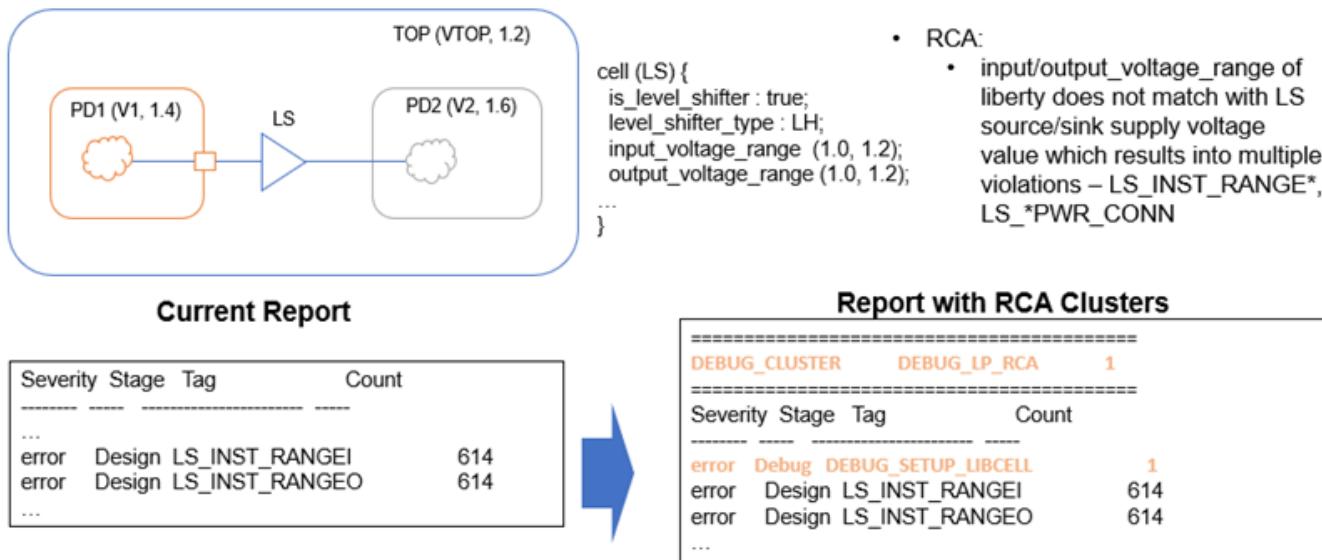
Similarly, if PG and CSN connections are not matching, then PG\_CSN\_CONN violation is reported. In VC LP, the PG connection has highest precedence so ISO\_INST\_MISSING is reported. In this case, the PG\_CSN\_CONN violation is the root cause violation for ISO\_INST\_MISSING.

After enabling this feature, VC LP report smart clusters DEBUG\_LP\_RCA violation capturing CauseViol as PST\_VOLTAGE\_DROPPED/PG\_CSN\_CONN, and EffectViol as ISO\_STRATEGY\_REDUND/ISO\_INST\_MISSING. This helps in quickly identifying the problem.

#### 4.3.1.2 Incorrect Library Setup Related Issues Impacting Electrical Violations

In the example in [Figure 4-5](#), the following violations are flagged:

- ❖ LS\_INST\_RANGEI
- ❖ LS\_INST\_RANGEO

**Figure 4-6 Incorrect Library Setup Related Issues Impacting Electrical Violations**

Here, the root cause is, library cell used for the level shifter cell's input\_voltage\_range/output\_voltage\_range does not match with level shifter source/sink supply voltage values, which can result into LS\_INST\_RANGEI/O violations.

Here, root cause is library cell setup is incorrect. VC LP reports a smart cluster, DEBUG\_LP\_RCA saying, CauseViol as DEBUG\_SETUP\_LIBCELL and EffectViol as LS\_INST\_RANGEI/O violations.

### 4.3.2 Use Model

Set the following two commands in the VC LP Tcl file to enable smart grouping of violations:

- ❖ Set the following command before the `read_file` command:

```
enable_rca_cluster -app lp
```

To enable cluster on a set of tags, use the `-tag_list` option in the `enable_rca_cluster` Tcl command. You can provide a file which contains a list of tags for the `-tag_list` option. The RCA clustering will only happen on those set of tags. This option works only with the `-app lp` option. The other applications will ignore its presence.

Only the effect violation tags are considered from the list specified. The cause violation tags and ML/RCA uncovered tags will be ignored. VC LP report info message on the shell if the cause violation or uncovered tag are mentioned in the file.

#### Example

```
vc_static_shell> enable_rca_cluster -app lp -tag_list tag_list.lst
[Info] Tag name DEBUG_SETUP_LIBCELL specified in tag list is a root cause, only effect violation tags will be considered.
[Info] Tag name DEBUG_SETUP_SUPPLY specified in tag list is a root cause, only effect violation tags will be considered.
[Info] Tag name DEBUG_SUPPLY_LESSON specified in tag list is a root cause, only effect violation tags will be considered.
```

In the tag file (example, tag\_list.lst), each tag should be specified in a new line.  
If the tag file is empty, VC LP will fall back to the original behavior to do RCA on all the tags.

#### Example

```
enable_rca_cluster -app LP -tag_list order.lst
```

```

order.lst (/remo...p_flat) - GVIM12 - □ ×
File Edit Tools Syntax Buffers Window DrChi
File Save Open Print Find Replace Cut Copy Paste Undo Redo Select All
1 UPF_SUPPLY_UNDRIVEN
2 ISO_STRATEGY_MISSING
3 ISO_STRATEGY_MISSING
4 ISO_STRATMISSING_NOBOUNDARY
5 ISO_BUFINV_STATE
6 ISO_INST_MISSING
7 ISO_INST_TRANSPARENT
8 ISO_STRATEGY_REDUND

```

- ❖ Set the following command after check\_\* commands:

```
execute_rootcause_analysis -app lp
```



#### Note

This feature is supported under separate add-on license VC-STATIC-LP-DEBUG.



#### Note

When execute\_rootcause\_analysis is invoked after restoring the session, the following error is reported.

```
vc_static_shell> execute_rootcause_analysis -app {lp} -limit 0
[Error]: Command cannot be executed as root cause analysis is not supported in restored session for -app
lp
```

### 4.3.3 Supported Application Variables

The following application variables are introduced for this support:

- ❖ lp\_rca\_debug\_dump: This variable if set to TRUE, all the RCA related intermediate data will be dumped in the *vcst\_rtdb/internal/adv\_debug/rca\_debug\_dump* directory.
- ❖ lp\_rca\_skip\_debug\_violations: If this variable is set to TRUE, then for a cluster whose root cause list contains both meta high level violations and UPF level violations, all the meta violations will be removed from root cause list.
- ❖ lp\_rca\_display\_xover\_info: When this variable is set to true, the crossover path information for each cluster is added to that cluster information.

### 4.3.4 Violations Introduced for This Support

The following high level violations are reported when the smart grouping of violations is enabled:

- ❖ DEBUG\_CELL\_EXCLUDE
- ❖ DEBUG\_CONSTANT\_SETTINGS
- ❖ DEBUG\_CONTROL\_CONN
- ❖ DEBUG\_CONTROL\_INVERT
- ❖ DEBUG\_CLAMP\_MISSING

- ❖ DEBUG DESIGN FEEDTHROUGH
- ❖ DEBUG EXCEPTIONRAIL\_CONN
- ❖ DEBUG ISO CONST
- ❖ DEBUG ISO DROPPED
- ❖ DEBUG ISO EXCLUDE
- ❖ DEBUG ISO HETERO
- ❖ DEBUG ISO MISSING
- ❖ DEBUG ISO NEEDED
- ❖ DEBUG ISO OPTION
- ❖ DEBUG ISO PAD
- ❖ DEBUG ISO REDUND
- ❖ DEBUG ISO SETUP
- ❖ DEBUG ISO UNUSED
- ❖ DEBUG LS CONST
- ❖ DEBUG LS EXCLUDE
- ❖ DEBUG LS FORCE
- ❖ DEBUG LS HETERO
- ❖ DEBUG LS MISSING
- ❖ DEBUG LS NEEDED
- ❖ DEBUG LS OPTION
- ❖ DEBUG LS PAD
- ❖ DEBUG LS REDUND
- ❖ DEBUG LS SETUP
- ❖ DEBUG LS UNUSED
- ❖ DEBUG MAP WRONG
- ❖ DEBUG PG Conn
- ❖ DEBUG PST STATE
- ❖ DEBUG SETUP ANALOG
- ❖ DEBUG SETUP BBOX
- ❖ DEBUG SETUP CLAMP
- ❖ DEBUG SETUP CONSTANT
- ❖ DEBUG SETUP HANGING
- ❖ DEBUG SETUP INOUT
- ❖ DEBUG SETUP LIBCELL
- ❖ DEBUG SETUP PAD
- ❖ DEBUG SETUP SUPPLY

- ❖ DEBUG\_SPARE\_CELL
- ❖ DEBUG\_SUPPLY\_OFF
- ❖ DEBUG\_SUPPLY\_LESSON
- ❖ DEBUG\_SUPPLY\_ON
- ❖ DEBUG\_SUPPLY\_SHORT
- ❖ DEBUG\_RET\_NOSTRAT
- ❖ DEBUG\_RET\_MISSING
- ❖ DEBUG\_SUPPLY\_UNUSED
- ❖ DEBUG\_UPF\_CSN
- ❖ DEBUG\_UPF\_ATTR
- ❖ DESIGN\_SUPPLY\_RANGE
- ❖ DESIGN\_SUPPLY\_SHORT

#### 4.3.5 Viewing the RCA Results Dashboard

The -dashboard option in the `execute_rootcause_analysis` Tcl command generates an HTML based dashboard to view the RCA results grouped by effect violations (only for app lp).

### VC LP ML RCA Dashboard

| Effect Tag<br>(Covered/Total) | Root Cause               | Effect Tag (Coverage for Root Cause) | Cluster Information | Cluster Proof |
|-------------------------------|--------------------------|--------------------------------------|---------------------|---------------|
| DESIGN_POWER_DIFF (12/12)     | DEBUG_SUPPLY_DIFF (2)    | DESIGN_POWER_DIFF (12/12)            | 1, 2                | 1, 2          |
| ISO_DATA_CONSTANT (3/12)      | DEBUG_SETUP_CONSTANT (3) | ISO_DATA_CONSTANT (3/3)              | 7, 8, 9             | 7, 8, 9       |
| ISO_OUTPUT_STATE (6/6)        | DEBUG_PST_STATE (1)      | ISO_OUTPUT_STATE (6/6)               | 3, 4, 5             | 3, 4, 5       |
|                               | DEBUG_SETUP_CONSTANT (3) | ISO_OUTPUT_STATE (6/6)               | 3, 4, 5             | 3, 4, 5       |
|                               | DEBUG_SUPPLY_LESSON (1)  | ISO_OUTPUT_STATE (6/6)               | 3, 4, 5             | 3, 4, 5       |
| RET_STRATEGY_MISSING (1/1)    | DEBUG_RET_MISSING (1)    | RET_STRATEGY_MISSING (1/1)           | 6                   | 6             |
| UPF_POWER_DIFF (6/6)          | DEBUG_SUPPLY_DIFF (1)    | UPF_POWER_DIFF (6/6)                 | 1                   | 1             |

[Link to all tag violations in this category](#)

[Link to cluster specific text. We will dump individual report for each cluster. Content – cluster, root causes, effect details](#)

#### Example

Tcl:

```
check_lp -stage <Stages>
execute_rootcause_analysis -app lp -dashboard #Performs RCA
```

An effect violation may have multiple root cause violation, by default, all the cluster information for an effect violation will be generated. To limit report generation for each effect violation, use the `-dashboard_limit` option along with the `-dashboard` in the `execute_rootcause_analysis`.

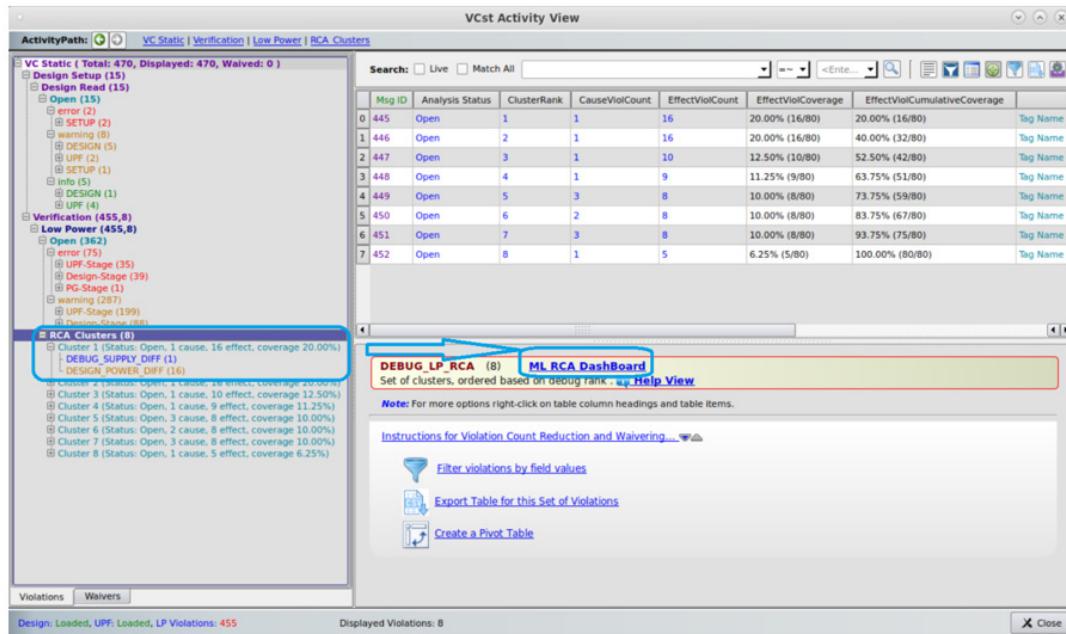
To generate a maximum of five cluster reports per effect violation, use the following command:

❖ execute\_rootcause\_analysis -app lp -dashboard -dashboard\_limit 5

#### 4.3.6 Viewing Smart Grouping of Violations in GUI

You can view the smart grouping of violations in the GUI as shown in [Figure 4-7](#).

**Figure 4-7** **RCA Clusters in GUI**



- ❖ [Figure 4-8](#) shows the summary of clusters. The following figure shows the updated ML RCA dashboard and its functions.

The **ML RCA DashBoard** table contains the bottom-up results of the existing Clusters based on Effect tags. In addition, a similar view based on Cause tags is available in a separate tab.

The DashBoard table links provides appropriate effect, cause and cluster messages in the main Data View to debug further.

**Figure 4-8 Summary of clusters**

Information on effect tag coverage. (covered/uncovered/total)

Effect Tag Centric | Cause Tag Centric

Two tabs; One root cause centric ; one effect centric

Cluster state modifiability

Cross probing facility to the violations in clusters

ML RCA: Cluster Proof for Cluster

Sample Path: [Info] REPORT\_TRACE\_PATHS\_HEADER: path info: Point Type Related\_Power\_Related\_Ground\_Power\_Sample\_Ground\_Inference\_Domain\_Policy\_SPA\_Clamp\_Value

ML RCA Dashboard

ML RCA Dashboard Set of clusters, ordered by Violation\_Count - Default View

Instructions for Violation Count Reduction and Waking... With

Filter violations by Field values

Export Table for this set of Violations

Create a Pivot Table

Cluster Proof in Info view that can show any HTML Help

#### 4.3.7 Tag Coverage Supported

Currently, only crossover and device centric checks are supported. The PSW, RET and other checks are not supported. The following tags are supported.

**Table 4-4 Tag Coverage Supported**

|                   |                     |                         |
|-------------------|---------------------|-------------------------|
| AOB_INPUT_UNCONN  | ISO_STRATEGY_PAD    | PG_STRATEGY_CONN        |
| AOB_OUTPUT_UNCONN | ISO_STRATEGY_REDUND | PG_VOLTMAP_INCONSISTENT |

**Table 4-4 Tag Coverage Supported**

|                       |                             |                       |
|-----------------------|-----------------------------|-----------------------|
| AON_INST_REDUND       | ISO_STRATEGY_UNUSED         | PST_GROUND_NONZERO    |
| ISO_BUFINV_FUNC       | ISO_STRATMISSING_NOBOUNDARY | PST_VOLTAGE_DROPPED   |
| ISO_BUFINV_STATE      | ISO_STRATSUPPLY_INCORRECT   | PST_VOLTAGE_MULTIPLE  |
| ISO_BUFINV_WASTE      | ISO_SUPPLY_MISSING          | PST_VOLTAGE_UNUSED    |
| ISO_CONTROL_CONN      | LS_BUFINV_VOLTAGE           | PSW_ACK_UNCONN        |
| ISO_CONTROL_INVERT    | LS_INPUT_TIEHI              | PSW_CONTROL_TIEHI     |
| ISO_DATA_BLOCKED      | LS_INPUT_TIELO              | PSW_CONTROL_TIELO     |
| ISO_DATA_CONSTANT     | LS_INPUT_UNCONN             | PSW_CONTROL_UNCONN    |
| ISO_DATA_UNCONN       | LS_INPUT_VOLTAGE            | PSW_INPUT_CONN        |
| ISO_ELSINPUT_FUNC     | LS_INPWR_CONN               | PSW_OUTPUT_CONN       |
| ISO_ELSINPUT_STATE    | LS_INST_FORCE               | RET_CLAMP_STATE       |
| ISO_ELSINPUT_WASTE    | LS_INST_INCORRECT           | RET_INPUT_UNCONN      |
| ISO_ELSOUTPUT_FUNC    | LS_INST_MISSING             | RET_OUTPUT_UNCONN     |
| ISO_ELSOUTPUT_STATE   | LS_INST_NOSHIFT             | RET_CLAMP_CONN        |
| ISO_ELSOUTPUT_WASTE   | LS_INST_NOSTRAT             | RET_CLAMP_INVERT      |
| ISO_ENABLE_UNCONN     | LS_INST_PAD                 | RET_CLAMP_FUNC        |
| ISO_INPUT_FUNC        | LS_INST_RANGEI              | RET_CLAMP_MISSING     |
| ISO_INPUT_STATE       | LS_INST_RANGEO              | RET_CLAMP_LEAKAGE     |
| ISO_INPUT_WASTE       | LS_INST_REDUND              | RET_INST_NOSTRAT      |
| ISO_INST_CLAMPED      | LS_INST_THRESH              | UPF_CROSSOVER_NOSTATE |
| ISO_INST_INTERNAL     | LS_INST_WRONG               | UPF_CSN_AON           |
| ISO_INST_MISSING      | LS_INTRACELL_STATE          | UPF_CSN_ISO           |
| ISO_INST_NOISO        | LS_LOCATION_WRONG           | UPF_CSN_LS            |
| ISO_INST_NOSTRAT      | LS_MAP_MISMATCH             | UPF_CSN_MACRO         |
| ISO_INST_PAD          | LS_MAP_WRONG                | UPF_CSN_OTHER         |
| ISO_INST_REDUND       | LS_OUTGND_CONN              | UPF_CSN_PAD           |
| ISO_INST_TRANSPARENT  | LS_OUTPUT_UNCONN            | UPF_CSN_PSW           |
| ISO_INSTMISSING_NOISO | LS_OUTPUT_VOLTAGE           | UPF_CSN_RET           |

**Table 4-4 Tag Coverage Supported**

|                         |                        |                            |
|-------------------------|------------------------|----------------------------|
| ISO_INTERNAL_REDUND     | LS_OUTPWR_CONN         | UPF_CSN_SRM                |
| ISO_LSINPUT_FUNC        | LS_SAMEVOLTXING_GND    | UPF_CSN_ZPR                |
| ISO_LSINPUT_STATE       | LS_SAMEVOLTXING_PWR    | UPF_MACRO_NOSTATE          |
| ISO_LSINPUT_WASTE       | LS_SAMEVOLTXING_PWRGND | UPF_PAD_SUPPLY             |
| ISO_LSOUPUT_FUNC        | LS_SCMR_CONN           | UPF_PORT_UNCONSTRAINED     |
| ISO_LSOUPUT_STATE       | LS_STRATEGY_FORCE      | UPF_PORTSTATE_NOSTATE      |
| ISO_LSOUPUT_WASTE       | LS_STRATEGY_INCORRECT  | UPF_SPADRIVER_STATE        |
| ISO_MAP_MISMATCH        | LS_STRATEGY_MISSING    | UPF_SPARECEIVER_STATE      |
| ISO_MAP_WRONG           | LS_STRATEGY_NOSHIFT    | UPF_SPASUPPLY_CONN         |
| ISO_OUTPUT_FUNC         | LS_STRATEGY_OK         | UPF_SPASUPPLY_MISSING      |
| ISO_OUTPUT_STATE        | LS_STRATEGY_PAD        | UPF_SPASUPPLY_VOLTAGE      |
| ISO_OUTPUT_UNCONN       | LS_STRATEGY_REDUND     | UPF_SRSN_PORT              |
| ISO_OUTPUT_WASTE        | LS_STRATEGY_THRESH     | UPF_STRATEGY_RESOLVE       |
| ISO_SINK_STATE          | LS_STRATEGY_UNUSED     | UPF_SUPPLY_NOSTATE         |
| ISO_STRATCLAMP_MISMATCH | PG_CSN_CONN            | CORR_CONTROL_STATE         |
| ISO_STRATEGY_ANALOG     | PG_DATA_SUPPLY         | CORR_CONTROL_STATE_WITHISO |
| ISO_STRATEGY_INTERNAL   | PG_DIODE_CONN          | CORR_CONTROL_STATE_NOISO   |
| ISO_STRATEGY_MISSING    | PG_DOMAIN_CONN         | UPF_ATTR_DIRECTION         |
| ISO_STRATEGY_NOISO      | PG_MACRO_TIE           |                            |
| ISO_STRATEGY_OK         | PG_PIN_UNCONN          |                            |

#### 4.3.8 Limitations on The Smart Grouping of Violation Feature

- ❖ You cannot run `execute_rootcause_analysis -app lp` command multiple time in the same session/run.

### 4.4 Debugging using Tcl

You can debug the violation reported by VC LP using the Tcl commands described in the sections below.

#### 4.4.1 Reporting Violations with `report_violations -app LP`

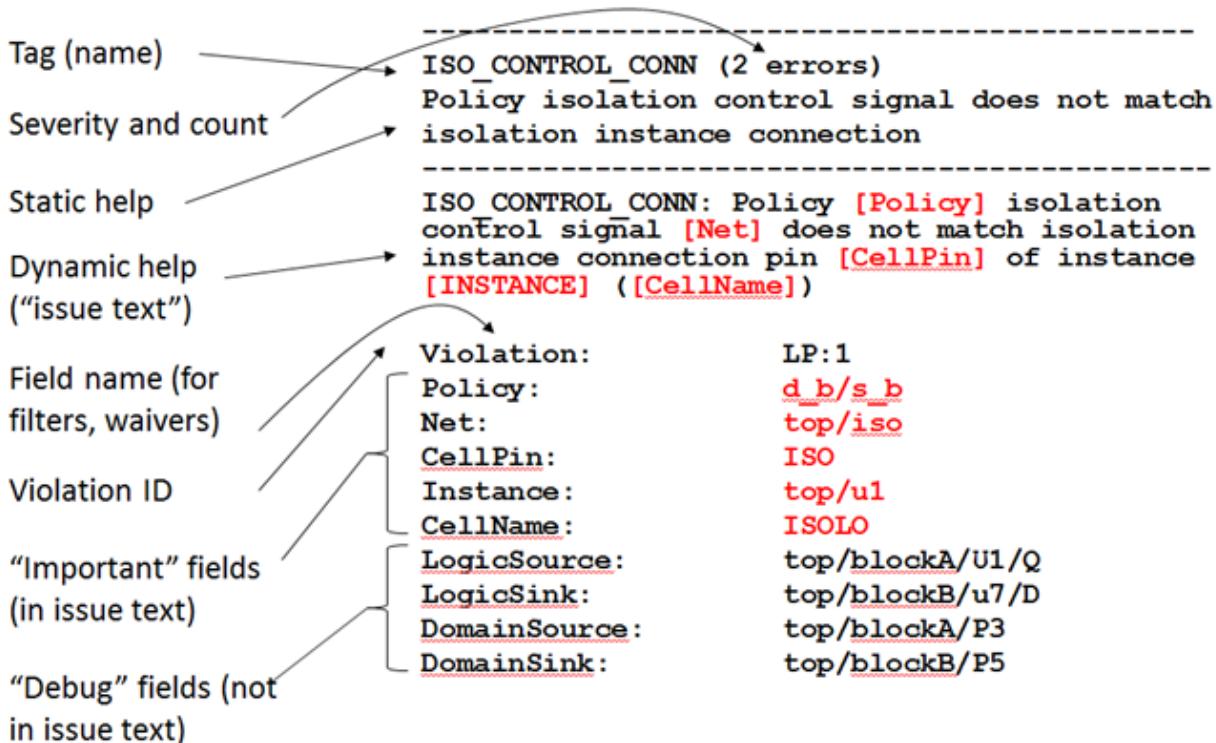
The `report_violations -app LP` command reports the messages generated after low power analysis is done on the design.

The report\_violations -app LP command is the main output command for low power checking. By default, it writes a summary of the messages. The -verbose option is required to write the details of each message. By default, only the first 1000 of each message tag are printed. To print more, use the -limit flag. Here is an example message.

The short name, severity, and count of the message appear on the first line. The second line is a short, static (not dependent on design data) informative description of the problem. Each message of this type appears in a section following; only the first one is shown here. The first line of the message has a design data dependent description. Following this is the violation ID which can be used for reference in this run. The "important" fields which are contained in the dynamic description appear next, one per line, and then the remaining lines are less important fields which may be useful for debugging.

The following figure gives an example output of the ISO\_CONTROL\_CONN violation.

**Figure 4-9 Violation Fields Example**



## Syntax

```
%vc_static_shell> report_violations -app LP -help
Usage: report_violations -app LP # Reports low power check information
      [-no_summary]          (Suppresses summary information)
      [-list]                (List all messages in simple form)
      [-verbose]              (List all messages in detail form)
      [-limit <count>]       (Limit the number of output records per tag)
      [-include_waived]       (Include waived messages in the report)
      [-include_compressed]   (Include compressed messages in the report)
      [-only_waived]          (Report on waived messages)
      [-all_tags]             (Include all tested tags)
      [-tag <tag>]            (Select violations based on tag)
```

```

[-waived <list>]          (Select violations based on waiver name)
[-id <tag>]                (Select violations based on IDs)
[-stage <stage>]           (Select violations based on stage:
                           Values: all, design, pg, upf)
[-severity <list>]          (Select violations based on severity:
                           Values: all, error, info, warning)
[-filter <expression>]      (Select violations based on expression)
[-regexp]                   (Indicates filter expression type to be regular expression
(default glob-style))
[-nocase]                  (Filter expressions ignore case when matching string
values)
[-file <filename>]          (Write the results to the designated file)
[-append]                   (Append results to the designated file)
[-compressions]             (Include all violations in the same compression set(s).)
[-skip_full_path_for_waiver_file]
                           (Displays only base name for waiver file)
[-add_markers]              (Display BEGIN_LP,END_LP markers in verbose mode)
[-display_severity]         (Dump severity info for each record in verbose mode)
[-display_compressed <compression name>]
                           (Dump report corresponding to given compression)
[-id_list]                  (Sets Matching violation ids as command result. Under this
option command will return list of matching violation ids
corresponding to given filter.)

```

## Use Model

Using the various options that the `report_violations -app LP` command provides, you can constrain what you want to see in the report.

Here is an example message.

- ❖ If you just give `report_violations -app LP` in VC LP shell, you will see a summary view of all messages, as explained below:

Management Summary

| Stage  | Family           | Errors | Warnings | Infos |
|--------|------------------|--------|----------|-------|
| UPF    | Isolation        | 3      | 0        | 0     |
| UPF    | LevelShifter     | 5      | 0        | 0     |
| Design | Isolation        | 6      | 0        | 0     |
| Design | LevelShifter     | 5      | 0        | 0     |
| Design | SignalCorruption | 3      | 2        | 0     |
| <hr/>  |                  |        |          |       |
| Total  |                  | 22     | 2        | 0     |

Tree Summary

| Severity | Stage  | Tag                  | Count |
|----------|--------|----------------------|-------|
| error    | UPF    | ISO_CONTROL_STATE    | 1     |
| error    | UPF    | ISO_STRATEGY_MISSING | 2     |
| error    | UPF    | LS_STRATEGY_MISSING  | 5     |
| error    | Design | CORR_CONTROL_STATE   | 3     |
| error    | Design | ISO_CONTROL_INVERT   | 1     |
| error    | Design | ISO_ENABLE_UNCONN    | 1     |
| error    | Design | ISO_INST_NOPOL       | 2     |

|         |        |                  |       |
|---------|--------|------------------|-------|
| error   | Design | ISO_SINK_STATE   | 2     |
| error   | Design | LS_INST_MISSING  | 5     |
| warning | Design | CORR_CONTROL_ISO | 2     |
|         |        | -----            | ----- |
| Total   |        |                  | 24    |

- ❖ Here is an example message for verbose reporting:

```
%vc_static_shell> report_violations -app LP -verbose -file report_lp.log
```

By default, verbose reporting in the `report_violations -app LP` command limits 100 messages per violation tag. If you want to apply a different cap of the number of violations per ID, use the `-limit` option. For example, `report_violations -app LP -limit 0 -verbose -file report_lp.log` gives a verbose report of all message IDs without any limit as follows:

```
-----
ISO_CONTROL_CONN(1 error/0 waived)
-----
Tag : ISO_CONTROL_CONN
      Description : Strategy [Strategy] isolation control signal
[UPFNet] does not match isolation instance [Instance] connection pin [CellPin]
      Strategy : V1/V1_ISO_OUT
      UPFNet
      NetName : v2/v2IsoDwn
      NetType : Design/UPF
      Instance : test_s_UPF_ISO
      Cell : CELL1
      CellPin : B
      DesignNet
      NetName : n734
      NetType : Design
      StrategyNode : v1/test_a
      Source
      PinName : v1/v1/S0
      SegmentSourceDomain : IVC
      Sink : new_mode/A1/A
      SegmentSinkDomain : new
      DomainSource : v1/v1/vc/testa
      DomainSink : v1/v1/vc/testa
      SourceInfo
      PowerNet
      NetName : VDDV10S
      NetType : UPF
      PowerMethod : FROM_UPF_POWER_DOMAIN
      GroundNet
      NetName : VSSC
      NetType : UPF
      GroundMethod : FROM_UPF_POWER_DOMAIN
      SinkInfo
      PowerNet
      NetName : VDDC
      NetType : UPF
      PowerMethod : FROM_UPF_POWER_DOMAIN
      GroundNet
      NetName : VSSC
      NetType : UPF
      GroundMethod : FROM_UPF_POWER_DOMAIN
```

- ❖ Here is an example of the usage of the `-all_tags` option. If a check is run and there were no violations found, the `report_violations -app LP -all_tags` command prints a zero into the summary as follows:

```
-----
Tree Summary
-----
Severity Stage Tag Count
-----
error UPF ISO_STRATEGY_MISSING      0
error UPF ISO_STRATEGY_REDUND     2
error PG PG_DOMAIN_CONN    1
error PG PG_PIN_UNCONN   66
error PG PG_STRATEGY_CONN  2
error PG PG_CSN_CONN        0
-----
Total 72
```

- ❖ Reports can also be generated in a colon based format using the `report_violations -app LP` command as follows:

```
set xsldfile $env(VC_STATIC_HOME)/auxx/monet/schemas/lpColon.xsld
%vc_static_shell>report_violations -app LP -file foo -verbose -xsld $xsldfile
```

#### Example:

```
Tag : LS_INST_MISSING
  Description : Level shifter required on crossing from [Source] to [Sink], but level
shifter missing
  Violation : LP:24
  PstRuleValue : HL_TYPE
  Source:PinName : i_core1/inv0/Z
  SegmentSourceDomain : PD_CORE
  Sink : i_core1/std_buf/A
  SegmentSinkDomain : PD_CORE
  LogicSource:PinName : A
  LogicSink : Y
  DomainSource : A
  DomainSink : Y
  SourceInfo:PowerNet:NetName : VDDtop
  SourceInfo:PowerNet:NetType : Design/UPF
  SourceInfo:PowerMethod : FROM_PG_NETLIST
  SourceInfo:GroundNet:NetName : VSStop
  SourceInfo:GroundNet:NetType : Design/UPF
  SourceInfo:GroundMethod : FROM_PG_NETLIST
  SinkInfo:PowerNet:NetName : VDDsw
  SinkInfo:PowerNet:NetType : Design/UPF
  SinkInfo:PowerMethod : FROM_PG_NETLIST
  SinkInfo:GroundNet:NetName : VSStop
  SinkInfo:GroundNet:NetType : Design/UPF
  SinkInfo:GroundMethod : FROM_UPF_POWER_DOMAIN
```

- ❖ In verbose report output (`report_violations -app LP -verbose`), the severity is not available in the violation fields by default. Specify the `-display_severity` option to get the severity information for each violation.

#### Example

```
report_violations -app LP -tag ISO_STRATEGY_MISSING -verbose -display_severity
-----
```

```
Tag : ISO_STRATEGY_MISSING
```

```
Description : Isolation required on crossing from [Source] to [Sink], but
strategy missing
Severity : error
```

- ❖ Use the add\_markers option to have BEGIN\_LP:<ID> and END\_LP:<ID> markers for easy delineation in verbose mode.

#### Example

```
report_violations -app LP -tag ISO_STRATEGY_MISSING -verbose -add_markers
-----
BEGIN_LP:9
Tag : ISO_STRATEGY_MISSING
Description : Isolation required on crossing from [Source] to [Sink], but
strategy missing
.....
Type : STRATEGY_INST
Count : 1
END_LP:9
```

- ❖ By default, the report\_violations -app LP -verbose output of VC LP did not flag waiver file name. When a waiver file is specified in the manage\_waiver\_file command along with the absolute path as follows:

```
manage_waiver_file -add
/remote/arch_proj/testcases/pv_dbs/malitha/e_libs/nishadi/waive.tcl
```

The output of the report\_violations -app LP command is as follows:

```
report_violations -app LP -verbose [-include_waived]/ [-only_waived]
Tag :ISO_STRATEGY_MISSING
Description : Isolation required on crossing from [Source] to [Sink], but
strategy missing
...
Waiver
Name : ISO_STRATEGY_MISSING_10
Filename : /remote/archproj/
testcases/pv_dbs/malitha/e_libs/nishadi/waive.tcl
```

To get only the name out of the absolute path given for the manage\_waiver\_file command, the -skip\_full\_path\_for\_waiver\_file option is introduced. When this option is specified, the output of the report\_violations -app LP command reports the waiver file name as follows:

```
report_violations -app LP -verbose [-include_waived]/ [-only_waived] -
skip_full_path_for_waiver_file
Tag : ISO_STRATEGY_MISSING
Description : Isolation required on crossing from [Source] to [Sink], but
strategy missing
...
Waiver
Name : ISO_STRATEGY_MISSING_10
Filename : waive.tcl
```

The -skip\_full\_path\_for\_waiver\_file option can be used along with waive\_lp command, such that the absolute path is dropped, and just the bare waive file name is reported.

```
waive_lp - skip_full_path_for_waiver_file
Waiver : ISO_STRATEGY_MISSING_10
Filename : waive.tcl
Violations : 1
```

```
Select
Tag : ISO_STRATEGY_MISSING
```

- ❖ Consider the following Tcl file:

```
define_compression -tag ISO_STRATEGY_UNUSED -field DomainSource -name ISO
compress_violations -enable ISO
```

```
report_lp -display_compressed ISO
```

The following output is reported when `-display_compressed` option is used:

---

```
Management Summary
```

---

```
Stage Family Errors Warnings Infos Compressed
```

---

```
Design Isolation 0 3 0 3
```

---

```
Total 0 3 0 3
```

---



---

```
Tree Summary
```

---

```
Severity Stage Tag Count Compressed
```

---

```
warning Design ISO_STRATEGY_UNUSED 3 3
```

---

```
Total 3 3
```

The `-display_compressed` option in the `report_violations` command supports wild card characters.

```
report_lp -display_compressed IS*
```

#### 4.4.2 Viewing User-defined Object Names in Violation Reports

By default, VC LP violation reports contain the internally generated names. Many of these internally generated names have limited usage. When you set the `enable_nearest_user_defined_obj` application variable to true, VC LP reports the nearest user-defined crossover node in the respective violations' logic source and logic sink in the crossover based violation reports.



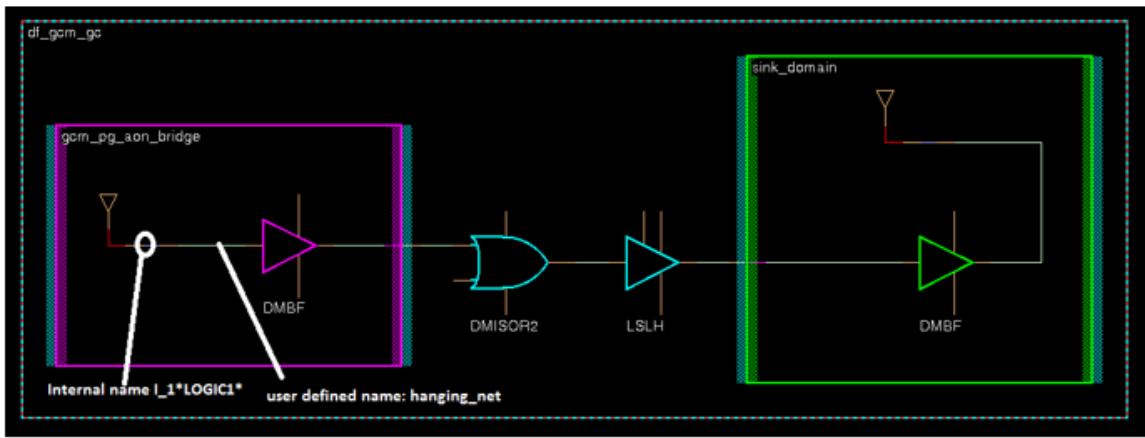
##### Note

Other segments in the violation report (especially in the `report_violations -app` LP file) may contain internal names.

##### Example

Consider the design as illustrated in [Figure 4-10](#).

**Figure 4-10 The PG\_DATA\_TIED Violation**



Currently, the PG\_DATA\_TIED violation reported for the example in [Figure 4-10](#) is as follows:

```

PG_DATA_TIED (1 warning/0 waived)
-
Tag : PG_DATA_TIED
Description : Data pin [CellPin] instance [Instance] ([Cell]) tied to literal constant [TieType] in PG design
Violation : LP:30
CellPin : A
Instance : gcm_pg_aon_bridge/U35
Cell : DMBF
TieType : TIE_HIGH
Source
  PinName : gcm_pg_aon_bridge/I_1_*Logic1*
  ConstType : Design
  ConstValue : 1
Sink : gcm_pg_aon_bridge/U35/A

```

**Source reported has an tool generated name**

When the `enable_nearest_user_defined_obj` application variable is set to true, the violation is reported as follows:

```

PG_DATA_TIED (1 warning/0 waived)
-
Tag : PG_DATA_TIED
Description : Data pin [CellPin] instance [Instance] ([Cell]) tied to literal constant [TieType] in PG design
Violation : LP:30
CellPin : A
Instance : gcm_pg_aon_bridge/U35
Cell : DMBF
TieType : TIE_HIGH
Source
  PinName : gcm_pg_aon_bridge/hanging_net
  ConstType : Design
  ConstValue : 1
Sink : gcm_pg_aon_bridge/U35/A

```

**A name defined in the design**

This means,

- ❖ Tags which are not crossover based are not supported under this application variable. Example: DESIGN\_FORK, user-defined tags, CORR\_CONTROL\_STATE
- ❖ Tags which are validated before crossover path reduction are not supported under this enhancement. Example: ISO\_STRATEGY\_IGNORED, LS\_STRATEGY\_IGNORED

- ❖ Tags can report internal (tool generated) names under segment sink. Example: LS\_INST\_REDUND, LS\_INST\_RANGE, LS\_INST\_MISSING, LS\_STRATEGY\_MISSING, ISO\_STRATEGY\_REDUND, LS\_STRATEGY\_REDUND
- ❖ Tags can have driver nodes with internal names. Example: ISO\_SINK\_STATE, ISO\_BUFINV\_STATE, ISO\_ELSINPUT\_STATE
- ❖ Tags can have load nodes with internal names. Example: UPF\_CROSSOVER\_NOSTATE
- ❖ Tag can report internal names for segment source. Example: LS\_NOINST\_OK

### Note

In the future releases, the `enable_reporting_rtl_names_lp` will be replaced by `enable_nearest_user_defined_obj`. Currently, you must set the `enable_reporting_rtl_names_lp` application variable to *false*, when the `enable_nearest_user_defined_obj` application variable is set to *true*.

For more details on the `enable_reporting_rtl_names_lp` application variable, see the man page of `enable_reporting_rtl_names_lp`.

#### 4.4.2.1 Exception for UPF\_CROSSOVER\_NOSTATE

In the UPF\_CROSSOVER\_NOSTATE violation report, for a given crossover, when the load node is the same as the logic source and the logic sink, the reported name of the logic source and sink is also an internal name. This scenario occurs very rarely.

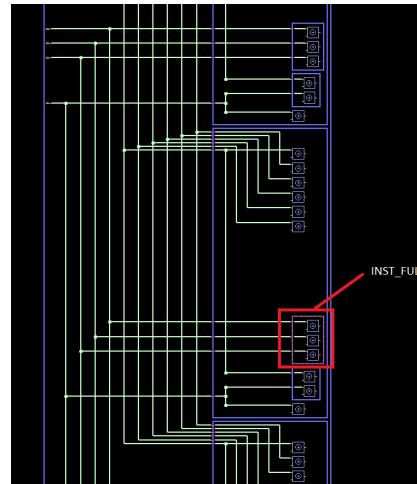
**Figure 4-11 RTL of the Reported Module**

```

15 module [buff full] (in1_6, in2_7, in3_8, out1_6, out2_7, out3_8);
16
17 input [63:0] in1_6;
18 input [63:0] in2_7;
19 input [63:0] in3_8;
20
21 output [63:0] out1_6;
22 output [63:0] out2_7;
23 output [63:0] out3_8;
24
25 assign out1_6 = in1_6 + 1;
26 assign out2_7 = in2_7 + 1;
27 assign out3_8 = in3_8 + 1;
28
29 endmodule

```

**Figure 4-12 Schematic**



**Figure 4-13 The view\_activity GUI**

| Violations: UPF_CROSSOVER_NOSTATE |        |                                           |                                           |           |                                           |
|-----------------------------------|--------|-------------------------------------------|-------------------------------------------|-----------|-------------------------------------------|
|                                   | Msg ID | LogicSource                               | LogicSink                                 | UPFSupply | LoadNode                                  |
| 0                                 | 7      | LOGIC_TOP2/I_ADD_out_2/OUT[45]            | LOGIC_TOP2/I_ADD_out_2/OUT[45]            | PD2_VDD   | LOGIC_TOP2/I_ADD_out_2/OUT[45]            |
| 1                                 | 8      | LOGIC_TOP2/I_ADD_out_2/OUT[45]            | LOGIC_TOP2/I_ADD_out_2/OUT[45]            | VSS       | LOGIC_TOP2/I_ADD_out_2/OUT[45]            |
| 2                                 | 9      | LOGIC_TOP2/I_ADD_out_3/OUT[59]            | LOGIC_TOP2/I_ADD_out_3/OUT[59]            | PD2_VDD   | LOGIC_TOP2/I_ADD_out_3/OUT[59]            |
| 3                                 | 10     | LOGIC_TOP2/I_ADD_out_3/OUT[59]            | LOGIC_TOP2/I_ADD_out_3/OUT[59]            | VSS       | LOGIC_TOP2/I_ADD_out_3/OUT[59]            |
| 4                                 | 11     | LOGIC_TOP2/INST_FULL/I_ADD_out2_7/OUT[44] | LOGIC_TOP2/INST_FULL/I_ADD_out2_7/OUT[44] | PD2_VDD   | LOGIC_TOP2/INST_FULL/I_ADD_out2_7/OUT[44] |
| 5                                 | 12     | LOGIC_TOP2/INST_FULL/I_ADD_out2_7/OUT[44] | LOGIC_TOP2/INST_FULL/I_ADD_out2_7/OUT[44] | VSS       | LOGIC_TOP2/INST_FULL/I_ADD_out2_7/OUT[44] |
| 6                                 | 13     | LOGIC_TOP3/I_ADD_out_10_#31_0/OUT[18]     | LOGIC_TOP3/I_ADD_out_10_#31_0/OUT[18]     | PD3_VDD   | LOGIC_TOP3/I_ADD_out_10_#31_0/OUT[18]     |
| 7                                 | 14     | LOGIC_TOP3/I_ADD_out_10_#31_0/OUT[18]     | LOGIC_TOP3/I_ADD_out_10_#31_0/OUT[18]     | VSS       | LOGIC_TOP3/I_ADD_out_10_#31_0/OUT[18]     |
| 8                                 | 15     | LOGIC_TOP3/I_ADD_out_9_#31_0/OUT[28]      | LOGIC_TOP3/I_ADD_out_9_#31_0/OUT[28]      | PD3_VDD   | LOGIC_TOP3/I_ADD_out_9_#31_0/OUT[28]      |
| 9                                 | 16     | LOGIC_TOP3/I_ADD_out_9_#31_0/OUT[28]      | LOGIC_TOP3/I_ADD_out_9_#31_0/OUT[28]      | VSS       | LOGIC_TOP3/I_ADD_out_9_#31_0/OUT[28]      |
| 10                                | 17     | LOGIC_TOP2/I_ADD_out_1/OUT[23]            | LOGIC_TOP2/I_ADD_out_1/OUT[23]            | PD2_VDD   | LOGIC_TOP2/I_ADD_out_1/OUT[23]            |
| 11                                | 18     | LOGIC_TOP2/I_ADD_out_1/OUT[23]            | LOGIC_TOP2/I_ADD_out_1/OUT[23]            | VSS       | LOGIC_TOP2/I_ADD_out_1/OUT[23]            |
| 12                                | 19     | LOGIC_TOP3/INST_PART/I_ADD_out1_9/OUT[30] | LOGIC_TOP3/INST_PART/I_ADD_out1_9/OUT[30] | PD3_VDD   | LOGIC_TOP3/INST_PART/I_ADD_out1_9/OUT[30] |

#### 4.4.3 Filtering Messages

You can filter messages based on the following using the -filter option.

- ❖ Family
- ❖ Severity
- ❖ Filter based on debug fields
- ❖ Wildcards and expressions

##### Example 1

```
%vc_static_shell> report_violations -app LP -tag ISO_CONTROL_CONN
%vc_static_shell> report_violations -app LP -family isolation
%vc_static_shell> report_violations -app LP -severity ERROR
```

##### Example 2

**Usage of report\_violations -app LP with -filter**

Consider an example where you:

- ❖ Need a report of all messages related to isolation strategy redundant (ISO\_STRATEGY\_REDUND)
- ❖ Need a filter for a DomainSource containing the string ahbs0\_0\_\_HRDATA\_17

The following command extracts the required report:

```
%vc_static_shell> report_violations -app LP -tag ISO_STRATEGY_REDUND -filter
{ (DomainSource=="top/*ahbs0_0__HRDATA_17*") }
```

### **Proper Construction of Filter/Waivers for List Fields**

In `report_violations -app LP`, a very small number of violations contain a field which is a list. Care must be taken when filtering or waiving list fields. Due to a bug, some filter/waivers may have worked before when they should not have. This section describes the proper construction of filter/waivers for list fields. Because this can be a little tricky, VC Static intentionally avoids using lists unless there is no other practical way to represent the violation.

Here is an example of a list in a `report_violations -app LP` violation:

```
Tag : LS_MAP_UNAVAIL
Description : ... MappedCells Mapped
Cell : MLSX3_RVT Mapped
Cell : LSX4_RVT
```

This violation contains a list of two cells. There is no guarantee that the cells will appear in the same order in every violation, on every platform, in every software release. So, a properly written filter should not rely on the order of the list.

When a filter is applied to a list field, the string presented to match against is a single string containing all of the items. The filter pattern you provide is not matched against each cell, but against the entire string. For this example, the string presented is " MLSX3\_RVT LSX4\_RVT ". In particular, notice the leading and trailing space, and the space between the items.

Any filter you write must take into account these spaces and must be independent of the order of the items. Unfortunately, previous software versions had a bug where there were no spaces at all, which led to some incorrectly written filters matching in some cases. For example, the following incorrect filter may have worked sometimes, but not all times:

```
%vc_static_shell> report_violations -app LP -filter MappedCells:MappedCell=~MLSX3*
```

This matched properly, but only as long as the MLSX3\_RVT cell happened to appear first in the list. If it had appeared second in the list, then this filter would not have matched. The proper way to write this is:

```
%vc_static_shell> report_violations -app LP -filter MappedCells:MappedCell=~" * MLSX3*"
* "
```

The spaces are important to avoid matches on substrings. The leading and trailing stars are important to allow matching anywhere in the list.

#### **4.4.4 Operations on Tag Definitions**

Previously, in the VC Static platform, violation tags/violations and the sequence of debug fields present in these violations are fixed and cannot be changed/reordered. In some cases, debugging becomes difficult. For example, in a tag/violation message, all the design elements are displayed first, then all the power information. So, correlating the design element with its power information is not convenient. To overcome this issue, you can change the violation tags/violations and the sequence of debug fields present in these violations.

VC Static provides the following Tcl commands to operate tags/violations. For example, renaming of tag/violation, reorder/disable debug fields of a tag/violation.

- ❖ **get\_tags:** Returns tag names. If a single tag name is given as an argument, then it prints the same tag name. If used with a wild-card character it expands the wild-card character to a list of matching tags. It lists the tag/tags independently of the violations/tags present in the current run of the report database. It does not accept a list of tags, but wild-card characters are accepted.

For example:

```
%vc_static_shell> get_tags ISO_*_MISSING
ISO_INST_MISSING ISO_STRATEGY_MISSING
```

- ❖ **get\_tag\_info:** Accepts a single tag at a time and lists the information about the tag. It does not accept a list of tags or wild-card characters. It runs independent of the violations/tags present in current run of report database.

For example:

```
%vc_static_shell> get_tag_info ISO_INST_REDUND
LP warning Design Isolation enabled 0
```

- ❖ **get\_tag\_fields:** Lists all the fields for a particular tag. It does not accept wild-card characters and list of tags. It lists the tag fields independent of the violations/tags present in current run of the report database.

For example:

```
%vc_static_shell> get_tag_fields LS_STRATEGY_MISSING
{Msg ID} Tag PstRuleValue Source SegmentSourceDomain Sink SegmentSinkDomain
LogicSource LogicSink LogicSinkUnconnected DomainSource DomainSink
DomainBoundaryList SourceInfo SinkInfo
```

- ❖ **getViolation\_tags:** Returns ordered list of tags for the violations present in the report database. This command requires no arguments.

For example:

```
%vc_static_shell> getViolation_tags
LS_INST_MISSING LS_OUTPWR_CONN PG_BIAS_INSUFFICIENT PG_CSN_CONN PG_DOMAIN_SETUP
```

- ❖ **rename\_tag:** Takes a single tag and replaces its name with the new alias. This command should be used BEFORE the check\* commands. It does not accept a list of tags or wild-card characters.

For example:

```
%vc_static_shell> rename_tag ISO_INST_NOSTRAT NO_POLICY_ISO_FOR_DEVICE
```

- ❖ **disable\_tag\_field:** Disables tag fields and prevents them from getting printed in the report. This command should be used only AFTER check\* commands. It does not accept a list of tags or list of fields or wild-card characters.

For example:

```
%vc_static_shell> disable_tag_field ISO_CONTROL_CONN SourceInfo
```

- ❖ **reorder\_tag\_field:** In order to generate a report where Info fields appear just after the design element fields, reorder\_tag\_fields command can be used to change the order of the fields reported. This command does not accept wild-card characters and list of tags. It lists the tag fields independent of the violations/tags present in the current run of the report database.

For example:

```
%vc_static_shell> set order [get_tag_fields ISO_INST_MISSING]
```

```
%vc_static_shell> set new_order [lsort $order]
%vc_static_shell> reorder_tag_fields ISO_INST_MISSING $new_order
%vc_static_shell> report_violations -app LP -verbose
```

## 4.4.5 Custom Checks

You can create custom checks in a Tcl script apart from the ones provided by VC LP, and add them as part of the standard VC LP report. These custom violations appear in `report_violations -app LP` report, and you can filter and waive these violations like any other standard violations. This section defines commands that you can use to define custom violations.

### 4.4.5.1 The `add_lpViolation` Command

This command adds a new violation instance of the given tag to the reporting database.

#### Syntax

```
%vc_static_shell> add_lpViolation -help
Usage: add_lpViolation      # adds a violation to the LP database
       [-tag <name>]          (Tag name)
       [-fields <name=value name=value>]
                           (Field name/value pairs)
       [-nosave]              (Skip saving added violation to disk)
       [-save]                (Save all added violation(s) to disk)
       [-nocover]             (Disable checking of whether displayable fields are
covered by given field options)
```

#### Options

- ❖ `-tag`

The TagName must be built-in, or, already defined using `define_lpViolation` prior to using `add_lpViolation` command. The TagName input is treated as case insensitive. The report is shown in all capital letters.

- ❖ `-fields {FieldName1FieldNameN}`

The field name can be one of the following at present:

- ◆ domain
- ◆ domainsink
- ◆ domainsource
- ◆ logicsink
- ◆ logicsource
- ◆ sink
- ◆ source
- ◆ strategy
- ◆ upfssupply
- ◆ instance
- ◆ cellpin

The values have to be valid values from the design. For example, `Instance=top/mid_inst_1/bot_inst_2` is a valid input when the design actually has the

hierarchy top->mid\_inst\_1->bot\_inst\_2. If some invalid input is given, an error is shown. The Vector type values are not allowed. You should only use scalar type, or bit-select of vector type.

The field name is treated case insensitive. Any alphabetical value is always case sensitive and should be as in design. Multiple FIELD = VALUE pairs can be given, separated by a single space, within braces or quotes. A field is predefined to be capable of populating values of one or more fields (displayable fields) that are to be displayed as defined for the built-in/user defined tag. You will therefore be able to populate values of multiple displayable fields without actually providing values to all of them. The values provided to option-fields work as hooks to design which populates one/more displayable fields. [Table 4-5](#) lists what option-field(s) can populate which display-field(s).

**Table 4-5 Option-Fields and Display-Fields**

| Option-field | Displayable Fields populated by<br>option-field      |
|--------------|------------------------------------------------------|
| domain       | Domain                                               |
| domainsink   | DomainSink                                           |
| domainsource | DomainSource                                         |
| logicsink    | LogicSink                                            |
| logicsource  | LogicSource                                          |
| sink         | Sink, SinkInfo                                       |
| source       | Source, SourceInfo                                   |
| strategy     | Strategy                                             |
| upfsupply    | UpfSupply                                            |
| instance     | Instance, CellType                                   |
| cellpin      | CellPin, Instance, PinPgType,<br>DesignNet, CellType |

- ◆ To add record fields in the add\_lp\_violations command.

```
set fields
{UPFNet:NetName=ABC1 UPFNet:NetName=ABC2}
add_lpViolation -tag ISO_CONTROL_CONN -fields $fields -nocover
ISO_CONTROL_CONN (1 error/0 waived)
-----
Tag : ISO_CONTROL_CONN
Description : Strategy [Strategy] isolation control signal [UPFNet] does not
match isolation instance [Instance] connection pin [CellPin]
Violation : LP:1
UPFNet
NetName : ABC1
NetName : ABC2
```

- ◆ To add list fields

```
set fields
{Strategies:StrategyInfo:StrategyNode=VAL1
Strategies:StrategyInfo:UPFCclampValue=VAL2}
```

```

add_lpViolation -tag RET_CONTROL_BOTH -fields $fields -nocover
-----
RET_CONTROL_BOTH (1 warning/0 waived)
-----
Tag : RET_CONTROL_BOTH
Description : Retention control signals [UPFNet] are the same for multiple
retention strategies while their polarities are different
Violation : LP:2
Strategies
StrategyInfo
UPFClampValue : VAL2
StrategyNode : VAL1

```

❖ **-nosave**

Skips saving the added violation instance to the database immediately. This achieves faster operation.

 **Note**

If this switch is used for one or more consecutive `add_lpViolation` commands, you MUST use the `add_lpViolation -save` command before any other command that interacts with the reports (for example, `reportViolations -app LP`, `waive_lp`, `define_lpViolation`, `remove_lpViolation` and so on). By default, without this switch, the violation instance is saved to the database.

❖ **-save**

Enforces database save for all previously inserted violation instances. It is redundant if used with a successful `add_lpViolation -tag` command; because even without this switch, database save takes place to save all previously inserted non-committed data. The `add_lpViolation -save` command enforces database save for all previously issued insert operations. The recommended steps for faster `add_lpViolation` operation are as follows:

- Use one or more `add_lpViolation -tag ... -field ... ... -nosave` [you can use a loop of `remove_lpViolation` commands in your Tcl script]
- After step a is performed, use `add_lpViolation -save` command [outside the loop].

❖ **-nocover**

This is an optional switch. By default, the tool checks whether Option-Fields can populate values for all Displayable Fields defined for the built-in/user defined tag. If not, an ERROR is issued. Under `-nocover`, this check is skipped.

## Example

```

add_lpViolation -tag MY_VIOLATION -fields {DomainSink=CORE2/in1
DomainSource=CORE1/out1 LogicSink=CORE2/and_i1/A LogicSource=CORE1/ff_i1/Q Domain=TOP
Sink=CORE2/buf_i1/A Source=CORE1/buf_i1/Z Instance=CORE1/ff_i1 CellPin=CORE1/ff_i1/Q
UPFSupply=PD2.primary.power Strategy=PD1/isol PowerStateTable=top_pst State=top_pst/s1
VoltageValue={1.0 1.2}}

```

|              |                                            |
|--------------|--------------------------------------------|
| Tag          | : MY_VIOLATION                             |
| Description  | : This is my [Source] and [Sink] violation |
| Violation    | : LP:3                                     |
| DomainSource | : CORE1/out1                               |
| DomainSink   | : CORE2/in1                                |
| LogicSource  |                                            |

```

PinName      : CORE1/ff_i1/Q
LogicSink    : CORE2/and_i1/A
Domain       : TOP
Source
  PinName   : CORE1/buf_i1/Z
  Sink      : CORE2/buf_i1/A
  Instance   : CORE1/ff_i1
  CellPin   : Q
  Cell      : DMFD2
  CellType  : StandardCell
SourceInfo
  PowerNet
    NetName  : VDD1
    NetType   : UPF
  PowerMethod : FROM_UPF_POWER_DOMAIN
  GroundNet
    NetName  : VSS
    NetType   : UPF
  GroundMethod : FROM_UPF_POWER_DOMAIN
SinkInfo
  PowerNet
    NetName  : PD2.primary.power
    NetType   : UPF
  PowerMethod : FROM_UPF_POWER_DOMAIN
  GroundNet
    NetName  : PD2.primary.ground
    NetType   : UPF
  GroundMethod : FROM_UPF_POWER_DOMAIN
DesignNet
  NetName   : CORE1/w1
  NetType   : Design
UPFSupply   : PD2.primary.power
Strategy     : PD1/isol
PowerStateTable : top_pst
State        : top_pst/s1
VoltageValue : 1.0 1.2

```

#### 4.4.5.2 The define\_lpViolation Command

Use the `define_lpViolation` command to define a new violation tag.

##### Syntax

```

vc_static_shell> define_lpViolation -help
Usage: define_lpViolation      # adds definition of a violation tag to the LP database
schema
      [-tag <name>]          (Tag name)
      [-stage <stage>]         (Stage:
                                Values: design, pg, upf)
      [-family <family>]       (Family:
                                Values: analog, aob, bias,
                                compatibility, designconsistency,
                                designfeedthrough, designfork, diode,
                                isolation, levelshifter, powerground,
                                powerstatetable, powerswitch, retention,

```

```

                signalcorruption, upfconsistency)
[-severity <error|warning|info>]
    (Severity level:
     Values: error, info, warning)
[-description <description>]
    (Textual description of the violation)
[-dynamic_help <dynamic help>]
    (Help message with dynamic field value)
[-fields <name name>] (Field names)
[-nosave]             (Skip saving added violation definition to disk)
[-save]               (Save all added violation definition(s) to disk)

```

## Options

❖ -tag <name>

If a built-in tag name is given, an error message is shown. Built-in tags cannot be redefined. If a user-defined tag name is given, then:

- ◆ If no instances of the same exist, redefinition occurs and an INFO message is shown informing you that redefinition has taken place.

OR

- ◆ An error message is shown prompting you to remove violations first. The given TagName is treated case insensitive. The report shown in all capital letters.

❖ -stage <stage>

The stage name can be either of DESIGN | PG | UPF. The StageName is treated case insensitive.

❖ -family <family>

The family name can be one of the following:

- ◆ DESIGNFEEDTHROUGH,
- ◆ DESIGNFORK
- ◆ ISOLATION
- ◆ LEVELSHIFTER
- ◆ POWERGROUND
- ◆ POWERSTATETABLE
- ◆ POWERSWITCH
- ◆ RETENTION
- ◆ SIGNALCORRUPTION
- ◆ UPFCONSISTENCY

The value is case insensitive.

❖ -severity <error | warning | info>

SeverityValue can be either of ERROR | WARNING | INFO type. The given SeverityValue is treated case insensitive.

❖ -description

Description can be any user specified string.

❖ **-dynamic\_help <dynamic help>**

The dynamic help option can be any user specified string with actual dynamic fields within a pair of colons ':'. For example, "This is a valid dynamic help text involving : Instance: and : CELLPIN: as dynamic fields". Do not use ':' in the string other than its special meaning as delimiters. In such cases an error is shown. Dynamic fields appearing in colons must also be provided in the list of fields given using -fields switch, otherwise an error is shown.

❖ **-fields <name name>**

These are pre-defined field names as they appear in built-in violation reports as shown by the report\_violations -app LP command. For the records, you have to provide the top-level record name; individual record elements are not allowed as fields. Multiple field names should be given space separated, within braces or quotes. All the predefined fields that you see for a built-in violation types cannot be user-defined at present. If a field cannot be user defined, or, the given field does not exist, an error message is shown. The list of fields which can be user-defined are:

- ◆ Domain
- ◆ DomainSink
- ◆ DomainSource
- ◆ LogicSink
- ◆ LogicSource
- ◆ Sink
- ◆ SinkInfo
- ◆ Source
- ◆ SourceInfo
- ◆ Strategy
- ◆ UpfSupply
- ◆ Instance
- ◆ CellType
- ◆ CellPin
- ◆ PinPgType
- ◆ DesignNet
- ◆ PowerStateTable
- ◆ State
- ◆ VoltageValue

The field name/s are treated case insensitive. The report is shown in CamelCase.

❖ **-nosave**

Skips saving the added violation definition to the database immediately. This results in quicker operation.

**Note**

If this switch is used for one or more consecutive `define_lpViolation` command, you MUST use the `define_lpViolation -save` command before any other command that interacts with reports (for example `reportViolations -app LP / waive_lp / add_lpViolation / remove_lpViolation` etc.) is used. By default, without this switch, the violation definition is saved in the database.

- ❖ `-save`

Enforces database save for all previously inserted violation definitions. It is redundant if used with a successful `define_lpViolation -tag` command; because even without this switch, the database save takes place to save all previously inserted non-saved data. The command `define_lpViolation -save` enforces database save for all previously issued insert operations. The recommended steps for faster `define_lpViolation` operation are:

- a. Use one or more `define_lpViolation -tag ... -field ... ... -nosave`.
- b. After Step 1 is done, use the `define_lpViolation -save` command.

**Example**

```
define_lpViolation -tag MY_VIOLATION -stage Design -family Isolation -severity Warning
-description Description -dynamic_help "This is my :source: and :sink: violation" -
-fields {DomainSink DomainSource LogicSink LogicSource Domain Sink Source Instance
CellPin Cell CellType SourceInfo SinkInfo DesignNet UPFSupply Strategy PinPGType
PowerStateTable State VoltageValue}
```

**4.4.5.3 The `remove_lpViolation` Command**

Use the `remove_lpViolation` command to remove the violation instances from the database. However, the violation definition are not removed. After you remove the violations, the violation appears as an INFO message and the count of the violation instances is reduced.

**Note**

This command can be used only after UPF reading is successful.

**Syntax**

```
%vc_static_shell> remove_lpViolation -help
Usage: remove_lpViolation      # removes violations from the LP database
       [-tag <name>]          (Remove violations based on Tag name)
       [-id <id>]              (Remove violations based on IDs)
       [-nosave]                (Skip updating removed violation in disk)
       [-save]                  (Update all removed violation(s) in disk)
```

**Options**

- ❖ `-tag <name>`

The tag name can be a built-in tag, or a user-defined tag previously defined using the `define_lpViolation` command. The tag name is treated case insensitive. Multiple tag names can be given, separated by a space. Supports the use of wildcards for a tag name.

- ❖ `-id <id>`

Can be any existing violation's ID. Multiple IDs can be given, separated by space.

- ❖ `-nosave`

Skips saving the change for a removed violation instance to the database immediately. This achieves faster operation.



### Note

If this switch is used for one or more consecutive `remove_lpViolation` command, you MUST use the `remove_lpViolation -save` command before any other command that interacts with reports (for example `reportViolations -app LP / waive_lp / define_lpViolation / add_lpViolation etc.`) is used. By default, without this switch, the removal of violation instance is saved to database.

#### ❖ -save

Enforces database save for all previously removed violation instances. It is redundant if used with a successful `remove_lpViolation -tag ...` command; because even without this switch, database save takes place to save all previously removed non-saved data. The `remove_lpViolation -save` command enforces database save for all previously issued remove operations. The recommended steps for faster `remove_lpViolation` operation are:

- Use one or more `remove_lpViolation -id ... -nosave` [you can use a loop of `remove_lpViolation` commands in your Tcl script]
- After Step 1 is complete, use the `remove_lpViolation -save` command [outside the loop].

### Example

```
remove_lpViolation -tag MY_VIOLATION
```

## 4.4.6 Waiving Messages

### 4.4.6.1 Waiving Messages Using the `waive_lp` Command

While reviewing violations using the `reportViolations -app LP` command, you may not want to see those messages again and want to filter them out. To accomplish this, you require a waiver mechanism which can be applied while reviewing the violations report. Hence, as part of sign off process cumulative waivers will be reviewed.

To meet this requirement, VC LP provides a robust waiver mechanism using the `waive_lp` command.

#### Syntax

```
%vc_static_shell> waive_lp -help
Usage: waive_lp      # Manages waivers for low power check violations
      [-add <name>]          (Add waiver)
      [-append name]         (Append additional filter parameters to an existing
waiver)
      [-comment <comment>]    (Waiver comment)
      [-delete <name(s)>]     (Delete waiver)
      [-delete_all]           (Delete all waivers)
      [-tcl]                  (Display the waiver list in Tcl command format)
      [-force]                (Create a container for waive_lp append operations)
      [-tag <tag>]             (Waive violations based on tag)
      [-id <tag>]              (Waive violations based on IDs)
      [-stage <stage>]         (Waive violations based on stage:
                                Values: all, design, pg, upf)
      [-severity <list>]        (Waive violations based on severity:
                                Values: all, error, info, warning)
      [-filter <expression>]   (Waive violations based on expression)
```

```

[-regexp]           (Indicates filter expression type to be regular expression
(default glob-style)
[-nocase]          (Filter expressions ignore case when matching string
values)

```

### **-add mode**

```
%vc_static_shell> waive_lp -add <name> [-comment <string>] [-force] [-tag <tags>] [-severity <severities>] [-id <ids>] [-stage <stages>] [-family <families>] [-filter <expression>] [-regexp] [-nocase]
```

This command mode adds the named waiver to a waiver table and marks those violations selected by the filter options. All options after -comment are exactly as they appear in the report\_violations -app LP command. These filter options may be simply cut-and-pasted from the report\_violations -app LP command. It is important to note that the set of waived violations may be larger than the set of report\_violations -app LP displayed violations. This is because report\_violations -app LP does not (by default) display those violations already waived while the waive\_lp command applies the newly created waiver to all violations filtered by these command line options.

### **-append mode**

```
%vc_static_shell> waive_lp -append <name> [-force] [-tag <tags>] [-severity <severities>] [-id <ids>] [-stage <stages>] [-family <families>] [-filter <expression>] [-regexp] [-nocase]
```

This command mode adds additional filter options to the named (existing) waiver. This mode does more than keeping you under the waiver limit. It allows you to group whole collections of violations not easily filtered by a single set of filter criteria under the same named waiver. For instance, you may wish to have a single waiver named waived, the set of violations best described by a collection of filter statements. Without this mode you will have to create artificial categorizations (waived1, waived2, etc) simply to describe what is logically a single category.

### **-delete\_all mode**

```
%vc_static_shell> waive_lp -delete_all
```

Removes all waivers from the waiver table and the low power application repository.

### **-delete mode**

```
%vc_static_shell> waive_lp -delete <name>
```

Removes the named waiver from the waiver table and resets the previously selected waivers from the low power application repository.

### **List mode**

```
%vc_static_shell> waive_lp [-tcl]
```

The waive\_lp command with no options, displays all currently assigned waivers in a tabular format. With the -tcl option waivers are displayed with a syntax fully compatible with the waive\_lp -add mode. This allows for the capture and replay of the current set of waivers in subsequent design sessions.

## **4.4.6.2 Improving Runtime Performance for the waive\_lp Command**

You can improve the performance of the waive\_lp command using the enable\_new\_filter\_logic application variable.

```
%vc_static_shell> set_app_var enable_new_filter_logic true
```

You can arbitrarily enable or disable this application variable during the same run, unless you do not see any difference in results; that is, the same output should come from the `report_violations -app LP -filter` command, and the same number of violations should be waived by the `waive_lp -filter` command. The only difference you should see is the change in the runtime.

#### 4.4.6.3 Migrating Waivers from Block level to Top Level

When a top design contains different blocks created by different designers, the top level designer cannot directly reuse the waivers from each block directly at the Top level. To resolve this issue, VC LP has introduced the `migrate_waivers` command to enable you to migrate waivers from the block level to the Top level.

The `migrate_waivers` command does not check for the syntax when migrating block level waivers to the Top Level. Therefore, you must ensure that all the waivers in the block level waiver file are written correctly without any errors.

#### Syntax

```
vc_static_shell> migrate_waiver -help
Usage: migrate_waivers      # Migrate block waivers to top level
       [-module <module_name>]
                           (Name of block module to process)
       [-instance <instance_name>]
                           (Name of block instance to process)
       [-waiver_name_suffix <pattern>] (Add suffix for migrated waiver names)
       [-referenceListFile <reference_list_file_name>]
                           (Specify the file containing mapping for modules)
       [-referenceList <reference_list_members>]
                           (Specify the list of space seperated mapping for modules)
       [-discard_supply]      (Instead of migrating supply fields, remove them)
       [-append]              (Append migrated waivers into the given top_level_file)
       [-apply]               (Apply the migrated waivers into tool for waiver
application)
       <input_filename>       (Input block waiver filename)
       [<output_filename>]     (Output top waiver filename)
```

#### Use Models

```
%vc_static_shell>migrate_waivers block_waive.tcl top_waive.tcl -instance BLOCK1
%vc_static_shell>migrate_waivers block_waive.tcl top_waive.tcl-module block
```

#### Block level waiver

```
waiveViolation -app lp -add LS_LOCATION_WRONG tag LS_LOCATION_WRONG
migrate_waivers -module abc abc_waiver.tcl abc_top_waiver.tcl
waiveViolation -app lp -add LS_LOCATION_WRONG_abc_0 -tag LS_LOCATION_WRONG
migrate_waivers -module abc abc_waiver.tcl abc_top_waiver.tcl -waiver_name_suffix SYN
waiveViolation -app lp -add LS_LOCATION_WRONG_abc_0_SYN -tag LS_LOCATION_WRONG
```

#### 4.4.6.3.1 Tips for Creating Waivers at the Block Level

Try to follow these tips when you create waivers at the block level because not all block level waivers can be migrated to top level without errors/warnings. You may later need to verify or update the migrated waivers that give warning/error messages.

- ❖ Avoid using general waivers like -severity all/warning/info at the block level as they may affect the other blocks when migrated to the Top level.

- ❖ Avoid using block waivers with the `-delete` or `-delete_all` option in the `waive_lp` command as they may unintentionally affect the other blocks when migrated to the top level.
- ❖ Do not use waivers like `-LP:id` at the block level because it changes on every run and signature because it may vary from tool to tool.

#### 4.4.6.3.2 Examples of Migrated Waivers to the Top Level

There are different types of `waive_lp` command at the block level. This section describes the different kinds of blocks for which `waive_lp` command is applied, and how these block level waivers are migrated to the Top level. For some of the scenarios, you may get a warning message indicating that a manual check is required.

##### Example 1: Block Waivers with Filters on Instance Objects

If the `waive_lp` command contains filters like Instance, PivotInstance, Domain, UPFScope, Strategy at the block level, then when you migrate the block level waivers to Top level, the waivers added at the Top level are applied to make the filter unique among instances.

Consider an example where a block ma contains the following waiver:

```
%vc_static_shell>waive_lp -add w1 -filter instance==x/u1
```

When you migrate the waiver from block level to Top level, and if the Top design contains two instances of block ma: y/b1 and z/b2, then the following waivers are generated at the Top level.

```
%vc_static_shell>waive_lp -add w1_ma_0 -filter instance==y/b1/x/u1
%vc_static_shell>waive_lp -add w1_ma_1 -filter instance==z/b2/x/u1
```

The Suffixes `_ma_0`, `_ma_1` are added to make the filter unique between instances when the waivers are migrated from different blocks.

##### Example 2: Block Waivers with Filters on Cell Objects

If the `waive_lp` command contains filters like filter cell or filter any tag, then when you migrate these block level waivers to Top level, the waivers added at the Top level are such that the filter options are attached to the required block, and not any other block. Most of the violations have either an instance or a StrategyNode field to bind the scope, therefore the instance or a StrategyNode is added in the Top level waiver after migration.

Consider an example where block ma contains following waiver:

```
%vc_static_shell>waive_lp -add w1 -filter cell==MY_DFF
```

When you migrate the waivers to the Top level, the waiver waives all violations containing the `cell==MY_DFF` for all the instances of the block ma at the Top level.

```
%vc_static_shell>waive_lp -add w1_ma_0 -filter { (cell==MY_DIFF) && (Instance=~y/b1/*)
|| (StrategyNode=~y/b1/*) }
```

If the tags do not contain the instance or a StrategyNode field, then when you migrate such waivers to the Top level, you get the following warning message:

[Warning] WAIVER\_MIGRATION\_ANCHOR: Waiver tag has no anchor field Waiving by tag alone is not possible for this tag since it has no instance specific fields.

The waivers cannot be prevented from operating on all blocks of the design.

##### Example 3: Block Waivers with Filters on Port Objects

Consider the following waiver applied at the block level (the block in the figure above):

```
%vc_static_shell>waive_lp -add w1 -filter LogicSink=~P*
```

The port P2 is the logicsink at the block level. However, at the Top level the LogicSink can change as per the design. Therefore, when you migrate the waiver to the Top level, the waiver applied at the Top level is as follows:

```
%vc_static_shell>waive_lp -add w1_ma_0 -filter LogicSink=~y/b1/I1/P*
```

If the block level waiver waives the tags using any of the primary ports, the following message is reported:

[Warning] WAIVER\_MIGRATION\_PRIMARY: Primary port reference in waiver w1: LogicSink=~P\*

The waiver terms of the block's primary inputs may no longer operate as expected. The waiver is migrated but it should be manually verified.

#### Example 4: Block Waivers with Filters on Supply Objects

If the `waive_lp` command contains the `-filter` option which is a supply object at the block level, then when you migrate the waivers to the Top level, the supply name for the block level may change at the Top level. Therefore, you get a warning message and you should manually check if the waivers are migrated correctly. If there are not migrated correctly, the you must change the supply name in the Top level waiver for proper execution.

Consider the following waiver written at the block level:

```
%vc_static_shell>waive_lp -add w1 -filter {SinkInfo:PowerNet:NetName == top_vdd}
```

If you migrate this waiver to the Top level, and the following waiver is applied at the Top level:

```
%vc_static_shell>waive_lp -add w1_black_box_0 -filter {SinkInfo:PowerNet:NetName == "i_bbox/top_vdd"}
```

You get the following warning message:

[Warning] WAIVER\_MIGRATION\_SUPPLY: Supply reference in waiver w1:  
SinkInfo:PowerNet:NetName==top\_vdd

*The block level supplies are connected to the top level supplies which might change the supply name for that waiver term.*

#### 4.4.6.3.3 Error and Warning Messages for the `migrate_waivers` Command

The following are the error and warning messages that you might see while using the `migrate_waivers` command.

- ❖ [Warning] WAIVER\_MIGRATION\_PRIMARY: Primary port reference in waiver waiver1:  
LogicSink=~out\*

Waiver terms on block primary inputs may no longer operate as expected. The term is migrated but should be manually verified.

- ❖ [Warning] WAIVER\_MIGRATION\_SUPPLY: Supply reference in waiver waiver2:  
UPFSupply==PD1.primary.power

Waiver terms on supplies will not operate as expected. The term is migrated but should be changed for proper execution.

- ❖ [Error] LP\_INVALID\_WAIVER\_NAME: Waiver name (waiver1\_BLO@CK1\_0) contains invalid characters.

Waiver names are constrained to the following character set: [a-zA-Z0-9\_][a-zA-Z0-9\_\$-]\*.

- ❖ [Warning] WAIVER\_MIGRATION\_SOURCE: Sourced waiver file not migrated: foo.tcl

Waiver migration operates on a single file. Files which are sourced by the first file are not migrated. You must separately migrate the sourced file.

- ❖ [Warning] WAIVER\_MIGRATION\_DELETE: Waiver deletion not migrated Instead of migrating a script which creates and later deletes a waiver, please simplify the file by removing the self-canceling command pair
- ❖ [Warning] WAIVER\_MIGRATION\_STAGE: Waiving by stage alone not supported for migration  
Waiving by stage alone is not supported for waiver migration. The waiver cannot be prevented from operating on all blocks of the design.
- ❖ [Warning] WAIVER\_MIGRATION\_ID: Waiving by ID not supported for migration  
Waiving by ID is not recommended in a normal flow and will give the wrong result during waiver migration. Please waive by some other condition
- ❖ [Warning] WAIVER\_MIGRATION\_ANCHOR: Waiver tag has no anchor field Waiving by tag alone is not possible for this tag since it has no instance specific fields.  
The waiver cannot be prevented from operating on all blocks of the design.

#### 4.4.6.3.4 Limitations

- ❖ The `migrate_waivers` command only updates the `waive_lp` command of block level tcl file. It puts the other lines as it is in the top level .tcl file. Even the source other than the `waive.tcl` inside `block.tcl` is also just copied. If those waivers are also required to be migrated, then you must migrate them separately.
- ❖ You can not use all escape characters in an instance name because the `migrate_waivers` command has constraints for the instance. The waiver names are constrained to the following character set: [a-zA-Z0-9\_][a-zA-Z0-9\_\$-\*].
- ❖ The `-module` switch is not supported in the `migrate_waivers` command for VHDL design.

#### 4.4.6.4 Identifying Incorrect Waivers on Certain Violations

Consider a scenario where an incorrect UPF is detected by corruption during dynamic simulation in VCS-NLP, and violations such as ISO\_STRATEGY\_MISSING are reported in VC LP. The design team declares this scenario as safe and therefore, you write a waiver in VC LP to remove this violation. However, the diagnosis on the failed chip confirms the violation was a real one, which should not have been waived.

VC LP provides the `analyze_waiver_correctness` command to enable you to identify such waivers that should not have been waived. Using this command, you can generate a simulation assertion for each unique power state rejected due to a waiver on a violation. If one of these assertions triggers during RTL simulation, this indicates the waiver(s) must be more carefully reviewed.

The `analyze_waiver_correctness` command can be used only for the violations that describe a combination of power states which exists in the design and that can cause an electrical problem. For example, ISO\_STRATEGY\_MISSING is reported when a source supply is off and a sink supply is on, which causes an electrical problem. In VC LP, some waivers are created to waive-off one of these power state violations, because you think that the combination of these power states cannot actually occur in NLP. The `analyze_waiver_correctness` command can be used to verify the correctness of such waivers.

There are other violations which do not describe any power state combinations. For example, PG\_PIN\_UNCONN shows a structural problem, where a PG pin in a netlist is left floating. You cannot use the `analyze_waiver_correctness` command for identifying incorrect waivers on such violations.

The steps to be used for this feature are as follows.

1. Find all the waived violations which describe combinations of power states.
2. Extract the unique set of power state combinations which are covered by the waived violations.

3. Write a file containing simulation assertions. For each unique power state combination, write one assertion, and a comment describing which waivers and violations caused this assertion to be generated.
4. (Optional) You may want to review the comments in the file, and decide to edit the file to remove certain assertions if they were created for other reasons.
5. Run RTL simulations with the assertions.
6. If any assertion triggers, use the comments in the file to determine which waiver(s) need to be carefully reviewed.

#### 4.4.6.4.1 The analyze\_waiver\_correctness Command

The `analyze_waiver_correctness` command enables you to identify the incorrect waivers on certain violations in VC LP. This command performs extensive analysis before writing output.

The command looks at all the waivers and violations in the current run, which you should set up before executing the command.

The command takes a filename as its input, where the output must be written.

The command has the `-verbose` option which causes a much larger output file to be written. By default, a short comment is written for each assertion with a single violation ID and waiver name. If you use the `-verbose` option, the entire list of violation ID's and waivers are written. This can be large, however, it is helpful for debugging.

#### Example

Suppose you have the following violations and a waiver applied on the violation. (Only the relevant fields of the violation are shown.)

```

ISO_STRATEGY_MISSING
  ViolationID      LP:72
  StrategyNode    b1/p1
  SourceInfo
    PowerNet
      NetName     vdda
  SinkInfo
    PowerNet
      NetName     vdd
  Waiver          w1
ISO_STRATEGY_MISSING
  ViolationID      LP:73
  StrategyNode    b1/p2
  SourceInfo
    PowerNet
      NetName     vdda
  SinkInfo
    PowerNet
      NetName     vdd
  Waiver          w1
vcst_shell> waive_lp -add w1 -tag ISO_STRATEGY_MISSING
           -filter {sourceinfo:powernet:netname==vdda}

```

Then you use the following command:

```
%vc_static_shell> analyze_waiver_correctness foo.sva
```

VC static generates the following file as output.

```

import UPF::*;
module generated_assertions;
    // Variable declarations for supplies
    UPF::supply_net_type local_vdd;
    UPF::supply_net_type local_vdda;
    initial begin
        $upf_mirror_state ("vdd", local_vdd);
        $upf_mirror_state ("vdda", local_vdda);
    end
    // Assertions
    // Violations: 1, first is 1; Waivers: 1, first is w1
    awc_1 : assert #0 (!((local_vdd.voltage != 0) &&
                           (local_vdda.voltage == 0)));
endmodule

module bind_file;
    bind top generated_assertions generated_assertion_inst (*.);
endmodule

```

The `upf_mirror_state` command is a (new) proprietary command in VCS. This is required to link, or mirror, a UPF supply to a simulation variable which can be used in an assertion.

#### 4.4.6.4.2 Supported Violation Tags

The following is the list of tags supported along with the details of how the assertion for the particular tag is generated. For example, each waived violation of `ISO_INST_MISSING` generates an assertion for the state where the source supply is off, and the sink supply is on.

- ❖ `ISO_INST_MISSING {{Source OFF} {Sink ON}}`
- ❖ `ISO_SINK_STATE {{Isolation OFF} {Sink ON}}`
- ❖ `ISO_STRATEGY_MISSING {{Source OFF} {Sink ON}}`
- ❖ `ISO_STRATEGY_NOISO {{Source OFF} {Sink ON}}`
- ❖ `ISO_STRATSUPPLY_INCORRECT {{Source OFF} {Sink ON}}`
- ❖ `ISO_BUFINV_FUNC {{Source ON} {Instance OFF} {Sink ON}}`
- ❖ `ISO_BUFINV_STATE {{Driver OFF} {Instance ON}}`
- ❖ `ISO_BUFINV_STATE {{Instance OFF} {Sink ON}}`
- ❖ `ISO_INPUT_FUNC {{Source ON} {Isolation OFF} {Sink ON}}`
- ❖ `ISO_LSINPUT_FUNC {{Source ON} {Level_Shifter OFF} {Sink ON}}`
- ❖ `ISO_ELSINPUT_FUNC {{Source ON} {ELS OFF} {Sink ON}}`
- ❖ `ISO_OUTPUT_FUNC {{Source ON} {Isolation OFF} {Sink ON}}`
- ❖ `ISO_LSOUPUT_FUNC {{Source ON} {Level_Shifter OFF} {Sink ON}}`
- ❖ `ISO_ELSOUTPUT_FUNC {{Source ON} {ELS OFF} {Sink ON}}`
- ❖ `ISO_INPUT_STATE {{Driver OFF} {Isolation ON}}`
- ❖ `ISO_LSINPUT_STATE {{Driver OFF} {Level_Shifer ON}}`
- ❖ `ISO_ELSINPUT_STATE {{Driver OFF} {ELS ON}}`

- ❖ *ISO\_OUTPUT\_STATE {{Driver OFF} {Isolation ON}}*
- ❖ *ISO\_LSOOUTPUT\_STATE {{Driver OFF} {Level\_Shifter ON}}*
- ❖ *ISO\_ELSOUTPUT\_STATE {{Driver OFF} {ELS ON}}*
- ❖ *ISO\_INPUT\_STATE {{Isolation OFF} {Sink ON}}*
- ❖ *ISO\_LSIINPUT\_STATE {{Level\_Shifter OFF} {Sink ON}}*
- ❖ *ISO\_ELSINPUT\_STATE {{ELS OFF} {Sink ON}}*
- ❖ *ISO\_ISOOOUTPUT\_STATE {{Isolation OFF} {Sink ON}}*
- ❖ *ISO\_LSOOUTPUT\_STATE {{Level\_Shifter OFF} {Sink ON}}*
- ❖ *ISO\_ELSOUTPUT\_STATE {{ELS OFF} {Sink ON}}*

#### 4.4.6.4.3 Limitations

- ❖ The `analyze_waiver_correctness` command generates assertions only for power nets and not for ground nets.
- ❖ The `analyze_waiver_correctness` command is available as an LCA feature. Please contact [mvsupport@synopsys.com](mailto:mvsupport@synopsys.com) before using this command.

#### 4.4.6.5 Support for `generate_waiver_commands` Command

The `generate_waiver_commands` Tcl command is introduced to provide mechanism to generate waiver for cluster violation which will waive the related effect violations.

Using this command, you can create a waiver corresponding to cluster violation using all cause violation (all fields) of the cluster. As all causes should uniquely define a cluster violation.

##### Syntax

```
Usage: generate_waiver_commands      # Generates field based waiver commands corresponding
to violations
      -app <app>                  (List of apps :
                                     Values: cdc, constraints, design, dft,
   lint, lp, rdc, sdc, setup, upf)
      [-status <debug status>]       (Waivers will be generated for given violation state:
                                     Values: Acknowledged, NeedsInfo, Open,
   Waived, Waived_Temp)
      -viol <viol list>            (Field based waivers will be generated for given violation
id(s))
```

##### Example

```
generate_waiver_commands -app LP -viol { 30 31 }
```

#### 4.4.6.6 Support for `configure_waiver` Tcl Command

VC LP has introduced the `configure_waiver` Tcl command. This configuration command provides an additional control over waiver processing. You can instruct tool to ignore or consider waivers based on certain constraints.

##### Syntax

```
configure_waiver      # Configures waiver processing
      [-disable_multithread <field list>]
                           (List of fields for which MT will be considered during
filter processing)
```

```

[-fields <field list>] (List of fields which will be considered in filter
evaluation)
[-tags <tag list>] (List of tags on which waiver will be considered)
[-names <waiver list>] (List of names of waivers which will be considered)
[-severity <severity list>]
                           (List of severity on which waiver will be considered)
[-reset]                  (Reset configurations provided to the tool )
[-enable]                 (clear given configurations mentioned in the command
options)
[-disable]                (Append given configurations. Waivers meeting the
mentioned conditions will get ignored)

```

Note that the options available gets processed independently. See below example for details.

### Example

```
configure_waiver -fields {*:GroundNet:NetName} -name ISO_INST_MISSING_32 -disable
```

This command instructs tool to ignore all the fields in waivers containing *GroundNet:NetName* sub field. Waivers are processed as if this field was not mentioned in the waivers. This behavior will apply to all the waivers in the tool.

The command also instruct the tool to ignore the waiver with the name *ISO\_INST\_MISSING\_32*. Tool will not consider this waiver with the above name.

### Recommended Use Model:

- ❖ Design Read
- ❖ UPF Read
- ❖ LP checks
- ❖ configure\_waiver commands to the recommended settings
- ❖ Source waiver files.

Note that the waiver configurations should set before sourcing waivers. Otherwise the configurations will not apply. You may have to delete the existing waivers and source them back if you add configuration after sourcing waivers.

## 4.5 Tcl Query Commands

The following sections provide the list of design and low power query commands.

### 4.5.1 Design Query Commands

This section lists the various Tcl commands available to access design objects after the design is successfully built.

- ❖ all\_connected
- ❖ all\_designs
- ❖ all\_fanin
- ❖ all\_fanout
- ❖ all\_inputs
- ❖ all\_instances
- ❖ all\_outputs
- ❖ all\_registers

- ❖ get\_app\_var
- ❖ get\_blackbox
- ❖ get\_cells
- ❖ get\_designs
- ❖ get\_lib\_cells
- ❖ get\_lib\_pins
- ❖ get\_nets
- ❖ get\_sources
- ❖ get\_trace\_paths
- ❖ get\_trace\_path\_elements
- ❖ get\_object\_name
- ❖ get\_pins
- ❖ get\_ports
- ❖ link
- ❖ read\_verilog
- ❖ read\_sverilog
- ❖ read\_vhdl
- ❖ report\_blackbox
- ❖ report\_trace\_paths
- ❖ set\_app\_var
- ❖ set\_always\_on\_cell
- ❖ set\_blackbox
- ❖ set\_isolation\_cell
- ❖ set\_level\_shifter\_cell
- ❖ set\_power\_switch\_cell
- ❖ set\_retention\_cell
- ❖ set\_message\_severity
- ❖ set\_pin\_model
- ❖ set\_pg\_pin\_model

For more information on each of the Design query commands, see the *VC Static Command Reference Guide*.

#### 4.5.1.1 Common Behaviors

Many commands supported by VC LP construct collections of objects. In almost all cases, the commands allow various mechanisms for flexible specifications of what objects should be included in these collections. These mechanisms include:

- ❖ Wild-card name specification. These wild-cards can be “glob-style”, or “regular expression” style.
- ❖ Boolean expressions of object attributes and values

This section describes this behavior which is common to many commands. These details are explained here once, rather than for every switch for every such command.

#### 4.5.1.2 Regular Expression Behavior

Many commands have a `-regexp` switch. When used, this switch has the following behavior.

The command will view the patterns (name) argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the `=~` and `!~` filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl `regexp` command. When using the `-regexp` option, be careful how you quote the pattern argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding `".*"` to the beginning or end of the expressions, as needed.

For commands that have both switches, the `-regexp` and `-exact` options are mutually exclusive. You can specify only one of these options.

#### 4.5.1.3 Filter Expression Behavior

Many commands have a `-filter` switch which allows the collection to be composed as a function of attributes of the objects being collected.

`-filter` expression: Filters the collection with the specified expression.

For each object which might be added to the collection, the expression is evaluated based on the objects's attributes. If the expression evaluates to true, the object is included in the result.

To see the list of object attributes that you can use in the expression, use the `list_attributes -application -class <class>` command.

#### 4.5.1.4 Object Name Filtering

Almost all commands which return collections have a `patterns` field. This field is used as a convenience to filter based on object name and wild-card value for the name.

`patterns`

Creates a collection of objects whose names match the specified patterns. Patterns can include the \* (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the wildcards man page. Pattern matching is case sensitive unless you use the `-nocase` option. If `-filter` is also specified, then objects must evaluate to true for the filter expression, and the object name must pass the `patterns` field.

If you do not specify any of these arguments, the command uses \* (asterisk) as the default pattern.

#### 4.5.1.5 The upf\_terminal\_boundary\_transitive Application Variable

The `upf_terminal_boundary_transitive` application variable is introduced to add an enhanced functionality to `find_objects` UPF query command. The usage of the `upf_terminal_boundary_transitive` application variable in [Table 4-6](#).

**Table 4-6 The upf\_terminal\_boundary\_transitive Application Variable**

| Application Variable value | Transitive false                                                    | Transitive true                                                                    |
|----------------------------|---------------------------------------------------------------------|------------------------------------------------------------------------------------|
| False (default)            | Search in the given hierarchy.<br>Stops at hierarchical boundaries. | Search through hierarchical boundaries,<br>scope boundaries and terminal boundary. |

| Application Variable value | Transitive false                                                    | Transitive true                                                                                                  |
|----------------------------|---------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| True                       | Search in the given hierarchy.<br>Stops at hierarchical boundaries. | Search through hierarchical boundaries,<br>scope boundaries, but stops if a terminal<br>boundary is encountered. |

 **Note**

When the `upf_terminal_boundary_transitive` application variable is set to true, VC LP does not query the objects on the terminal boundary.

### Examples

Consider the following example

BlockA has a port in1. BlockA is a terminal boundary. Design top also has a port in1.

- ❖ When the `upf_terminal_boundary_transitive` application variable is set to true, and the following command is executed at the top scope:

```
find_objects . -pattern in1 -transitive true -object_type port
```

The following output is reported:

```
in1 (Not printing the terminal boundary port BlockA/in1)
```

- ❖ When the `upf_terminal_boundary_transitive` application variable is set to false (default), and the following command is executed at top scope

```
find_objects . -pattern in1 -transitive true -object_type port
```

The following output is reported:

```
in1 BlockA/in1
```

## 4.5.2 Low Power Query Commands

### 4.5.2.1 Power Domain Related Commands

#### 4.5.2.1.1 get\_power\_domains

This command returns the UPF power\_domains which match the specification in the command.

This command creates a collection of power domain objects which match the specification provided by the user. If no power domain matches the specification, then an empty collection is returned. If no option is specified, it returns all the power domains.

This command can work on a collection for each of the inputs (for example, `-of_object`). You can supply a collection instead of a single instance.

The output of this command is a collection; hence you can do all collection related operations on the return value of this command.

#### Syntax

```
%vc_static_shell> get_power_domains -help
Usage: get_power_domains      # Returns a collection of specific power domains
      [-regexp]           (Patterns are regular expressions)
      [-exact]            (Specifies exactly matching)
      [-nocase]           (Regexp matches case-insensitive)
      [-quiet]            (Suppresses all messages. Syntax error messages are not
                           suppressed)
```

```

[-primary_power <supply_net>]
    (Specifies primary power net)
[-primary_ground <supply_net>]
    (Specifies primary ground net)
[-primary_supply <supply_set>]
    (Specifies primary supply set)
[-iso_power <supply_net>]
    (Specifies isolation power net)
[-iso_ground <supply_net>]
    (Specifies isolation ground net)
[-iso_supply <supply_set>]
    (Specifies isolation supply set)
[-retention_power <supply_net>]
    (Specifies retention power net)
[-retention_ground <supply_net>]
    (Specifies retention ground net)
[-retention_supply <supply_set>]
    (Specifies retention supply set)
[-of_objects <strategy/cell>]
    (Specifies strategy (ISO, LS, RET) or cell)
[<patterns>]           (List of power domain name patterns)

```

## Use Models

```

%vc_static_shell> get_power_domains
{"ChipTop/GENPP", "ChipTop/GPRS", "ChipTop/INST", "ChipTop/MULT", "ChipTop/TOP"}

%vc_static_shell> get_power_domains -name G*
{"ChipTop/GENPP", "ChipTop/GPRS"}

%vc_static_shell> get_power_domains -primary_power "VDD"
 {"ChipTop/GENPP", "ChipTop/TOP"}

```

To retrieve all power domains whose primary supply is CORE1\_ss

```
get_power_domains -primary_supply CORE1_ss
 {"PD_CORE_1", "PD_CORE_2"}
```

To retrieve power domains whose iso supply is CORE1\_ss

```
%vc_static_shell> get_power_domains -iso_supply CORE1_ss
 {"PD_CORE_1"}
```

To retrieve power domain for i\_core\_1 instance

```
%vc_static_shell> get_power_domains -of_objects i_core_1
 {"PD_CORE_1"}
```

### 4.5.2.1.2 report\_power\_domain

This command reports about the specified power domain. This command describes the details of a given power domain. By default, it reports the following:

- ❖ Name

- ❖ Design instances which belong to this power domain
- ❖ Primary supply set for this domain
- ❖ Isolation supply set for this domain
- ❖ Retention supply set for this domain

## Syntax

```
%vc_static_shell> report_power_domain -help
Usage: report_power_domain      # Reports specific power domains
       [-name]                  (Reports power domain name)
       [-elements]               (Reports elements)
       [-strategy]               (Reports strategy name)
       [-supply_net]              (Reports supply net name)
       [-primary_supply]          (Reports primary supply set)
       [-isolation_supply]        (Reports isolation supply set)
       [-retention_supply]         (Reports retention supply set)
       [<power_domain>]           (List of power domain name patterns or collections)
```

## Use Model

```
%vc_static_shell> report_power_domain [get_power_domains -name INST*]
Power Domain      : INST
Full Name        : INST
Current Scope    :
Elements          : InstDecode
Level Shifter strategies :
Isolation strategies   : P__iso_0, P__iso_1, inst_iso_in, inst_iso_in_reset,
inst_iso_out
Retention strategies   : inst_ret
Power Switch strategies :
Supply Nets       : VDD, VDDI, VSS

Connections       : -- Power --  -- Ground --
Primary Supply Set : VDDIS  VSS
Isolation Supply Set : INST.default_isolation.power
Retention Supply Set : INST.default_retention.power
vc_static_shell> report_power_domain IN* -elements
Power Domain      : INST
Elements          : InstDecode
INST.default_isolation.ground
INST.default_retention.ground
```

### 4.5.2.2 Isolation Strategy Related Commands

#### 4.5.2.2.1 get\_isolation\_strategies

This command returns collection of isolation strategies. This command creates a collection of isolation strategies which match the specification provided by the user. If no isolation strategy matches the specification an empty collection is returned.

The output of this command is a collection; hence the user can do all collection related operations on the return value of this command.

## Syntax

```
%vc_static_shell> get_isolation_strategies -help
Usage: get_isolation_strategies      # Returns a collection of specific isolation
strategies
```

```

[-regexp]          (Patterns are regular expressions)
[-exact]          (Specifies exactly matching)
[-nocase]         (Regexp matches case-insensitive)
[-source <supply_set>] (Specifies source supply set)
[-sink <supply_set>]  (Specifies sink supply set)
[-domain <power_domain>]
                     (Specifies power domain)
[-iso_power <supply_net>]
                     (Specifies power supply net)
[-iso_ground <supply_net>]
                     (Specifies ground supply net)
[-supply <supply_set>] (Specifies supply set)
[-iso_control_signal <logic_net>]
                     (Specifies control logic net)
[-of_objects <power_domain/cell/pin/port>]
                     (Specifies power_domain, cell, pin or port)
[-filter <expression>] (Filter collection with 'expression')
[-quiet]
                     (Suppresses all messages. Syntax error messages are not
suppressed)
[<patterns>]        (List of isolation strategy name patterns)

```

## Use Model

```

%vc_static_shell> get_isolation_strategies
{"ChipTop/GENPP/mult_iso_in", "ChipTop/GENPP/mult_iso_in_isoen",
"ChipTop/GPRS/a_iso_2", "ChipTop/GPRS/a_iso_3", "ChipTop/GPRS/gprs_iso_in",
"ChipTop/GPRS/gprs_iso_in_reset", "ChipTop/GPRS/gprs_iso_out", "ChipTop/INST/a_iso_0",
"ChipTop/INST/a_iso_1", "ChipTop/INST/inst_iso_in", "ChipTop/INST/inst_iso_in_reset",
"ChipTop/INST/inst_iso_out", "ChipTop/MULT/mult_iso_out"}

%vc_static_shell> get_isolation_strategies -iso_power VDD
{"ChipTop/GENPP/mult_iso_in", "ChipTop/GPRS/gprs_iso_in", "ChipTop/GPRS/gprs_iso_out",
"ChipTop/INST/inst_iso_in", "ChipTop/INST/inst_iso_out", "ChipTop/MULT/mult_iso_out"}

%vc_static_shell> get_isolation_strategies -domain [get_power_domains -name INST*]
{"ChipTop/INST/a_iso_0", "ChipTop/INST/a_iso_1", "ChipTop/INST/inst_iso_in",
"ChipTop/INST/inst_iso_in_reset", "ChipTop/INST/inst_iso_out"}

```

To retrieve all isolation strategies which that use the supply set SS\_TOP

```
%vc_static_shell> get_isolation_strategies -supply SS_TOP
{"iso_pd1,iso_pd2"}
```

To retrieve isolation strategy that survive on pin CORE1/in1

```
%vc_static_shell> get_isolation_strategies -of_objects CORE1/in1
{"iso_pd1"}
```

To retrieve isolation strategy associated to the isolation instance iso\_inst

```
%vc_static_shell> get_isolation_strategies -of_object iso_inst
{"iso_top"}
```

To retrieve all isolation strategies that has clamp value 0

```
%vc_static_shell> get_isolation_strategies -filter {clamp_value == 0}
{"iso_top, iso_pd1,iso_pd2"}
```

#### 4.5.2.2.2 report\_isolation\_strategy

This command reports about the specified isolation strategies.

This command describes the details of a given isolation strategy. By default, it will report all the components of this strategy. User can ask to report only specific components.

#### Syntax

```
%vc_static_shell> report_isolation_strategy -help
Usage: report_isolation_strategy      # Reports specific isolation strategies
      [-name]                      (Reports isolation strategy name)
      [-domain]                     (Reports power domain)
      [-elements]                   (Reports elements)
      [-exclude_elements]          (Reports exclude elements)
      [-source]                     (Reports source supply set)
      [-sink]                       (Reports sink supply set)
      [-location]                  (Reports location type)
      [-iso_power]                 (Reports power supply net)
      [-iso_ground]                (Reports ground supply net)
      [-clamp_value]                (Reports clamp value)
      [-iso_sense]                 (Reports sense type)
      [-signal]                     (Reports signal logic net)
      [-applies_to]                 (Reports applies type)
      [-diff_supply]                (Reports diff supply only)
      [-map_cells]                  (reports cell_names written in the UPF map strategy
                                    itself)
      [<isolation_strategy>]       (List of isolation strategy name patterns or collections)
```

By default, it reports all information about the isolation strategy. If any specific option is specified, then it prints only the relevant information.

If the isolation is defined with -no\_isolation, remove some fields: power/ground, clamp, diff supply only, location, sense, and keep other fields. Put -no\_isolation within bracket after strategy name.

#### Use Model

```
❖ %vc_static_shell> report_isolation_strategy \
    [get_isolation_strategies -name *mult_iso_out*]
Isolation strategy   : mult_iso_in
Full Name            : GENPP/mult_iso_in
Power Domain         : GENPP
Power Supply Net    : VDD
Ground Supply Net   : VSS
Elements             :
Source Supply Set   :
Sink Supply Set     :
Clamp Value          : 1
Applies To           : inputs
Diff Supply Only     : 0
Isolation Signal    : mult_iso_in
Location              : self
Isolation Sense     : high
```

```
vc_static_shell> report_isolation_strategy \
    [get_isolation_strategies -name *mult_iso_out*] -signal
Isolation strategy      : mult_iso_in
Isolation Signal       : mult_iso_in
```

- ❖ The `report_isolation_strategy -map_cells` command reports the `cell_names` written in the UPF map strategy itself. The cells are not validated before printing out the result. The output of this command is the content of the map strategy as it is.

### Example

Consider the following UPF:

```
map_isolation_cell ISO_and_module -domain TOP -lib_cells {ISOLO ISOHI ISO_INV}
```

The following Tcl command:

```
report_isolation_strategy TOP/ISO_and_module -map_cells
```

reports the following output:

```
Isolation Strategy   : ISO_and_module
Isolation Lib Cells : ISOLO ISOHI ISO_INV
```

### Reporting Reason for Dropped Isolation policy

VC Static reports a reason code if any of the Isolation Policy is dropped. This is implemented in the `enable_lp_dump_debug_reports -report_dropped_policy_info` application variable. By setting this variable to true, VC Static creates a internal file (in the folder `vcst_rtdb/lpdb/debug_reports`), which contains information of the policy associated on a path.

To enable this support, set the `enable_lp_dump_debug_reports -report_dropped_policy_info` application variable to true.

Report Snippet:

```
=====
SrcNode : iso_en
SinkNode : CORE2/CORE22/iso_en
PolicyAssocNode : CORE2/iso_en
ResolvedPolicy : PD2/iso1
DroppedPolicies :
PolicyName: PD2/iso2
PathBased: True
DropReasonCode: -source failed
=====
```

The reasons for any dropped policy could be - source failed, -sink/-diff\_supply\_only failed, Lower priority strategy.

When an isolation strategy is dropped, the reason of dropping the strategy is reported in the `DroppedPolicyReport.rpt`. The `ISO_STRATEGY_DROPPED` violation is introduced to report the reason for dropping the strategy at `check_lp -stage upf`. The debug field `ReasonCode` points out the reason for dropping the strategy. This tag is disabled by default because it is not common in most customer flows. To enable it, use `configure_lp_tag -enable`.

The following is list of reason code reported:

- ❖ `src_power_gnd_mismatch`
- ❖ `src_power_mismatch`
- ❖ `src_gnd_mismatch`

- ❖ sink\_power\_gnd\_mismatch
- ❖ sink\_power\_mismatch
- ❖ sink\_gnd\_mismatch
- ❖ -sink/-diff\_supply\_only
- ❖ Low\_Priority\_Strategy
- ❖ On multi-driven node
- ❖ derived\_diverse
- ❖ derived\_diff\_only

The following is an example of the violation snippet:

```
ISO_STRATEGY_DROPPED
Description : Isolation strategy dropped
Severity : warning
Stage : UPF
Family : Isolation
Enabled : false
```

#### 4.5.2.3 Power Switch Related Commands

##### 4.5.2.3.1 get\_power\_switch\_strategies

This command returns a collection of power switch strategies in the design.

This command creates a collection of power switch objects which match the specification provided by the user. If no power switch matches the specification, then an empty collection is returned. If no option is specified, it returns all the power switches.

This command can work on a collection for each of the inputs (for example, -output\_supply, -input\_supply). You can supply a collection instead of a single supply net.

The output of this command is a collection; hence you can do all collection related operations on the return value of this command.

##### Syntax

```
%vc_static_shell> get_power_switch_strategies -help
Usage: get_power_switch_strategies      # Return a collection of specific power switches
      [-regexp]                  (Patterns are regular expressions)
      [-exact]                   (Specifies exactly matching)
      [-nocase]                  (Regexp matches case-insensitive)
      [-filter <expression>]    (Filter collection with 'expression')
      [-quiet]                   (Suppresses all messages. Syntax error messages are not
suppressed)
      [-output_supply <supply_port/supply_net>]
                           (Specifies output supply port or net)
      [-input_supply <supply_port/supply_net>]
                           (Specifies input supply port or net)
      [-control <logic_port/logic_net>]
                           (Specifies control supply port or net)
      [-ack_port <logic_port/logic_net>]
                           (Specifies acknowledge supply port or net)
      [-of_objects <cell/supply_net/power_domain>]
                           (Specifies cells, supply net or power domain)
```

[<patterns>] (List of power switch name patterns)

## Use Models

```
%vc_static_shell> get_power_switch_strategies
{"ChipTop/gprs_sw", "ChipTop/inst_sw", "ChipTop/mult_sw" }

%vc_static_shell> get_power_switch_strategies -output_supply VDDGS
{"ChipTop/gprs_sw" }

%vc_static_shell> get_power_switch_strategies -input_supply VDDI
 {"ChipTop/inst_sw" }

%vc_static_shell> get_power_switch_strategies -name *mult*
 {"ChipTop/mult_sw" }
```

To retrieve power switch strategies associated to power switch instance psw\_inst

```
%vc_static_shell> get_power_switch_strategies -of_object psw_inst
 {"psw_top"}
```

To retrieve all power switch strategies that have the error state defined

```
%vc_static_shell> get_power_switch_strategies -filter {has_error_state == true}
 {"psw_top, psw_pd1, "}
```

### 4.5.2.3.2 report\_power\_switch

This command reports information about the supplied power switch object.

This command reports the details of the power switch specified in the argument. You can provide a single power switch or a collection of power switches. You can also choose to report on specific components of power switch. Reports details of this power switch. If no option is given, by default it reports the

- ❖ Name
- ❖ Current scope
- ❖ Switch power domain
- ❖ Input port net and voltage
- ❖ Output net and voltage

If any option is specified, it reports only those items. If only -verbose is given, all aspects of the power switch are reported.

## Syntax

```
%vc_static_shell> report_power_switch -help
Usage: report_power_switch      # Reports specific power switches
       [-name]                  (Reports power switch name)
       [-input_supply_port]     (Reports input supply port)
       [-output_supply_port]    (Reports output supply port)
       [-control_port]          (Reports control port)
       [-on_state]                (Reports on state)
       [-on_partial_state]      (Reports on partial state)
       [-off_state]                (Reports off state)
       [-error_state]              (Reports error state)
```

|                                                                                                                                                                                 |                                                                                                                                                                                                          |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>[-supply_set]</code><br><code>[-domain]</code><br><code>[-ack_port]</code><br><code>[-ack_delay]</code><br><code>[-verbose]</code><br><code>[&lt;power_switch&gt;]</code> | (Reports power switch supply set)<br>(Reports power domain)<br>(Reports acknowledge port)<br>(Reports acknowledge delay)<br>(Reports verbose)<br>(List of the power switch name patterns or collections) |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Use Model

```
%vc_static_shell> report_power_switch *inst*
Power Switch      : inst_sw
Full Name         : inst_sw
Current Scope    :
Power Domain     : TOP
Input Port Net   : VDDI
Output Port Net  : VDDIS
Control Port     : inst_on
On State          : { state2001 in !(inst_on) }
Off State         :
Error State       :
Partial On State :
Ack Port          :
Ack Delay         :

Connections       : -- Power --- Ground --
Supply Set        :

vc_static_shell> report_power_switch [get_power_switch_strategy -name *mult*]
Power Switch      : gprs_sw
Full Name         : gprs_sw
Current Scope    :
Power Domain     : GPRS
Input Port Net   : VDDG
Output Port Net  : VDDGS
Control Port     : gprs_on
On State          : { state2002 in !(gprs_on) }
Off State         :
Error State       :
Partial On State :
Ack Port          :
Ack Delay         :

Connections       : -- Power --- Ground --
Supply Set        :

vc_static_shell> report_power_switch *gprs* -on_state
Power Switch      : gprs_sw
On State          : { state2002 in !(gprs_on) }
```

### 4.5.2.4 Retention Strategy Related Commands

#### 4.5.2.4.1 get\_retention\_strategies

This command returns the retention strategies in a collection.

This command creates a collection of retention strategies that match the specification provided by the user. If no retention strategies match the specification, then an empty collection is returned. If no option is specified, it returns all the retention strategies.

This command can work on a collection for each of the inputs (for example, -save, -restore). You can supply a collection instead of a single object.

The output of this command is collection; hence you can do all collection related operations on the return value of this command.

## Syntax

```
%vc_static_shell> get_retention_strategies -help
Usage: get_retention_strategies      # Returns a collection of specific retention
strategies
      [-regexp]                  (Patterns are regular expressions)
      [-exact]                   (Specifies exactly matching)
      [-nocase]                  (Regexp matches case-insensitive)
      [-domain <power_domain>]   (Specifies power domain)
      [-retention_power <supply_net>] (Specifies retention power net)
      [-retention_ground <supply_net>] (Specifies retention ground net)
      [-supply <supply_set>] (Specifies supply set)
      [-save <logic_net>] (Specifies save logic net)
      [-restore <logic_net>] (Specifies restore logic net)
      [-of_objects <cell/power_domain>] (Specifies cell or power_domain)
      [-filter <expression>] (Filter collection with 'expression')
      [-quiet]                   (Suppresses all messages. Syntax error messages are not
suppressed)
      [<patterns>]                (List of retention strategy name patterns)
```

## Use Model

```
%vc_static_shell> get_retention_strategy
{"ChipTop/GENPP", "ChipTop/GPRS", "ChipTop/INST", "ChipTop/MULT", "ChipTop/TOP"}

%vc_static_shell> get_retention_strategy -name G*
{"ChipTop/GENPP", "ChipTop/GPRS"}

%vc_static_shell> get_retention_strategy -primary_power "VDD"
 {"ChipTop/GENPP", "ChipTop/TOP"}
```

To retrieve all retention strategies that use the supply set SS\_TOP

```
%vc_static_shell> get_retention_strategies -supply SS_TOP
 {"ret_pd1", "ret_pd2"}
```

To retrieve the retention strategy associated to retention instance ret\_inst

```
%vc_static_shell> get_retention_strategies -of_object ret_inst
 {"ret_pd1"}
```

To retrieve retention strategies that save signal sense as high

```
%vc_static_shell> get_retention_strategies -filter { save_level == high }
```

```
{"ret_pd1, ret_pd2, ret_top"}
```

#### 4.5.2.4.2 report\_retention\_strategy

This command reports details of the supplied retention strategy. This command reports the details of the retention strategy specified in the argument. You can provide a single retention strategy or a collection of retention strategies. You can also choose to report on specific components of the retention strategy.

Report details of the retention strategy. By default, it reports the following:

- ❖ Domain
- ❖ Elements
- ❖ exclude\_elements (if any)
- ❖ Retention power
- ❖ Retention ground
- ❖ Retention supply set
- ❖ Save signal
- ❖ Restore signal
- ❖ Save condition
- ❖ Restore condition
- ❖ Instances
- ❖ -transitive (if specified).

If any option is specified, then the command prints only the relevant values.

#### Syntax

```
vc_static_shell> report_retention_strategy -help
Usage: report_retention_strategy      # Reports spcific retention strategies
       [-name]                  (Reports retention strategy name)
       [-domain]                 (Reports power domain)
       [-elements]                (Reports elements)
       [-ret_power]                (Reports power supply net)
       [-ret_ground]               (Reports ground supply net)
       [-retention_supply]        (Reports retention supply set)
       [-save]                     (Reports save logic net)
       [-restore]                  (Reports restore logic net)
       [-save_condition]          (Reports save condition)
       [-restore_condition]       (Reports restore condition)
       [-retention_condition]     (Reports retention condition)
       [<retention_strategy>]      (List of retention strategy name patterns or collections)
```

#### Use Model

```
%vc_static_shell> report_retention_strategy
Retention strategy    : gprs_ret
Full Name             : GPRS/gprs_ret
Power Domain          : GPRS
Save Signal           : gprs_save(high)
Restore Signal         : gprs_nrestore(low)
Power Supply Net      : VDDG
Ground Supply Net     : VSS
```

```

Elements          :
Retention Supply Set : ChipTop/GPRS.gprs_ret.supply
                      VDDG                         VSS
Save Condition    :
Restore Condition :
Retention Condition :

-----
Retention strategy   : inst_ret
Full Name           : INST/inst_ret
Power Domain        : INST
Save Signal         : inst_save(high)
Restore Signal      : inst_nrestore(low)
Power Supply Net   : VDDI
Ground Supply Net  : VSS
Elements          :
Retention Supply Set : ChipTop/INST.inst_ret.supply
                      VDDI                         VSS
Save Condition    :
Restore Condition :
Retention Condition :

%vc_static_shell> report_retention_strategy \
                  [get_retention_strategies -name *gprs*]
Retention strategy   : gprs_ret
Full Name           : GPRS/gprs_ret
Power Domain        : GPRS
Save Signal         : gprs_save(high)
Restore Signal      : gprs_nrestore(low)
Power Supply Net   : VDDG
Ground Supply Net  : VSS
Elements          :
Retention Supply Set : ChipTop/GPRS.gprs_ret.supply
                      VDDG                         VSS
Save Condition    :
Restore Condition :
Retention Condition :
vc_static_shell> report_retention_strategy *inst* -domain
Retention strategy   : inst_ret
Power Domain        : INST

```

#### 4.5.2.5 Supply Net and Port Related Commands

##### 4.5.2.5.1 get\_root\_supply\_net

This command returns the top level root supply net of any pin of a design.

Given a pin on a design, leaf cell instance pins or hierarchical instance pins, returns the top level supply net which provides the supply for this pin. This computation can happen in three modes:

- ❖ **UPF only mode:** From the given pin, traversal is performed on UPF network.
  - ◆ If the pin is functional pin of a cell, then the related power pin (RPP) of that pin is computed. P=RPP (pin).
  - ◆ If the pin is a supply itself, then P=pin
  - ◆ If the pin is hierarchical instance pin then P=pin

Then the immediate UPF supply net connected to that P is computed. For that computation, the following precedence order is honored:

- ❖ connect\_supply\_net to that power pin.
- ❖ Strategy power pin if that cell is low power cell such as Isolation cell or level shifter cell.
- ❖ Domain supply.

Consider that the immediate UPF supply net is upf\_supply\_I. Then the UPF network is further traversed up if there is a connect\_supply\_net to upf\_supply\_I. This traversal continues until either reach the topmost supply net and no further upward traversal possible.

- ❖ **Netlist only Mode:** In this mode, power network is traversed upward only PG connectivity in netlist, UPF connectivity (connect\_supply\_net) is not taken into account. From a given pin, PG network is traversed up as much as possible and the supply net connected to that physical net is returned.
- ❖ **Combined Mode:** This is the default mode when the user does not specify any mode. In this mode, physical connectivity is given higher precedence compared to UPF connectivity. But where physical connectivity is missing, it uses UPF connectivity to traverse. Hence, in a fully PG connected design, this mode is equivalent to netlist only mode. And in a design which has no PG connectivity, this mode is equivalent to UPF only mode.

## Syntax

```
%vc_static_shell> get_root_supply_net -help
Usage: get_root_supply_net      # Returns a collection of root supply nets for specific
design pins/ports or UPF supply ports or crossover node
      [-power]                  (Returns root power supply net)
      [-ground]                 (Returns root ground supply net)
      [-upf]                     (Bases on UPF information)
      [-netlist]                 (Bases on PG connectivity)
      [-quiet]                   (Suppresses all messages. Syntax error messages are not
suppressed)
      [-nocase]                  (Case insensitive)
      <pin>                      (List of design pin names or collections)
```

## Use Models

```
%vc_static_shell> get_root_supply_net [get_pins \
    ChipTop_U_decompressor_ScanCompression_mode/dout[43] ]
{"ChipTop/VDD"}
%vc_static_shell> get_root_supply_net [get_pins \
    ChipTop_U_decompressor_ScanCompression_mode/dout[43]] -netlist
{"ChipTop/VDD"}
%vc_static_shell> get_root_supply_net [get_pins \
    ChipTop_U_decompressor_ScanCompression_mode/dout[43]] -ground
{"ChipTop/VSS"}
```

### 4.5.2.5.2 get\_root\_supply\_path

The get\_root\_supply\_path Tcl command takes the UPF supply net as an argument, and returns the collection of the entire path stack of supply net/ports up to root supply net (state source).

## Syntax

```
get_root_supply_path <upf-supply-net-name>
Usage: get_root_supply_path will return collection of upf root supply net (state
source) and all intermediate upf supply nets and ports in the path.
```

## Example Output

```
get_root_supply_path inst1/VDD
{ "VDD", "VDD", "inst1/VDD", "inst1/VDD"}
```

### 4.5.2.5.3 get\_related\_supply\_pin

This command returns the related supply pin for a logic pin of a leaf level cell, using property set in the database.

This command returns the related power pin of a logic pin on a cell. It is useful for cells which have multiple power rails such as dual-rail always-on buffer or multi-rail macro such as memory modules.

This command can only be applied to leaf level cell pins and returns supply pins in a collection.

#### Syntax

```
%vc_static_shell> get_related_supply_pin -help
Usage: get_related_supply_pin      # Returns a collection of the related pin
       [-power]                  (Returns related power net)
       [-ground]                 (Returns related ground net)
       [-bias]                   (Returns related bias net)
       [-quiet]                  (Suppresses all messages. Syntax error messages are not
suppressed)
       <pin>                     (List of design pin names or collections)
```

This query can be asked on a pin of an instance of library cell (library DB). It returns the power or ground pin (depending on if -power or -ground is specified, default option is -power), based on the DB cell attribute. This query can be asked on a pin of an instance of the library cell (library DB). It returns the power or ground pin (depending on if -power or -ground is specified, default option is -power), based on the DB cell attribute.

#### Use Model

```
%vc_static_shell> get_related_supply_pin U153/ZN
{ "U153/VDD" }
%vc_static_shell> get_related_supply_pin U153/ZN -ground
{ "U153/VSS" }
%vc_static_shell> get_related_supply_pin [get_pins U230/ZN]
{ "U230/VDD" }
```

### 4.5.2.5.4 get\_upf\_connection

The query command returns the UPF supply net that drives a given pin or port of a design. It returns a supply net that is an immediate connection and not root supply connection.

#### Syntax

```
vc_static_shell> get_upf_connection -help
Usage: get_upf_connection      # Returns immediate connection of specified type
       -type connection_type (Specifies connection type:
                             Values: csn, spa, srsn)
       [-power]              (Specifies power supply net)
       [-ground]             (Specifies ground supply net)
       [-pwell]               (Specifies pwell supply net)
       [-nwell]               (Specifies nwell supply net)
       [-driver]              (Specifies driver supply net)
       [-receiver]            (Specifies receiver supply net)
       [-repeater]            (Specifies repeater supply net)
```

|                                                                                   |                                                                                                            |
|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| <code>[-literal]</code><br><code>[-quiet]</code><br><code>&lt;pin port&gt;</code> | (Specifies literal supply net)<br>(Suppresses all messages.)<br>(pin or port name patterns or collections) |
|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|

## Examples

```
%vc_static_shell> get_upf_connection ISO_TOP/VDD -type csn -power
{"VDD_SW_ISO"}
vc_static_shell> get_upf_connection ISO_TOP/VSS -type csn -ground {"VSS"}

%vc_static_shell> get_upf_connection ISO_TOP/A -type srsn
{"VDD_SW_SRSN", "VSS"}

%vc_static_shell> get_upf_connection in -type spa -driver
{"VDD_CORE_1", "VSS"}

UPF BUF_TOP instance having csn defined
vc_static_shell> get_upf_connection BUF_TOP/VDD -type csn -power
{"VDD_SW_BUF"}
```

### To retrieve the ground supply net

```
%vc_static_shell> get_upf_connection BUF_TOP/A -type csn -ground
{"VSS"}
```

### To retrieve the SRSN supply net

```
%vc_static_shell> get_upf_connection ISO_TOP/A -type srsn
{"VDD_SW_SRSN", "VSS"}
```

### Port having SPA defined with -driver supply

```
%vc_static_shell> get_upf_connection in -type spa -driver
{"VDD_CORE_1", "VSS"}
```

### To retrieve nwell and pwell ports:

Consider the following UPF:

```
connect_supply_net Nwell -ports {blck1/macro_bus_cell1/vddsm}
connect_supply_net Pwell -ports {blck1/macro_bus_cell1/gndsm}
```

Tcl outputs:

- ❖ `get_upf_connection -type csn blck1/macro_bus_cell1/vddsm -nwell`  
Nwell
- ❖ `get_upf_connection -type csn blck1/macro_bus_cell1/gndsm -pwell`  
Pwell

### To retrieve -literal\_supply:

Consider the following UPF

```
set_port_attributes -literal_supply {LIT_SS} -ports {blck1/in1}
```

The following Tcl command reports the literal\_supply set by the set\_port\_attribute:

```
get_upf_connection -literal blck1/in1 -type spa
{VDD_LIT VSS}
```

## 4.5.2.6 PST Related Commands

### 4.5.2.6.1 get\_pst

This command returns the PST object as per user specified criteria.

This command returns a collection of PST objects from the UPF data model. You can further query on these PST objects using commands such as report\_pst, get\_pst\_states, and so on.

#### Syntax

```
%vc_static_shell> get_pst -help
Usage: get_pst      # Returns a collection of Power State Table objects
       [-quiet]           (Suppresses all messages. Syntax error messages are not
suppressed)
       [<patterns>]        (List of pst name patterns)
```

#### Use Model

```
%vc_static_shell> get_pst
{"ChipTop/chiptop_pst"}
%vc_static_shell> get_pst -name chiptop*
{"ChipTop/chiptop_pst"}
%vc_static_shell> get_pst -name dummy
Warning: No upf_state_table objects matched 'dummy' (SEL-004)
Error: Nothing matched for collection (SEL-005)
```

### 4.5.2.6.2 report\_pst

This command prints out the specified PST and all the states in a tabular manner.

This command prints the details of the specified PST in a tabular manner. Each row is a state in the PST with the values of the supply nets of that PST. Reports the PST state table in tabular format. Each row is a state in the PST. The columns are the supply nets. The values in each cell of the table are the port states of the supply ports/nets that are a part of this PST.

#### Syntax

```
%vc_static_shell> report_pst -help
Usage: report_pst      # Reports specific power state tables
       [<pst>]           (List of power state table name patterns or collections)
```

#### Use Model

```
%vc_static_shell> report_pst
PST Name          : chiptop_pst
Full Name         : chiptop_pst
              VDD | VDDI | VDDIS | VDDG | VDDGS | VDDM | VDDMS |
chiptop_pst/s0:  HV |   HV |     HV |   HV |     HV |   HV |     HV |
chiptop_pst/s01: HV |   HV |    OFF |   HV |     HV |   HV |     HV |
chiptop_pst/s02: HV |   HV |     HV |   HV |    OFF |   HV |     HV |
chiptop_pst/s03: HV |   HV |     HV |   HV |     HV |   HV |    OFF |
chiptop_pst/s1:  HV |   LV |     LV |   LV |     LV |   HV |     HV |
chiptop_pst/s11: HV |   LV |    OFF |   LV |     LV |   HV |     HV |
chiptop_pst/s12: HV |   LV |     LV |   LV |    OFF |   HV |     HV |
chiptop_pst/s13: HV |   LV |     LV |   LV |     LV |   HV |    OFF |
-----
```

#### 4.5.2.6.3 get\_pst\_states

Returns the PST states in the specified PST.

This command returns a collection of PST states, from the specified PSTs. If no PST is specified, then it searches for all available PSTs. You can further query on the PST states described in subsequent commands.

##### Syntax

```
%vc_static_shell> get_pst_states -help
Usage: get_pst_states      # Returns a collection of states (rows in PST) in the specific
PST
      [-of_objects <state_table>]
                           (Specifies state table name patterns or collections)
      [-quiet]           (Suppresses all messages. Syntax error messages are not
suppressed)
      [<patterns>]       (List of pst state name patterns)
```

##### Use Model

```
%vc_static_shell> get_pst_states -pst [get_pst -name chiptop*]
{"s13", "s01", "s02", "s11", "s0", "s03", "s12", "s1"}

%vc_static_shell> get_pst_states -pst [get_pst -name chiptop*] -name s03
{"s03"}

%vc_static_shell> get_pst_states -name s0*
 {"s01", "s02", "s0", "s03"}
```

To retrieve the pst states of pst states top\_pst

```
%vc_static_shell> get_pst_states -of_objects top_pst
 {"top_pst/all_on", "top_pst/sub_off", "top_pst/all_off"}
```

#### 4.5.2.6.4 report\_pst\_state

Reports specific PST states. This command returns specific states defined in PSTs. You can further query on the states to get information about the valid/invalid states of each PST.

##### Syntax

```
%vc_static_shell> report_pst_state -help
Usage: report_pst_state      # Reports specific PST states
      [-only_invalid]        (Include only invalid PST states)
      [-only_valid]          (Include only valid PST states)
      [<pst_state>]         (List of PST state name patterns or collections)
```

##### Use Model

```
set_scope /
.....
create_pst top_pst -supplies {VDD_TOP VSS_TOP}
add_pst_state s1 -pst top_pst -state {TOP_V12 gon}
add_pst_state s2 -pst top_pst -state {TOP_V10 gon}

set_scope BLOCK_A
.....
create_pst block_a_pst -supplies {VDD1 VSS1}
```

```

add_pst_state s1 -pst block_a_pst -state {BLKA_V12 gon}
add_pst_state s2 -pst block_a_pst -state {BLKA_V08 gon}
set_scope /
connect_supply_net VDD_TOP -ports { VDD_TOP BLOCK_A/VDD1 }

```

Note: \*\_V12 represents 1.2V, \*\_V10 represents 1.0V, \*\_V08 represents 0.8V

```
%vc_static_shell> report_pst_state -only_invalid
```

```

PST State          : s2
Full Name         : BLOCK_A/block_a_pst/s2
                     VDD_TOP| VSS_TOP|
BLOCK_A/block_a_pst/s2: BLKA_V08|      gon|
-----
```

```

PST State          : s2
Full Name         : top_pst/s2
                     VDD_TOP| VSS_TOP|
top_pst/s2: TOP_V10|      gon|
-----
```

```
%vc_static_shell> report_pst_state -only_valid
```

```

PST State          : s1
Full Name         : BLOCK_A/block_a_pst/s1
                     VDD_TOP| VSS_TOP|
BLOCK_A/block_a_pst/s1: BLKA_V12|      gon|
-----
```

```

PST State          : s1
Full Name         : top_pst/s1
                     VDD_TOP| VSS_TOP|
top_pst/s1: TOP_V12|      gon|
-----
```

```
%vc_static_shell>report_pst_state BLOCK_A/block_a_pst/s1
PST State          : s1
Full Name         : BLOCK_A/block_a_pst/s1
                     VDD_TOP| VSS_TOP|
BLOCK_A/block_a_pst/s1: BLKA_V12|      gon|
-----
```

#### 4.5.2.6.5 **get\_num\_merged\_pst\_states**

This command returns the number of states from the merged PST. This command returns the number of unique PST states after all individual PSTs are merged into one PST. This command can be used after read\_upf is successfully completed.

##### Syntax

```
%vc_static_shell> get_num_merged_pst_state -help
Usage: get_num_merged_pst_states      # Returns number of states in merged PST
       [-exclude]                  (Exclude supplies which are not used in any PST)
       [-limit <double>]           (Stop counting after this many states)
```

##### Use Model

```

set_scope /
.....
create_pst top_pst-supplies {VDD_TOP VSS_TOP}
add_pst_state s1 -pst top_pst-state {TOP_V12 gon}
add_pst_state s2 -pst top_pst-state {TOP_V10 gon}
add_pst_state s3 -pst top_pst-state {TOP_V08 gon}

create_pst iso_pst-supplies {VDD_ISO VSS_ISO}
add_pst_state s1 -pst iso_pst-state {ISO_V12 gon}
add_pst_state s2 -pst iso_pst-state {ISO_V10 gon}
add_pst_state s3 -pst iso_pst-state {ISO_V08 gon}

set_scope BLOCK_A
.....
create_pst block_a_pst -supplies {VDD1 VSS1}
add_pst_state s1 -pst block_a_pst -state {BLKA_V12 gon}
add_pst_state s2 -pst block_a_pst -state {BLKA_V08 gon}

set_scope /
connect_supply_net VDD_TOP -ports { VDD_TOP BLOCK_A/VDD1 }
Note: *_V12 represents 1.2V, *_V10 represents 1.0V, *_V08 represents 0.8V

```

During `read_upf`, VC LP merges the three PSTs mentioned in the example and the merged PST has all the possible cross combinations of these three PSTs.

The `get_num_merged_pst_states` command returns total num of states in merged PST.

In above example, `get_num_merged_pst_states` returns 6.

```
%vc_static_shell> read_upf merged_states.upf
%vc_static_shell> get_num_merged_pst_states
6
```

#### 4.5.2.6.6 report\_system\_pst

In designs with a large number of power state tables, it is difficult for you to predict the resulting system power states. However, VC LP provides a command, `report_system_pst`, which provides several ways to display the system power state table.

##### Syntax

```
vc_static_shell> report_system_pst -help
Usage: report_system_pst      # reports system power state table for all or some supplies
       [-summary]           (Show only a summary of the table size)
       [-supplies <supplies>] (List of supply net name patterns or collections)
       [-tables <tables>]    (List of power state table name patterns or collections)
       [-format <format>]    (Format, default comma separated values:
                           Values: csv, html, upf)
       [-symbolic]           (Show state names instead of voltage values)
       [-exclude]            (Exclude supplies which are not used in any PST)
       [-include]             (Include "uninteresting" supplies with one port state)
       [-loop]
       [-limit <double>]    (Stop counting after this many states)
       [-transient]           (Include transient power states)
       [-expand_triplets]    (Show voltage values in triplets format)
       [-force]               (Force computation of potentially huge output)
```

## Use Model

The following is a simple example of a design with two power state tables. This is not a full UPF file but just an illustration to convey the idea.

```
create_power_domain d_top -include_scope

create_supply_net VDD1
add_port_state VDD1 -state "H1 1.2"
create_supply_net VDD2
add_port_state VDD2 -state "H2 1.2" -state "Z2 off"

create_pst top_pst -supplies      "VDD1 VDD2"
add_pst_state t0 -psttop_pst -state "H1    H2"
add_pst_state t1 -psttop_pst -state "H1    Z2"
set_scope ua
create_power_domain d_a -include_scope

create_supply_net VDD3
add_port_state VDD3 -state "H3 1.2" -state "L3 1.1"
create_supply_net VDD4
add_port_state VDD4 -state "H4 1.2" -state "Z4 off"

create_pst ua_pst -supplies      "VDD3 VDD4"
add_pst_state a0 -pst ua_pst -state "H3    H4"
add_pst_state a1 -pst ua_pst -state "L3    Z4"
```

The following are several examples of the complete command lines and outputs:

```
%vc_static_shell>report_system_pst -summary
Supplies: 3 Port states: 7
Potential system states: 8 Actual system states: 4
```

```
%vc_static_shell>report_system_pst
VDD1,VDD2,ua/VDD3,ua/VDD4
1.2,1.2,1.2,1.2
1.2,1.2,1.1,off
1.2,off,1.2,1.2
1.2,off,1.1,off
```

Use the `-expand_triplets` option to expand the triplet combinations in the output. When the `-expand_triplets` option is not specified, only the nominal voltage is reported. One system power state may expand to multiple lines in the output.

```
vc_static_shell> report_system_pst -loop -include
VDD,VDD1,VDD2,VSS
1.1,1.15,1.1,0
```

```
vc_static_shell> report_system_pst -loop -include -expand_triplets
VDD,VDD1,VDD2,VSS
1,1,1,0
1.1,1.15,1.1,0
1.2,1.3,1.2,0
```

To get a list of multiple supplies, you must create a collection first.

```

foreach net {ua/VDD4 ua/VDD3} { append_to_collection s [get_supply_nets $net] }
report_system_pst -supplies $s
ua/VDD4,ua/VDD3
1.2,1.1
off,1.2

vc_static_shell>report_system_pst -format upf
create_pstsystem_pst -supplies "VDD1 VDD2 ua/VDD3 ua/VDD4"
add_pst_state sys000 -pstsystem_pst -state "H1 H2 H3 H4"
add_pst_state sys001 -pstsystem_pst -state "H1 H2 L3 Z4"
add_pst_state sys002 -pstsystem_pst -state "H1 Z2 H3 H4"
add_pst_state sys003 -pstsystem_pst -state "H1 Z2 L3 Z4"

```

The following new error messages and a new option `-force` is added in the `report_system_pst` command to prevent the abrupt behavior.

The following error messages are added for large designs with billions of system power states:

- ❖ vc\_static\_shell> report\_system\_pst -summary
 

*Supplies: 125 Port states: 250*  
*Potential system states: 4.24e+37 Actual system states: 1.70e+12*
  - ❖ vc\_static\_shell> report\_system\_pst
 

[Error] PST\_STATE\_COUNT: *The output you requested would take some time to generate because it will be 1.70e+12 lines long*

To avoid this error message, consider requesting fewer supplies, or using other debug commands.  
To print anyway, add `-force` to the command.
  - ❖ vc\_static\_shell> report\_system\_pst -loop
 

[Error] PST\_LOOP\_COUNT: *The output you requested would take some time to generate because the -loop option requires iterating 4.24e+37 potential states.*

To avoid this error message, consider not using `-loop`, or request for fewer supplies, or use other debug commands.
  - ❖ If you do not want to receive these error messages, and print the output anyway, use the `-force` option.
- If you use the `-force` option, neither of the messages will be reported, and VC LP performs the huge computation. For small designs, the `report_system_pst` behavior is consistent with the previous behaviors.

### Note

You cannot change the threshold. For the STATE message, the threshold above which the message appears is  $1e8$  (one hundred million). For the LOOP message, the threshold above which the message appears will be  $1e10$  (ten billion).

### Don't-Care Operator

Character: `"*"`

Example: If a design has 20 independent switchable power supplies, there are  $2^{20}$  (a million) actual system power states. However, using the don't-care operator, only one line is printed by `report_system_pst` command containing a don't-care character for each supply.

## Partial Don't-Care Operator

*Character:* "?"

Example: If a supply has four power states and two always appear together in the PSTs, they will be represented as partial dont-care operators.



The –loop option returns all system power states which is slower but more precise.



Use the –summary option first, which prints the potential state count. If this is more than 1e9 (billions), the runtime may be too long.

### 4.5.2.6.7 get\_equiv\_power\_ports

Given a power net, returns all the equivalent power nets (as per PG connectivity) and/or UPF specification.

#### Syntax

```
%vc_static_shell> get_equiv_power_ports -help
Usage: get_equiv_power_ports      # Returns a collector of supply ports or nets which are
equivalent to the specific supply net
      [-upf]                  (Bases on UPF information)
      [-netlist]                (Bases on PG connectivity)
      [-quiet]                 (Suppresses all messages. Syntax error messages are not
suppressed)
      <supply_net>            (Specifies supply net name patterns or collections)
```

### 4.5.2.6.8 get\_crossovers

Crossover/Protection/Retention/Isolation Get collection of crossovers, as specified.

#### Syntax

```
vc_static_shell> get_crossovers -help
Usage: get_crossovers      # Returns a collection of specific crossovers
      [-source <power_domain>]          (Specifies source power domain)
      [-dest <power_domain>]           (Specifies destination power domain)
      [-seg_source <supply_net or supply_port>]
   (Specifies segment source supply)
      [-seg_sink <supply_net or supply_port>]
   (Specifies segment sink supply)
      [-from_signal <net_or_pin>]
   (Specifies start signal)
      [-to_signal <net_or_pin>]
   (Specifies end signal)
      [-through_signal <net_or_pin>]
   (Specifies present signal)
      [-only_dropped]                 (only dropped crossovers)
      [-include_dropped]              (include dropped crossovers)
      [-filter <expression>]          (Filter collection with 'expression')
      [-quiet]                      (Suppresses all messages. Syntax error messages are not
suppressed)
```

#### Use Model

To retrieve all crossovers in the design with an un-driven source

```
%vc_static_shell> get_crossovers -filter {source_type == undriven}
{"undriven CORE4/in1 CORE4/dff1/D"}
```

To retrieve all crossovers whose driver is un-driven and load is unused

```
%vc_static_shell> get_crossovers -filter {source_type == undriven || sink_type ==
unused}
{"CORE3/dff1/Q CORE3/out1 unused", "undriven CORE4/in1 CORE4/dff1/D"}
```

To retrieve all crossovers that starts from top-level ports

```
%vc_static_shell> get_crossovers -filter {source_type == port}
{"in1 CORE1/in1 CORE1/dff1/D", "clk CORE1/clk CORE1/dff1/CP"}
```

To get a collection of dropped crossovers

```
get_crossovers -dropped -filter
{ dropped_reason == INVALID_SAME_DOMAIN_XOVER }
```

The result of the `get_crossovers dropped` contains only the source node and sink node. This option can be used along with the `-source`, `-dest`, `-from_signal`, `-to_signal`. To enable this feature, set the `enable_lp_dump_debug_reports` with the `-enable_report_ignored_xover_paths` option.

The new attribute `dropped_reason` is introduced for crossover object. The `get_crossovers -dropped` can be used with the `-filter` option with this attribute.

### Example

```
%get_crossovers -dropped -filter
{ dropped_reason == INVALID_SAME_DOMAIN_XOVER }
```

The options for the `dropped_reason` attribute are as follows:

- ❖ `ROOT_NODE_SINK_NODE`: Root node is a sink node
- ❖ `NO_BNDRY_AON_PROT_SRSN_SPA`: No power domain boundary/AON cell/Protection Device/SRSN/SPA Encountered during path traversal
- ❖ `UNCONNECTED_DRIVER`: UNCONNECTED driver and no AON/Protection cell/op pin found on path, TCL var: `ignore_unconnected_driver`
- ❖ `INVALID_SAME_DOMAIN_XOVER`: Invalid same domain crossover
- ❖ `SINK_NODE_PIN_OF_DIODE_PHYSICAL`: Sink node is pin of a Diode or Physical-only cell
- ❖ `UNCONNECTED_LOAD`: UNCONNECTED load/sink node, TCL var: `ignore_unconnected_load`
- ❖ `LITERAL_CONSTANT_DRIVEN_MACRO`: Literal constant driven macro and not in same hierarchy OR not direct connection, command: `lp_literal_constant -macro_driven`
- ❖ `CONSTANT_DRIVEN_MACRO_SAME_HIERARCHY`: Sink is a macro pin and is driven by a constant, in same hierarchy, no policy present on sink
- ❖ `CONSTANT_DRIVEN_MACRO_NOT_IN_SAME_HIERARCHY`: Sink is a macro pin and is driven directly by a constant, no power domain crossing in between, not in same logical hierarchy
- ❖ `LOGIC_DRIVING_PG_PATH`: Sink is a pg pin/net and is driven directly by a logic port/pin/net, no power domain crossing in between. No protection device can be placed here.

When the `enable_lp_dump_debug_reports -enable_report_ignored_xover_paths` command is set in the Tcl file, `get_crossovers -only_dropped` command return only the dropped crossovers and `get_crossovers -include_dropped` command return all normal crossover paths and dropped crossover paths.

## Example

```
vc_static_shell> get_crossovers
Warning: No crossover objects matched '*' (SEL-004)
vc_static_shell> get_crossovers -only_dropped
{"restore CORE1/restore", "clk CORE1/clk", "save1 save1", "in2 in2", "in1 CORE1/ff_i1/D", "in1
CORE1/ff_i2/D", "in1 CORE1/ff_i2/CP"}
vc_static_shell> get_crossovers -include_dropped{"restore CORE1/restore", "clk
CORE1/clk", "save1 save1", "in2 in2", "in1 CORE1/ff_i1/D", "in1 CORE1/ff_i2/D", "in1
CORE1/ff_i2/CP"}
```

### 4.5.2.6.9 get\_crossover\_info

Returns the source signal, destination signal, source domain, destination domain, iso/ls strategies and iso/ls devices present on a crossover.

#### Syntax

```
%vc_static_shell> get_crossover_info -help
Usage: get_crossover_info      # Returns specific power cross-over information
       [-src]                  (Returns source signal information)
       [-src_domain]           (Returns source domain information)
       [-dest]                  (Returns destination signal information)
       [-dest_domain]          (Returns destination domain information)
       [-boundary]              (Returns the domain boundary)
       [-device]                (Returns device information)
       [-strategy]              (Returns strategy information)
       [-type type]             (Specifies device or strategy type:
                                Values: iso, ls)
       [-quiet]                 (Suppresses all messages. Syntax error messages are not
suppressed)
       crossover                (Specifies crossover objects)
```

#### Use Model

```
#Get crossover object
set xover1 [sort_collection [get_crossovers -from_signal top.i_core.u4.Z] full_name]
{"i_core.u4.Z i_core.Z1 out1"}

#<crossover> is required
get_crossover_info -src_domain
Error: Required argument 'crossover' was not found (CMD-007)

#<crossover> needs to be a crossover collection/object. It cannot be a name string or
other type collections.
get_crossover_info top.i_core.u4.*
[Error] NEED_UPF_COL: Collection is needed
        Please use get_crossovers to get collection.
get_crossover_info [get_pin i_core.A1]
[Error] BAD_UPF_COL: Invalid collection type
        Need collection type 'crossover'.

#By default, it returns all crossover nodes from the given crossover object. It is the
same as "get_crossover_node <crossover>" command.
get_crossover_info $xover1
```

```

{"i_core.u4.Z", "i_core.Z1", "out1"}

#-type is needed for "-strategy" or "-device"
get_crossover_info $xover1 -strategy
[Error] COM_CMD008: Option group: ( -device -strategy ) depends on option group: ( -type )
)
get_crossover_info $xover1 -device
[Error] COM_CMD008: Option group: ( -device -strategy ) depends on option group: ( -type
)

#"strategy" or "-device" is needed for "-type"
get_crossover_info $xover1 -type iso
[Error] COM_CMD008: Option group: ( -type ) depends on option group: ( -device -strategy
)

#Get isolation device/strategy of the crossover.
#If there are multiple devices/strategies, all of them will be returned.
get_crossover_info $a -device -type iso
{"h1/isol"}
get_crossover_info $a -strategy -type iso
{"PD_P1/ISO1"}

#get src/dest node
get_crossover_info $xover1 -src
{"i_core.u4.Z"}
get_crossover_info $xover1 -dest
{"out1"}

#get src/dest domain
get_crossover_info $xover1 -src_domain
{"PD_CORE"}
get_crossover_info $xover1 -dest_domain
 {"TOP"}

```

#### 4.5.2.6.10 **get\_level\_shifter\_strategies**

Gets a collection of level shifter strategies which matches the specification.

##### Syntax

```

vc_static_shell> get_level_shifter_strategies -help
Usage: get_level_shifter_strategies      # Returns a collection of specific level shifter
strategies
      [-regexp]                  (Patterns are regular expressions)
      [-exact]                   (Specifies exactly matching)
      [-nocase]                  (Regexp matches case-insensitive)
      [-source <power_domain/supply_set>]
                                (Specifies source power domain or supply set)
      [-sink <power_domain/supply_set>]
                                (Specifies sink power domain or supply set)
      [-domain <power_domain>]
                                (Specifies power domain)
      [-input_supply <supply_set>]
                                (Specifies input supply set)
      [-output_supply <supply_set>]
                                (Specifies output supply set)

```

```

[-of_objects <power_domain/cell/pin/port>]
    (Specifies power_domain, cell, pin or port)
[-filter <expression>] (Filter collection with 'expression')
[-quiet] (Suppresses all messages. Syntax error messages are not
suppressed)
[<patterns>] (List of level shifter strategy name patterns)

```

## Use Model

To retrieve level shifter strategy that survive on the pin CORE1/in1

```
%vc_static_shell> get_level_shifter_strategies -of_objects CORE1/in1
{"ls_pd1"}
```

To retrieve level shifter strategy associated to the level shifter instance ls\_inst

```
%vc_static_shell> get_level_shifter_strategies -of_objects ls_inst
{"ls_top"}
```

To retrieve all level shifter strategies that has the rule high\_to\_low

```
%vc_static_shell> get_level_shifter_strategies -filter {rule == high_to_low}
{"ls_top, ls_pd1, ls_pd2"}
```

To retrieve all level shifter strategies that has the specified source

```
vc_static_shell> get_level_shifter_strategies -source ss1
{"PD1/lsP_2"}
```

### 4.5.2.6.11 get\_level\_shifter\_strategy\_elements

Returns a collection of the objects related to the collection of level shifter strategies.

#### Syntax

Usage: get\_level\_shifter\_strategy\_elements # Returns a collection different aspects of this LS strategy object

```

[-source] (Returns source power domain or supply set)
[-sink] (Returns sink power domain or supply set)
[-input_supply] (Returns input supply set)
[-output_supply] (Returns output supply set)
[-elements] (Returns the design elements on which this strategy is
applied)
[-quiet] (Suppresses all messages. Syntax error messages are not
suppressed)
<level_shifter_strategy> (Specifies level shifter strategy name or collections)
```

### 4.5.2.6.12 get\_supply\_states

Returns a collection of supply states for the specified supply net.

#### Syntax

```
%vc_static_shell> get_supply_states -help
Usage: get_supply_states # Returns a collection of supply states
[-of_objects <supply_port/supply_net/pst_state>]
    (Specifies supply port, supply net or pst state)
[-filter <expression>] (Filter collection with 'expression')
```

```

[-quiet]          (Suppresses all messages. Syntax error messages are not
suppressed)
[<net_port_state>] (Specifies supply state name patterns or collections)

```

## Use Model

To retrieve the supply states for port VSS,VTOP

```
%vc_static_shell> get_supply_states -of_objects VSS
{ "HV" }
get_supply_states -of_objects VTOP
{ "HV", "OFF" }
```

To retrieve the supply states pst states

```
%vc_static_shell> get_supply_states -of_objects subblock_inst1/sub_pst/all_on
{ "HV", "HV" }
```

To retrieve the supply states for voltage values

```
%vc_static_shell> get_supply_states -filter { voltage == 1.1 }
{ "HV", "HV" }
vc_static_shell> get_supply_states -filter { voltage == OFF }
{ "VOFF" }
```

### 4.5.2.6.13 get\_supply\_rail\_order

Returns true if supply\_net1 is strictly more ON than supply\_net2, else returns false. Returns all possible combinations of supply sets of these pair of supply nets. The command `get_supply_net_combinations` is based on the original PSTs defined in the .upf files. It means, it reports the pairs in one PST only.

It is not based on the merged PST. If two supply nets are defined in different PSTs in the .upf files, no result is returned.

#### Syntax

```
%vc_static_shell> get_supply_rail_order -help
Usage: get_supply_rail_order      # Returns true if there is a state where supply_net1 is
ON and supply_net2 is OFF
      [-state]          (Return a state name causing the condition, if any)
      <supply_net1>    (Specifies supply net name patterns or collections)
      <supply_net2>    (Specifies supply net name patterns or collections)
```

### 4.5.2.6.14 analyze\_power\_state\_exists

This command returns true/false if a particular combination of supply port states is legal in the system power state table. It enables a power architect to ensure that the key high level system power states are not deleted due to a design or tool bug.

This command can be run any time after `read_upf`. However, there are many types of power state table inconsistencies and errors that are checked by `check_lp -stage upf -pst -consistency`. In particular, invalid combinations of port states on a supply can result in violations such as `PST_VOLTAGE_DROPPED` or `PST_VOLTAGE_UNUSED`. Severe problems can result in `PST_STATE_ZERO`.

This command is not intended to duplicate all these existing checks. So, it is recommended that you run `check_lp -stage upf -pst -consistency` and fix any resulting violations before running the `analyze_power_state_exists` command.

## Syntax

```
vc_static_shell> analyze_power_state_exists -help
Usage: analyze_power_state_exists      # Check if legal state exists in merged PST
       [-quiet]                      (Suppresses all messages)
       <supply=port_state or supply or !supply> ...
   (Specify the list of supply and state pair, or supply
   expression. Here, supply needs to be fully qualified
   path name of a supply name. And port_state is legal
   port-state for that supply, considering the supply
   connection in UPF
   supply=port_state: Specify the supply and port_state
   pair to check.
   supply: Specify the supply to query any ON state.
   !supply: Specify the supply to query any OFF state.)
❖ supply: Specify the fully qualified path name of a supply name.
❖ port_state: Specify the legal port-state for that supply, considering the supply connection in UPF.
❖ supply=port_state: Specify the supply and legal port-state pair to check.
❖ supply: Specify the supply to query its ON state.
❖ !supply: Specify the supply to query its OFF state.
```

## Examples

```
analyze_power_state_exists SS1 !SS2
analyze_power_state_exists SS1 !SS2 SS3
```

When multiple power state matches the supply condition, a list of power state in collection is returned. If there is no match, a zero size collection to be returned.

Examples of multiple power state match:

- ❖ Multiple match in one power state table.

```
PST VCC VDD VDD2
S1 ON ON ON
S2 ON ON OFF
S3 ON OFF OFF
vc_static_shell>analyze_power_state_exists VCC !VDD2
{"PST/S2" "PST/S3"}
```

- ❖ Multiple match from more than one power state table

```
PST1 VCC VDD
S1 ON ON
S2 ON OFF

PST2 VDD VDD2
S1 ON ON
S2 ON OFF
S3 OFF OFF

vc_static_shell>analyze_power_state_exists VCC !VDD2
{"PST1/S1" "PST2/S2" "PST1/S2" "PST2/S3"}
```

## Error Messages

- ❖ If any field of the command is not of the form a=b then a syntax error will be issued for each incorrect field of the command.
- ❖ If any objects specified in the command do not exist, then an error will be issued for each object that does not exist.
- ❖ If a port state exists, but not for the supply where it is used, then an error is issued for each such port state. For example, if supply A is the only supply for which port state B is defined, and the field C=B is given, then the port state does not exist for that supply.
- ❖ If any errors are issued as a result of these checks, then the command returns false without performing any power state table check.

## Use Model

The top level of the design has three UPF supply ports, VA, VB, VC and corresponding supply nets. A design instance u has three supply ports VD, VE, VF and corresponding supply nets. The UPF connects VA to u/VD, VB to u/VE, and VC to u/VF. The following three power state tables are defined. The corresponding add\_port\_state commands are not shown to save space.

```
create_pst t1 -supplies {VA VB}
add_pst_state s11 -pst t1 -state {HV HV}
add_pst_state s12 -pst t1 -state {HV OFF}
add_pst_state s13 -pst t1 -state {OFF HV}
create_pst t2 -supplies {VB VC}
add_pst_state s21 -pst t2 -state {HV HV}
add_pst_state s22 -pst t2 -state {OFF OFF}
set_scope u
create_pst t3 -supplies {VD VE VF}
add_pst_state s31 -pst t3 -state {HV HV HV}
add_pst_state s32 -pst t3 -state {HV OFF HV}
add_pst_state s33 -pst t3 -state {OFF HV HV}
```

State s32 is invalid because t3 does not contain a state with VB off and VC on. This is shown in the output of report\_pst -only\_invalid.

The following transcript shows the result of several executions of the new command in VC LP.

```
%vc_static_shell> analyze_power_state_exists VA=Error: port state not specified for supply_net VA 0
%vc_static_shell> analyze_power_state_exists VA=HV foo=HV Warning: No supply_net objects matched 'foo' (SEL-004) 0
%vc_static_shell> analyze_power_state_exists VA=HV VB=foo Warning: No net_port_state objects matched 'foo' (SEL-004) 0
%vc_static_shell> analyze_power_state_exists VA=HV VB=HV1
%vc_static_shell> analyze_power_state_exists u/VD=HV VB=HV1
%vc_static_shell> analyze_power_state_exists VB=OFF VC=HV0
%vc_static_shell> analyze_power_state_exists u/VD=HV u/VE=OFF u/VF=HV0
```

## 4.5.2.7 Supply Set Related Commands

### 4.5.2.7.1 get\_supply\_sets

Returns a collection of supply set objects.

#### Syntax

```
%vc_static_shell> get_supply_sets -help
Usage: get_supply_sets      # Returns a collection of specific supply sets
```

```

[-regexp]          (Patterns are regular expressions)
[-nocase]         (Regexp matches case-insensitive)
[-of_objects <power_domain/supply_net>]
                  (Specifies power_domain or supply net)
[-quiet]          (Suppresses all messages. Syntax error messages are not
suppressed)
[<patterns>]      (List of supply set name patterns)

```

## Examples

To retrieve the supply set associated with net VDD

```
%vc_static_shell> get_supply_sets -of_objects VDD
{"TOP.primary"}
```

To retrieve the supply sets of domain PD\_CORE\_1

```
%vc_static_shell> get_supply_sets -of_objects PD_CORE_1
{"CORE1_ss", "PD_CORE_1.default_retention", "PD_CORE_1.default_isolation"}
```

Matches the pattern with case insensitive

```
%vc_static_shell> get_supply_sets core1_SS -nocase
{"CORE1_ss"}
```

### 4.5.2.7.2 get\_supply\_nets

Returns a collection of UPF supply nets that match the specified criteria.

#### Syntax

```

%vc_static_shell> get_supply_nets -help
Usage: get_supply_nets      # Returns a collection of specific supply nets
      [-regexp]          (Patterns are regular expressions)
      [-nocase]           (Regexp matches case-insensitive)
      [-of_objects
<power_domain/supply_set/supply_port/state_table/pst_state/net_port_state>]
                  (Specifies power_domain, supply_set, supply_port,
state_table, pst_state or net_port_state)
      [-filter <expression>] (Filter collection with 'expression')
      [-quiet]            (Suppresses all messages. Syntax error messages are not
suppressed)
      [-root]             (Returns root supply nets)
      [-include_unused]   (Includes supply nets unused in any state table)
      [<patterns>]        (List of supply net name patterns)

```

#### Use Model

To retrieve the supply net associated with the supply set CORE1\_ss

```
%vc_static_shell> get_supply_nets -of_objects CORE1_ss
{"VSS", "VDD_CORE_1"}
```

To retrieve the supply net connected with the port VDD\_CORE\_1

```
%vc_static_shell> get_supply_nets -of_objects VDD_CORE_1
{"VDD_CORE_1"}
```

To retrieve the supply net whose supply state is HV

```
%vc_static_shell> get_supply_nets -of_objects HV
{ "VDD_CORE_1", "VDD_CORE_2", "VDD_SW_BUF" }
```

To retrieve the supply nets of PST pst\_1

```
%vc_static_shell> get_supply_nets -of_objects pst_1
{ "VSS", "VDD_TOP", "VDD_CORE_1", "VDD_SW_BUF" }
```

To retrieve the supply nets used in PST

```
%vc_static_shell> get_supply_nets -filter { in_pst == true }
{ "VDD_CORE_1", "VDD_CORE_2", "VDD_SW_BUF", "VDD_SW_INV" }
```

To retrieve supply power nets

```
%vc_static_shell> get_supply_nets -filter { supply_type == power }
{ "VDD_CORE_1", "VDD_CORE_2", "VDD_SW_BUF" }
```

#### 4.5.2.7.3 **get\_connected\_supply\_net**

Returns the supply nets from which a path can be traced to the logic pin given in the command, without having to go through any gates; hierarchy boundaries may be crossed while tracing.

##### Syntax

```
%vc_static_shell> get_connected_supply_net -help
Usage: get_connected_supply_net      # Returns a collection of the connected supply net of
given data pin
      [-quiet]                  (Suppresses all messages. Syntax error messages are not
suppressed)
      <pin>                      (List of design pin names or collections)
```

##### Use Model

```
VDD_TOP -----> dff_1/D
%vc_static_shell> get_connected_supply_net dff_1/D
{ "VDD_TOP" }

VDD_TOP -----> i_core_1/D-----> i_core_1/dff_1/D
%vc_static_shell> get_connected_supply_net i_core_1/dff_1/D
{ "VDD_TOP" }

VSS ----> BUF -----> dff_2/D
%vc_static_shell> get_connected_supply_net dff_2/D
Warning: No supply_net objects matched 'dff_2/D' (SEL-004)
Error: Nothing matched for collection (SEL-005)
```

As BUF is percent between supply net and Logical pin no supply\_net connected match for logical pin dff\_2/D

To retrieve a supply net in a design where the supply net is directly connected to the logical pin

```
%vc_static_shell> get_connected_supply_net dff_1/D
{ "VDD_TOP" }
```

If the supply net is connected to the logical pin through the domain boundary

```
%vc_static_shell> get_connected_supply_net i_core_1/dff_1/D
{ "VDD_TOP" }
```

If the supply net is connected to the logical pin through BUF/INV

```
%vc_static_shell> get_connected_supply_net dff_2/D
Warning: No supply_net objects matched 'dff_2/D' (SEL-004)
Error: Nothing matched for collection (SEL-005)
```

#### 4.5.2.7.4 get\_supply\_ports

Returns a collection of UPF supply ports that match the specified criteria.

##### Syntax

```
%vc_static_shell> get_supply_ports -help
Usage: get_supply_ports      # Returns a collection of specific supply ports
      [-regexp]           (Patterns are regular expressions)
      [-nocase]            (Regexp matches are case-insensitive)
      [-of_objects <supply_net>]
                           (Specifies connected supply_net)
      [-filter <expression>] (Filter collection with 'expression')
      [-quiet]              (Suppresses all messages. Syntax error messages are not
suppressed)
      [<patterns>]          (List of supply port name patterns)
```

##### Use Model

For example, to get the supply port connected with net VDD

```
%vc_static_shell> get_supply_ports -of_objects VDD { "VDD" }
```

#### 4.5.2.7.5 all\_isolation\_cells

Returns a collection of the existing isolation cells in the design. The enabled level shifters also work as isolation cells, but the type of cell is not returned by this command.

##### Syntax

```
%vc_static_shell> all_isolation_cells -help
Usage: all_isolation_cells      # Returns a collection of all isolation cells
```

#### 4.5.2.7.6 all\_level\_shifters -type

Returns a collection of existing level shifter cells in the design.

If you specify -type else, the command returns only the enabled level-shifter cells.

If you specify -type simple, the command returns only regular level-shifter cells.

If the -type option is not specified, the command returns all level-shifter cells.

##### Syntax

```
%vc_static_shell> all_level_shifters -help
Usage: all_level_shifters      # Returns a collection of specific level shifter cells
      [-type type]           (Specifies the level shifter type:
                                Values: els, simple, dls)
```

#### 4.5.2.7.7 all\_macro\_cells

Returns a collection of macro cells in the design.

##### Syntax

```
%vc_static_shell> all_macro_cells -help
Usage: all_macro_cells      # Returns a collection of all macro cells
```

#### 4.5.2.7.8 get\_crossover\_nodes

Returns a collection of crossover\_nodes for the given objects.

##### Syntax

```
%vc_static_shell> get_crossover_nodes -help
Usage: get_crossover_nodes      # Returns a collection of xover nodes created for the
given name
      [-filter <expression>] (Filters in those objects whose attribute matches with
<expression>)
      [-quiet]                  (Suppresses all messages. Syntax error messages are not
suppressed)
      <obj>                     (Gives a name or an object (pin/port/net or crossover) to
create the crossover node)
```

A name string, a collection of design pins, ports or nets, or a collection of type <crossover> returned by command get\_crossovers.

#### 4.5.2.7.9 get\_root\_supply\_object

The get\_root\_supply\_object Tcl command is introduced.

##### Syntax

```
get_root_supply_object [<pin>]
[-power]
[-ground]
[-upf]
[-netlist]
[- <nocase>]
[-quiet]
```

This command retrieves a collection of root supply objects for specific design pins/ports or UPF supply ports. This command behaves same as the get\_root\_supply\_net command except for the following specific case:

For a db cell internal PG pin without a pg connection, the get\_root\_supply\_object command returns the internal pg pin as it is the root supply object. However, the get\_root\_supply\_net command returns a warning as no root supply net is recognized.

#### 4.5.2.8 Reporting Commands

##### 4.5.2.8.1 report\_crossover

Reports details of this crossover. By default, the following items are printed:

- ❖ Source domain
- ❖ Destination domain
- ❖ List of instances and nets in the crossover with protection devices annotated.

If any option is specified, only those specific items are printed.

## Syntax

```
%vc_static_shell> report_crossover -help
Usage: report_crossover      # Reports specific crossovers
       [-source]           (Reports source power domain)
       [-dest]              (Reports destination power domain)
       [-domain_boundary]  (Reports the domain boundary)
       [-device]            (Reports the device)
       [-instances]         (Reports the instances)
       [-strategy]          (Reports the strategy)
       [-type device_type] (Specifies device type:
                           Values: iso, ls)
       [<crossover>]        (Specifies crossover collections)
```

### 4.5.2.8.2 report\_level\_shifter\_strategy

Reports the details of the level shifter strategy. By default, the following items are printed:

- ❖ Name
- ❖ Domain
- ❖ Source domain
- ❖ Sink domain
- ❖ Level shifter rule
- ❖ Location
- ❖ Threshold (if any)
- ❖ Input supply
- ❖ Output supply
- ❖ Instances
- ❖ Elements
- ❖ If no\_shift is specified
- ❖ If force\_shift is specified
- ❖ applies\_to
- ❖ Transitive

If the strategy is defined with -no\_shift, removes two fields: location and rule, and keep other fields. Put -no\_shift within bracket after strategy name.

If specific options are specified, then print only those values.

## Syntax

```
Usage: report_level_shifter_strategy      # Reports specific level shifter strategies
       [-name]                (Reports level shifter strategy name)
       [-domain]               (Reports power domain)
       [-source]               (Reports source power domain or supply set)
       [-sink]                 (Reports sink power domain or supply set)
       [-rule]                 (Reports rule type)
       [-location]             (Reports location type)
       [-threshold]            (Reports threshold)
       [-input_supply]          (Reports input supply set)
```

```

[-output_supply]      (Reports output supply set)
[-elements]          (Reports elements)
[-exclude_elements] (Reports exclude elements)
[-no_shift]          (Reports if no_shift)
[-applies_to]         (Reports applies_to type)
[<level_shifter_strategy>] (List of level shifter strategy name patterns or
                           collections)

```

#### 4.5.2.8.3 report\_supply\_set

Reports details of the supply set. By defaults, it prints the name, all the functions specified for this supply set, and the reference ground.

If specific option is specified, then prints only relevant information.

##### Syntax

```

%vc_static_shell> report_supply_set -help
Usage: report_supply_set      # Reports specific supply sets
      [-power]            (Reports power net)
      [-ground]           (Reports ground net)
      [-name]              (Reports supply set name)
      [-pwell]             (Reports pwell)
      [-nwell]             (Reports nwell)
      [-deepwell]          (Reports deep pwell)
      [-deepnwell]         (Reports deep nwell)
      [-reference_ground] (Reports reference ground)
      [<supply_set>]       (List of supply set name patterns or collections)

```

#### 4.5.2.8.4 report\_mv\_library\_cells

Reports power management cells used in the design.

Reports details of the supply port. By default, the following items are printed:

- ❖ Name
- ❖ Current scope
- ❖ Direction
- ❖ Connected supply net
- ❖ Supply state names

##### Syntax

```

%vc_static_shell> report_mv_library_cells -help
Usage: report_mv_library_cells      # Reports all power management cells from the
libraries
      [-level_shifters]        (Reports all level shifters from the libraries)
      [-isolation_cells]       (Reports all isolation cells from the libraries)
      [-retention_cells]        (Reports all retention cells from the libraries)
      [-switch_cells]          (Reports all supply switch cells from the libraries)
      [-always_on_cells]        (Reports all always on cells from the libraries)
      [-cell_name <db_cell_name>]
                                (Specifies cell name)
      [-verbose]                (Reports details)

```

##### Use Model

```
%vc_static_shell> report_mv_library_cells
```

**Signal Pin Attributes:**

lse - level shifter enable pin  
 isoe - isolation cell enable pin  
 scmr - standard cell main rail

**Supply Pin Attributes:**

PRM - primary  
 BCK - backup  
 INT - internal  
 PWL - P-well  
 NWL - N-well  
 DPL - deep P-well  
 DNL - deep N-well

**Cell Attributes:**

ao - always on  
 ls - level shifter  
 iso - isolation cell  
 switch\_cg - switch  
 ret - retention

**4.5.2.8.5 report\_supply\_net**

Reports details of the supply net. By default, the following items are printed:

- ❖ Name
- ❖ Current scope
- ❖ Operating voltage
- ❖ Supply states
- ❖ Power domain

**Syntax**

```
%vc_static_shell> report_supply_net -help
Usage: report_supply_net      # Reports specific supply nets
       [<supply_net>]          (List of supply net name patterns or collections)
```

**Use Model**

```
%vc_static_shell> report_supply_net
  Supply Net           : GENPP.default_isolation.ground
  Current Scope        :
  Operating Voltage    : -- Best Case -- -- Worst Case --
                        - U -             - U -
  Supply States         : No supply states defined for the net
  Power Domain          :
-----
```

**4.5.2.8.6 The report\_properties Command**

Reports information on the properties of the design objects like cells, nets, ports and pins.

**Syntax**

```
%vc_static_shell> report_properties [-instance <instance>] [-port <port>] [-pin <pin>]
[-net <net>]
```

Where:

- ❖ [-instance <instance>]: Specify the instance name for which you want to retrieve information.
- ❖ [-port <port>]: Specify the port name for which you want to retrieve information.
- ❖ [-pin <pin>]: Specify the pin name for which you want to retrieve information.
- ❖ [-net <net>]: Specify the net name for which you want to retrieve information.

## Examples

- ❖ The following command returns information of standard cell, low power cells, module instance:  
`%vc_static_shell> report_properties -instance <instance>`

The following is the sample output for this example:

```
Basic
  Type          : instance
  Name          : ELS_TOP

Library
  Cell Name    : demo_els_10v_12v_pg/DMLSUAN10V
  Cell Type    : Isolation Level Shifter
  LS Type      : LowHigh
  SCMR         : VDD
  Iso Enable   : B
  PG Pin       : VDD
  PG Type      : Primary Power
  Related Power Supply Net : top/VDD_TOP
  Power Net Resolution Method : FROM_UPF_POWER_DOMAIN
  PG Pin       : VDDH
  PG Type      : Primary Power
  Related Power Supply Net : top/VDD_TOP
  Power Net Resolution Method : FROM_UPF_POLICY
  PG Pin       : VSS
  PG Type      : Primary Ground
  Related Ground Supply Net : top/VSS
  Ground Net Resolution Method : FROM_UPF_CSN

UPF
  Domain Name  : TOP
  Policy        : INT_ISO_CORE_SINK_B
  Power Net     : VDD_TOP
  Ground Net    : VSS
  Clamp Value   : Zero
  Iso Enable Signal : iso_en
  Iso Sense     : LOW
```

- ❖ The following command returns different properties like power/ground information, isolation/level shifter strategy applied on the design port.

`%vc_static_shell> report_properties -port i_core_1/Q`

The following is the sample output for this example:

```
Basic
  Type          : port
  Name          : i_core_1/Q
  Direction     : output

UPF
  From Domain   : PD_CORE_1
  To Domain     : TOP
  Related Power Supply Net : top/VDD_CORE_1
  Power Net Resolution Method : FROM_UPF_POWER_DOMAIN
```

```

Related Ground Supply Net    : top/VSS
Ground Net Resolution Method : FROM_UPF_POWER_DOMAIN
Isolation Policy            : INT_ISO_CORE_SINK_B
Is Heterogenous Fanout      : true

```

- ❖ The following command returns different properties like direction, power/ground information of the design pin.

```
%vc_static_shell> report_properties -pin ISO_TOP/B
```

The following is the sample output for this example:

```

Basic
Type          : pin
Name          : ISO_TOP/B
Direction     : input

UPF
Related Power Pin   : VDD
Related Ground Pin : VSS
Related Power Supply Net : top/VDD_SW_ISO
Power Net Resolution Method : FROM_UPF_CSN
Related Ground Supply Net : top/VSS
Ground Net Resolution Method : FROM_UPF_CSN
Power Domain       : TOP
Is Heterogenous Fanout : false

```

- ❖ The following command returns the direction, power/ground information of the design net.

```
%vc_static_shell> report_properties -net wire1
```

The following is the sample output for this example:

```

Basic
Type          : net
Name          : wire1

UPF
Power Domain : TOP

```

### 4.5.3 Support to Query SPA/SDA Attributes

You can query attributes used in the `set_port_attributes` (SPA) / `set_design_attributes` (SDA) UPF commands. The UPF scripting systems rely on both LRM defined and user-defined attribute names and values for SPA and SDA.

#### 4.5.3.1 Querying Predefined Attributes

You can query the following attributes on instances:

- ❖ Boolean: `enable_bias`, `macro_as_domain_boundary`, `terminal_boundary`, `lower_domain_boundary`, `is_hard_macro`, `is_soft_macro`
- ❖ String: `shared_voltage_area`, `disjoint_supply_nets`, `contains_switches`

You can query the following attributes on models:

- ❖ Boolean: `macro_as_domain_boundary`, `upf_reconcile_boundary`

Although these are placed on models in the UPF, they are stored on all the instances of that model in the database. That is, there is no capability to query models with the attribute; instead, query instances.

#### 4.5.3.2 Support for User Defined Attributes

Use the `define_upf_attribute` Tcl command (similar to the existing command `define_user_attribute`) to query user-defined attributes from the Tcl scripts. The `define_upf_attribute` command must be used before `read_upf`.

##### Syntax

```
define_upf_attribute -class <cell|design|pin|port> <name>
```

#### 4.5.3.3 Examples

- ❖ Consider the following UPF:

```
set_design_attributes -elements u1 -attribute {enable_bias true}
```

Execute the following query to see the result:

```
vc_static_shell> set s [get_attributes [get_cells u1] enable_bias]
true
```

- ❖ Consider the following UPF:

```
set_design_attributes -elements u1 -attribute {my_attribute my_value}
```

And this line in the Tcl script:

```
define_upf_attribute -class cell my_attribute
```

Execute the following query to see the result:

```
vc_static_shell> set s [get_attributes [get_cells u1] my_attribute] my_value
```

- ❖ Use the `get_attribute [current_design] <attribute_name>` option to query instance attributes as design attributes at the top level.

Example

```
vc_static_shell> get_attribute [current_design] enable_bias
[Info] COM_OPT035: Current design is 'top'.
true
```

#### 4.5.3.4 Limitations

There are several UPF attributes, for example `enable_bias`, which have this inheritance behavior. If `u1/u2/u3` does not define any value for the attribute, it should inherit the value from its nearest ancestor. VC LP does not support any hierarchical inheritance.

### 4.6 Debugging Using GUI

VC Static and Formal platform (VC Formal and VC LP) unifies the debug environment to be natively Verdi-based - that is, the unified GUI and debug capabilities are now based on the Verdi debug platform. By default, VC LP uses Verdi for debugging. With the unified support for Verdi-based debug, designers and verification engineers can access the combined power of formal-specific and static-specific debug features and use Verdi's de facto industry-standard workflow, interface and powerful debug capabilities.

The unified Verdi-based debug platform in the VC Static and Formal platform offers automation features that dramatically reduce the debug time. These integration features include the following:

- ❖ Easy-to-understand schematic views with check-specific annotations, coloring, and partitioning
- ❖ Powerful waveform viewers and analyzers
- ❖ Advanced cross-probing capabilities to allow fast and easy ability to switch between views
- ❖ Search and trace capabilities to enable rapid ability to identify the root cause of violations

- ❖ Highly-differentiated and high-value advanced debug features, such as two-design sequential equivalence checking debug with temporal flow analysis

In addition to the new natively-integrated debug capabilities, VC Static platform offers many improvements to core features, performance, capacity, and stability.

By default, the `enable_verdi_debug` application variable is set to true. When this application variable is set to true, VC Static uses Verdi for debug and Verdi uses the KDB generated by VCS to import the design information for debugging. This reduces the complexity of dual compilation (Verdi Vericom/vhdlcom and VC Static Simon), and improves the overall performance (CPU Time/Memory).

The following is the sample output after debug in Verdi is completed.

Command Line Message Output:

```
-----  
Info: Simon call complete  
Info: Exiting after Simon Analysis  
Info: Simon VCS Finished  
Top Level Modules:  
top  
TimeScale is 1 ns / 1 ns  
Verdi KDB elaboration done and the database successfully generated: 0 error(s), 0 warning(s)
```

The `enable_verdi_debug` variable must be set before any `analyze`, `elaborate`, `read_file` commands.

- ❖ In the RTL flow, when the `enable_verdi_debug` application variable is set to true, VCS is used to generate the elaborated KDB.
- ❖ In the netlist flow, by default, the KDB generation and compilation happens in parallel, thus improving the Verdi design import performance. If you set the following application variable to false, parallel KDB generation and compilation is disabled and performance of the design import might be affected.

```
%vc_static_shell> set_app_var enable_verdi_netlist_kdb false
```

To open the VC Static Native GUI, set the `enable_verdi_debug` application variable to false.

```
%vc_static_shell> set_app_var enable_verdi_debug false
```

When the `lp_strategy_association_json_dump` application variable is enabled, VC LP saves the `strategy_association_*.json` files at the end of the `check_lp` command, if the `enable_verdi_debug` application variable is also set to true.

By fault, the `lp_strategy_association_json_dump` application variable is set to true.

## 4.6.1 Handling Memory in Netlist flow

In the netlist KDB flow, both the VNR parser and VCS compile process tasks run on the same machine, this results in memory limitations for big designs. You can use the `busb` command to submit VCS compile job to a farm machine, while the VNR parser runs on the machine of the master process.

### 4.6.1.1 Use model

1. Set the following application variable in the Tcl file to enable parallel compile:

```
%vc_static_shell>set_app_var enable_verdi_netlist_kdb true  
%vc_static_shell>set_app_var enable_verdi_debug true
```

2. Add the `busb` cmd in the config file, for example, in file `temp.config`, add:

- ```
bsub -q bnormal -R "rusage[mem=1000]"
```
3. Add `-remoteConfig <fileName>` in the VC Static startup options, for example:  
`$VC_STATIC_HOME/bin/vc_static_shell -f vcst.tcl -remoteConfig temp.config`
  4. The `bsub` submit information is printed after the execution of the `read_file -netlist` command.  
You can get the job ID and submit status in this information, for example:  
*Job <337236> is submitted to queue <bnormal>.*
- After the VCS compiler job is done, the KDB is generated and you can load design in Verdi.

#### 4.6.1.2 Error handling

The `bsub` mode may fail for following reasons, and in each case an error message is reported in the shell or mail:

1. If there is no busb path found, the *command not found* error is reported.
2. If it fails for other reasons, a message indicating that a more detailed information about the task results is sent in an e-mail from the LSF system.

#### 4.6.2 Using Verdi for Debug

You can use the following capabilities of Verdi for RTL and Tcl designs using two modes: *Verdi First mode* and *VC Static first mode*.

- ❖ Verdi source and hierarchy viewer
- ❖ Verdi schematic viewer
- ❖ Verdi UPF source and hierarchy viewer
- ❖ Verdi UPF power annotated schematic viewer

##### 4.6.2.1 Verdi First Mode

In this mode, you can use the `-gui` option in the VC static shell.

```
%vc_static_shell -gui
```

[Figure 4-14](#), [Figure 4-15](#), [Figure 4-16](#), [Figure 4-17](#), and [Figure 4-18](#) show the *Verdi - VC Static Integrated View*, *Verdi Schematic View*, *Verdi Source View*, *Verdi UPF view* and the *Verdi UPF Schematic view* when you use the Verdi First Mode.

**Figure 4-14 Verdi - VC Static Integrated View**

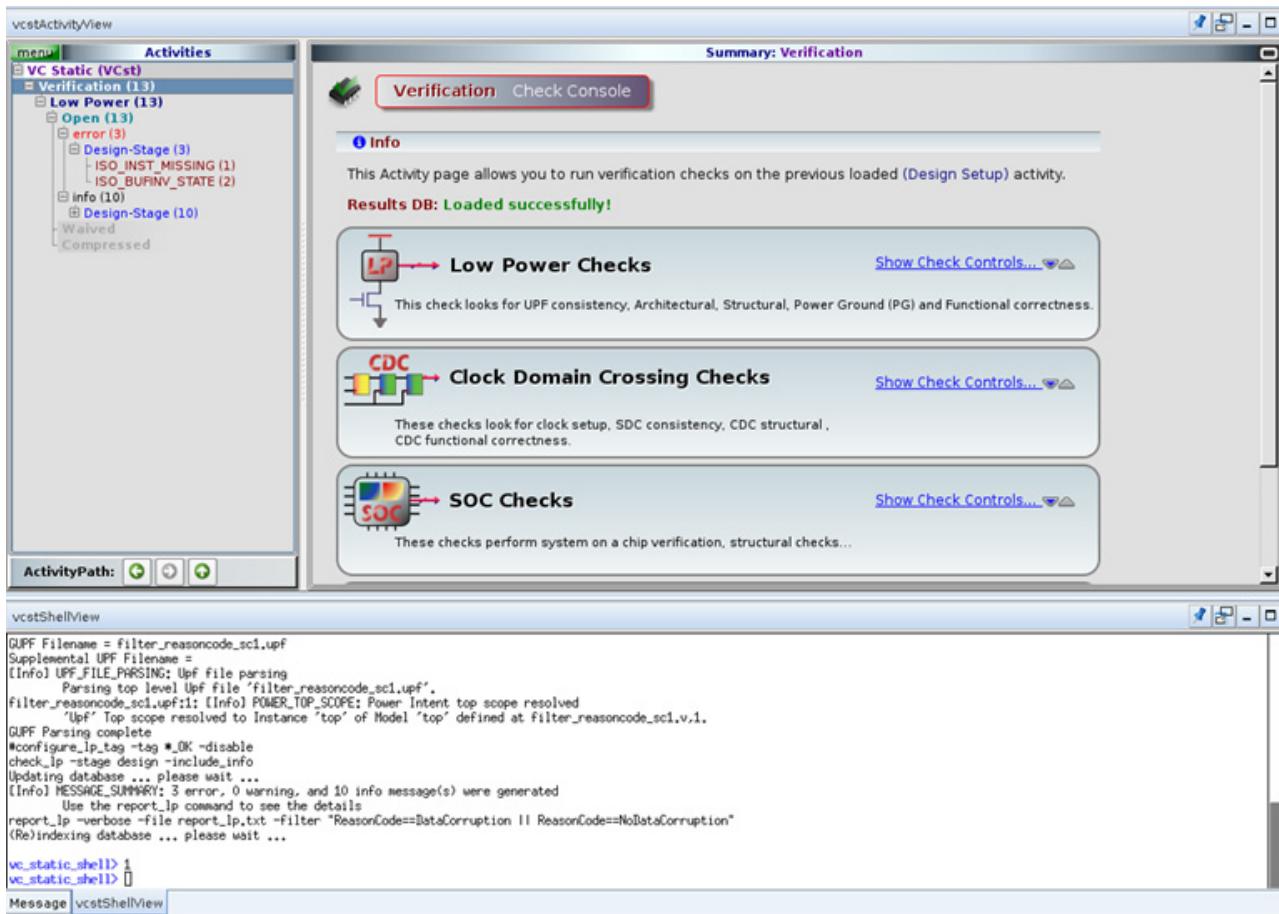


Figure 4-15 Verdi Schematic View

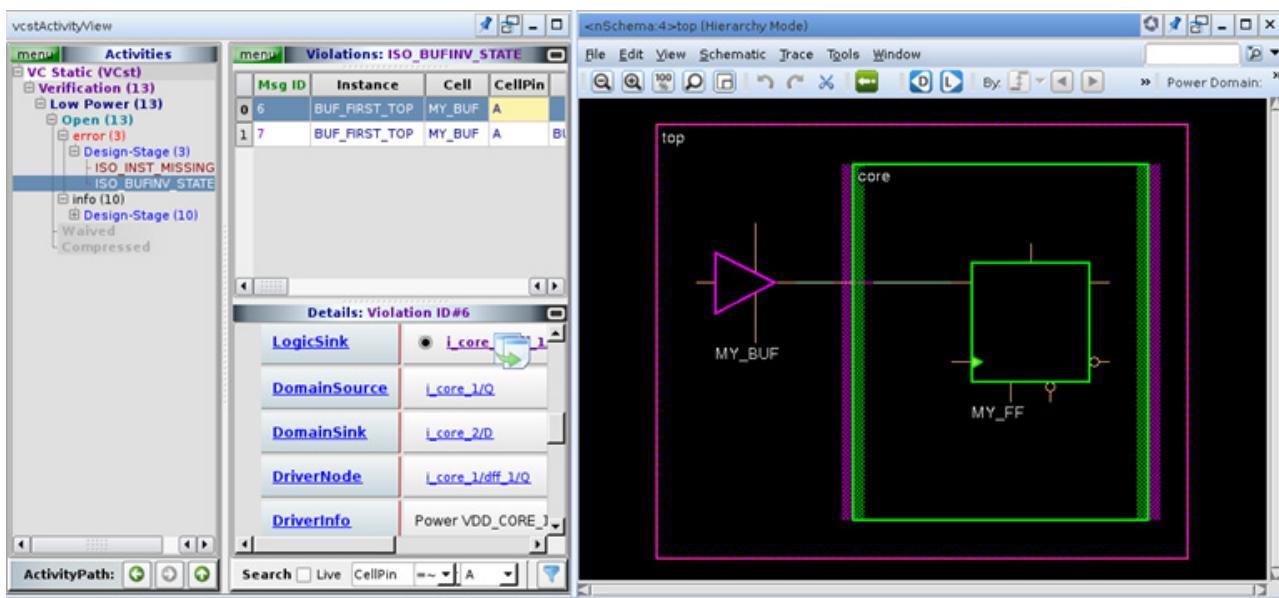
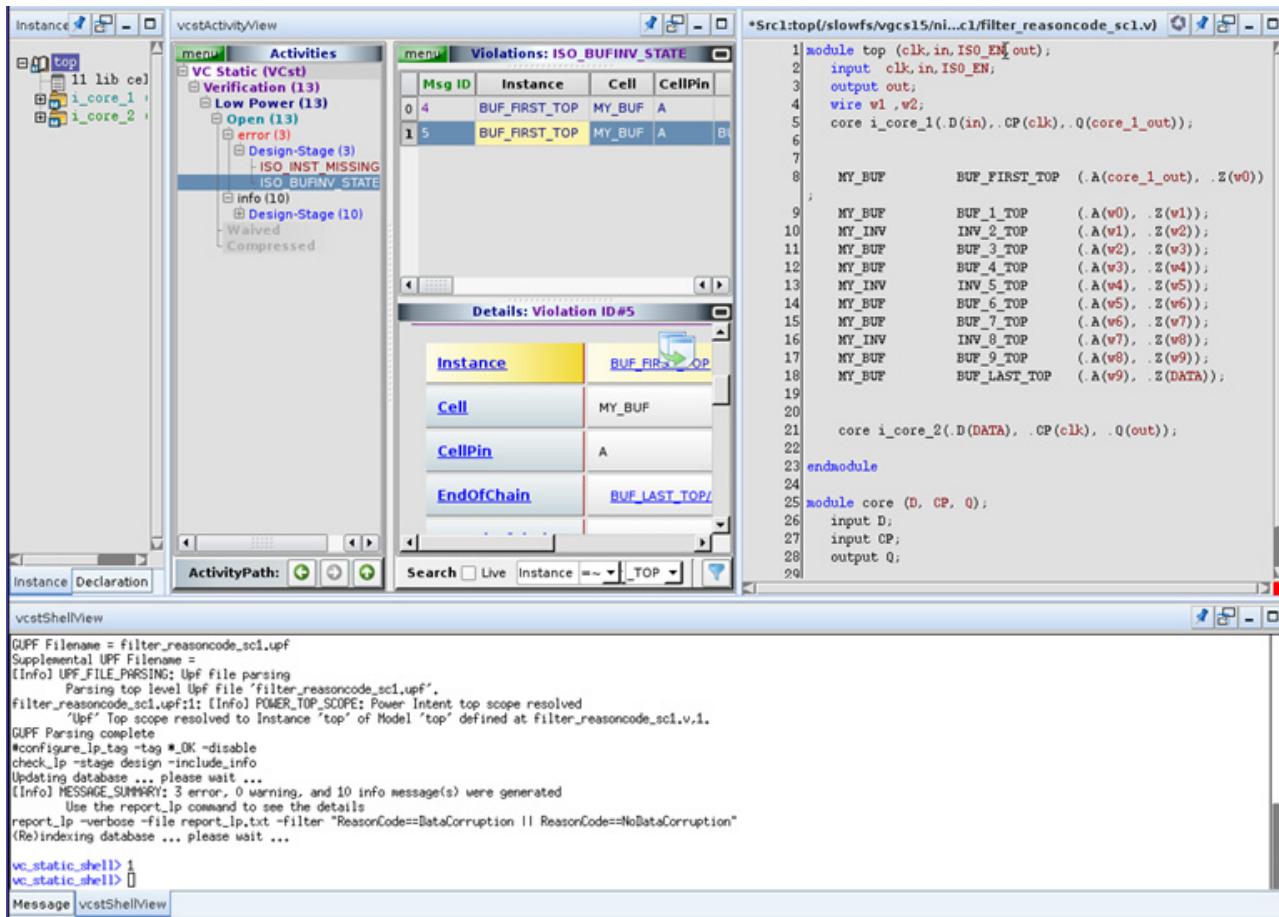
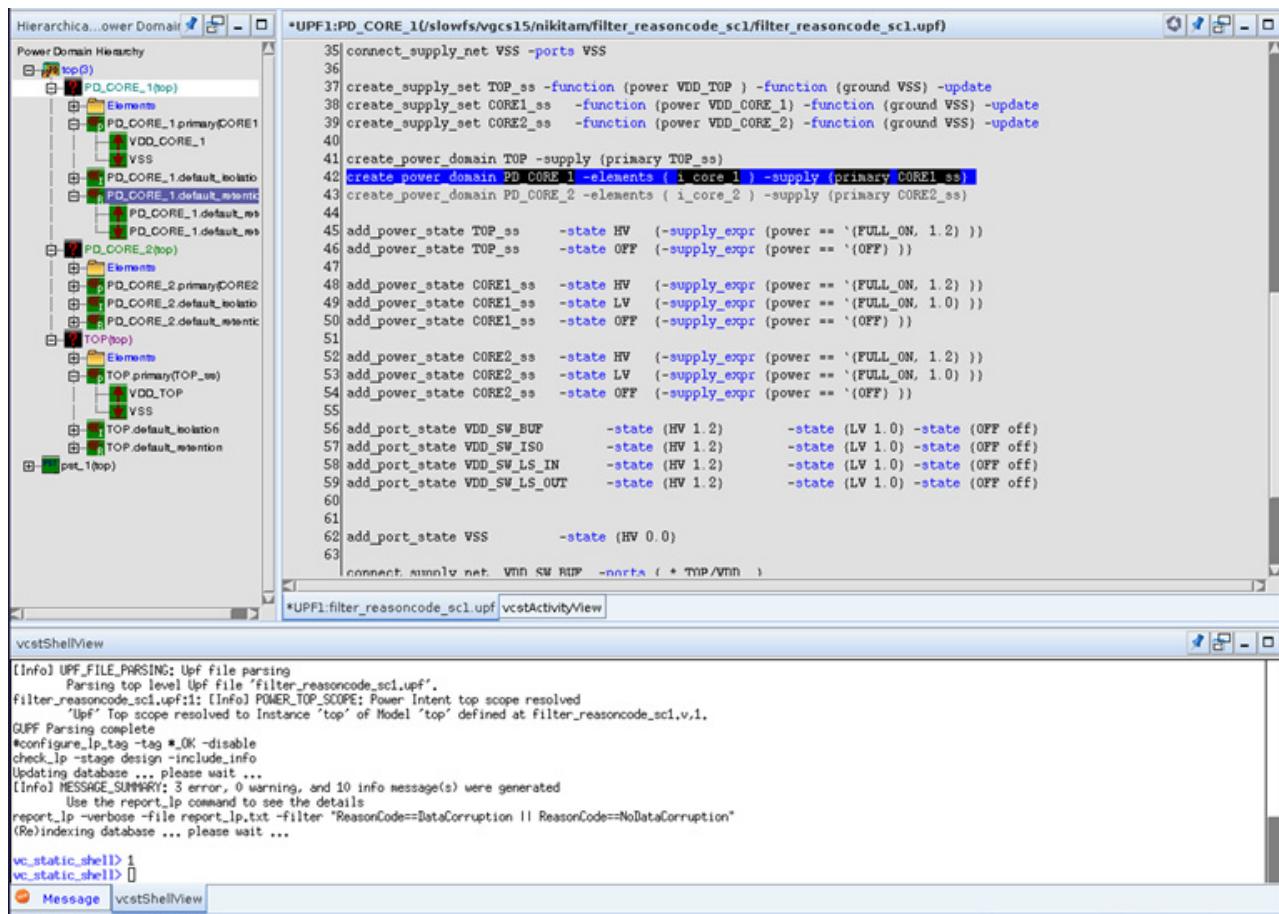
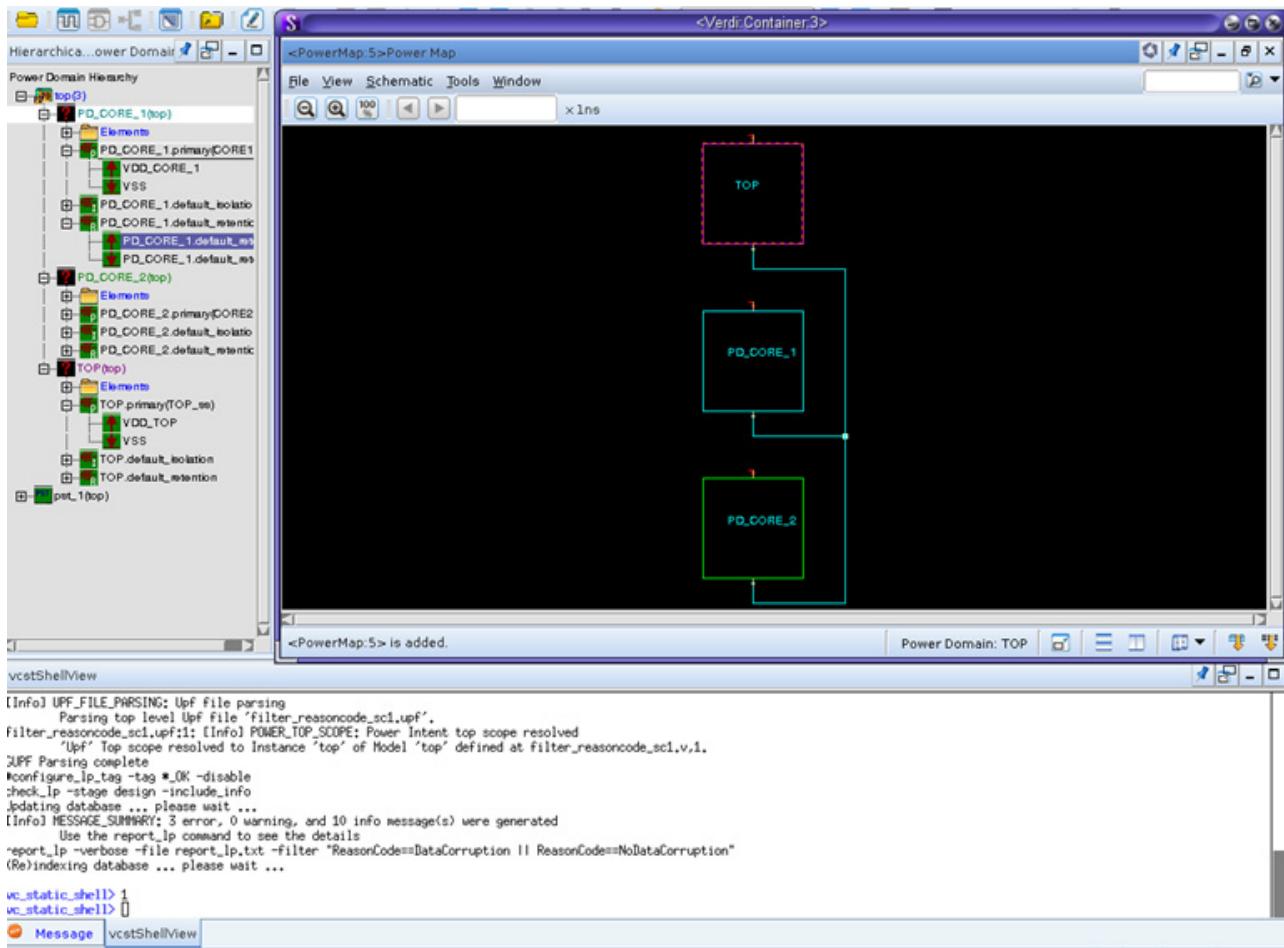


Figure 4-16 Verdi Source View



**Figure 4-17 Verdi UPF View****Figure 4-18 Verdi UPF Schematic View**

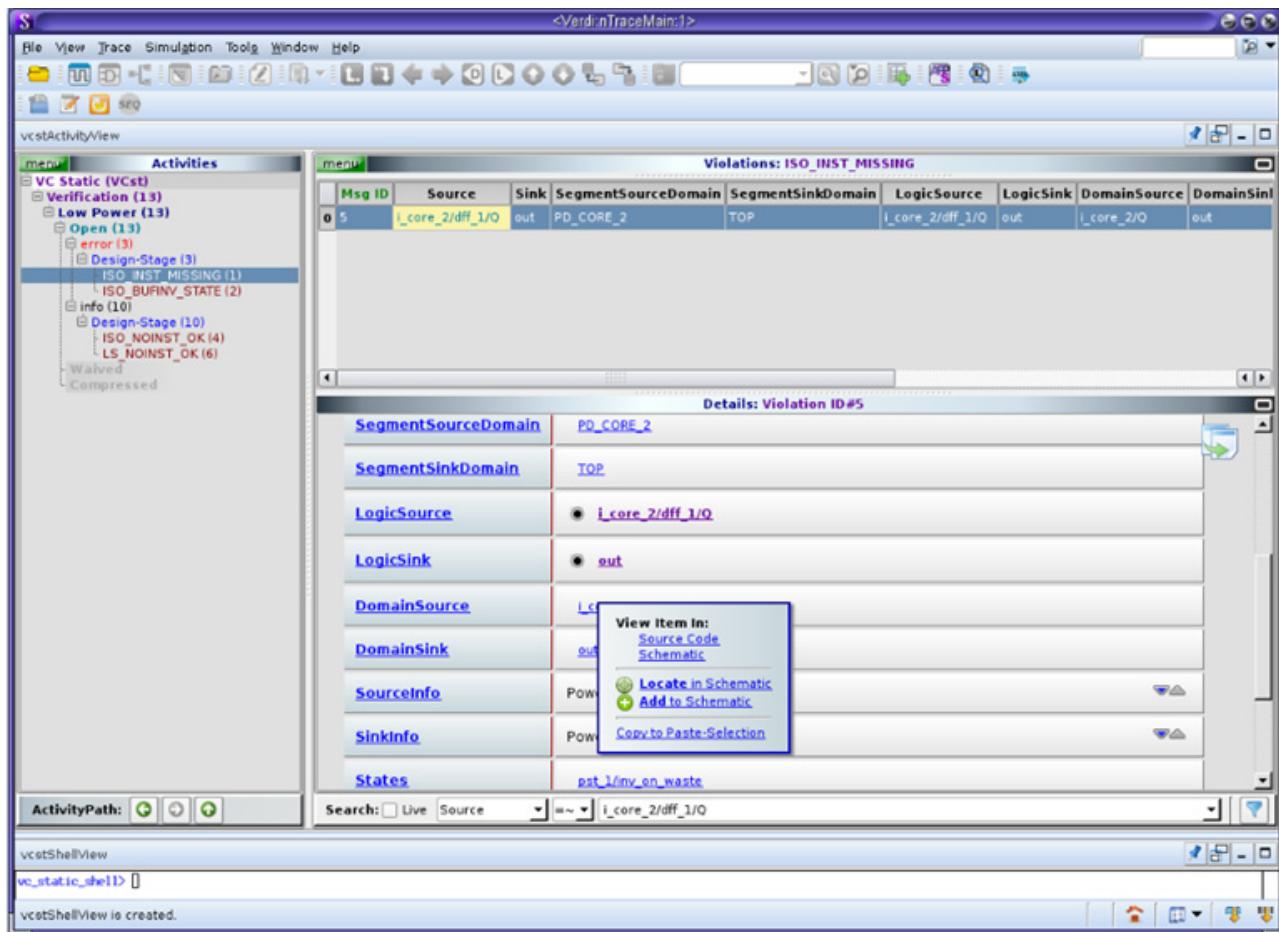


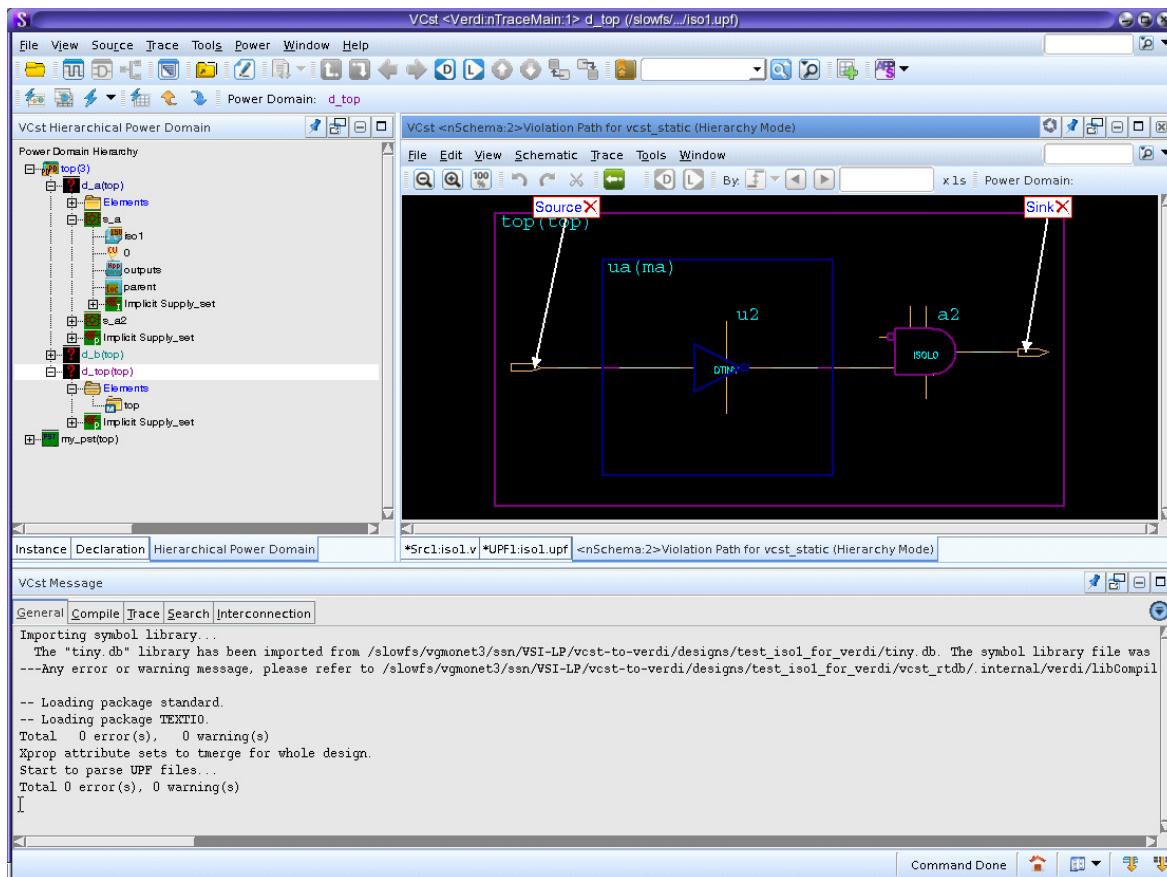
#### 4.6.2.2 VC Static First Mode

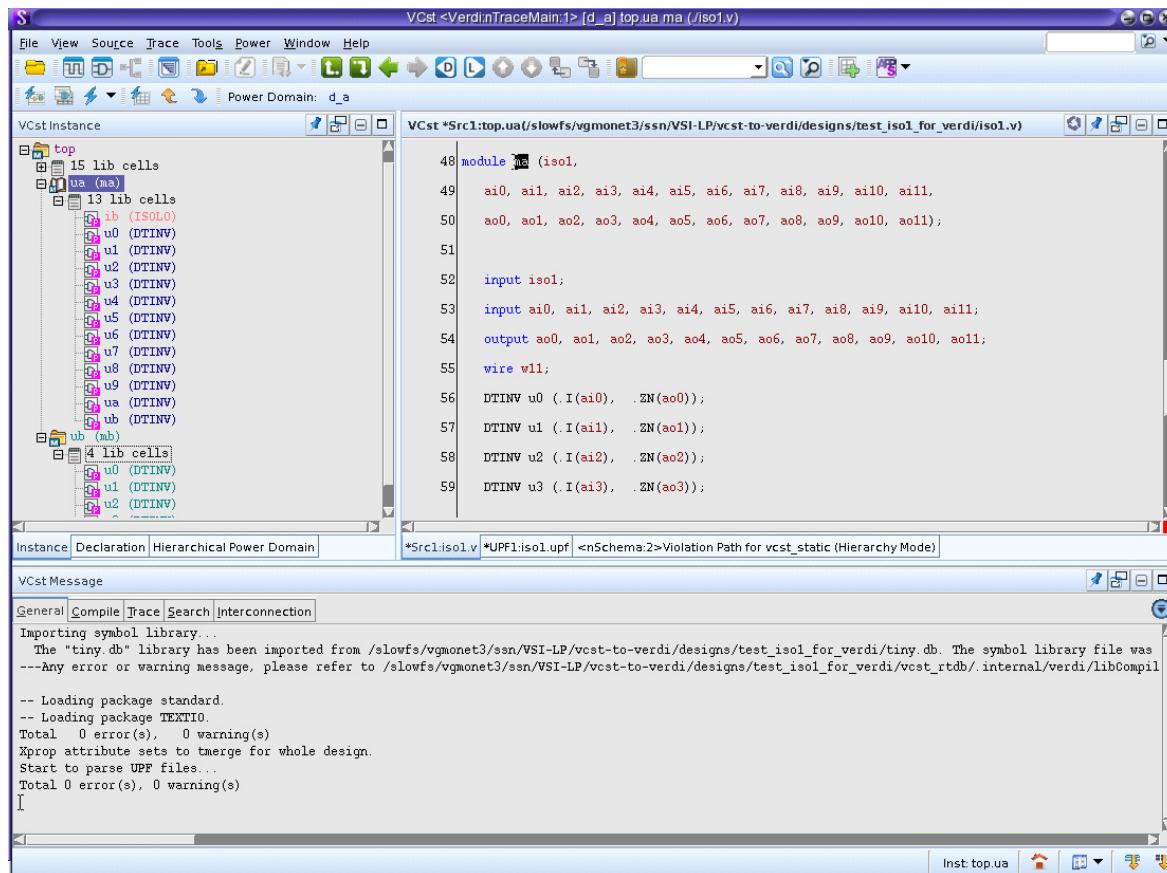
You can set the following environment variable to open Verdi debug environment in VC Static:

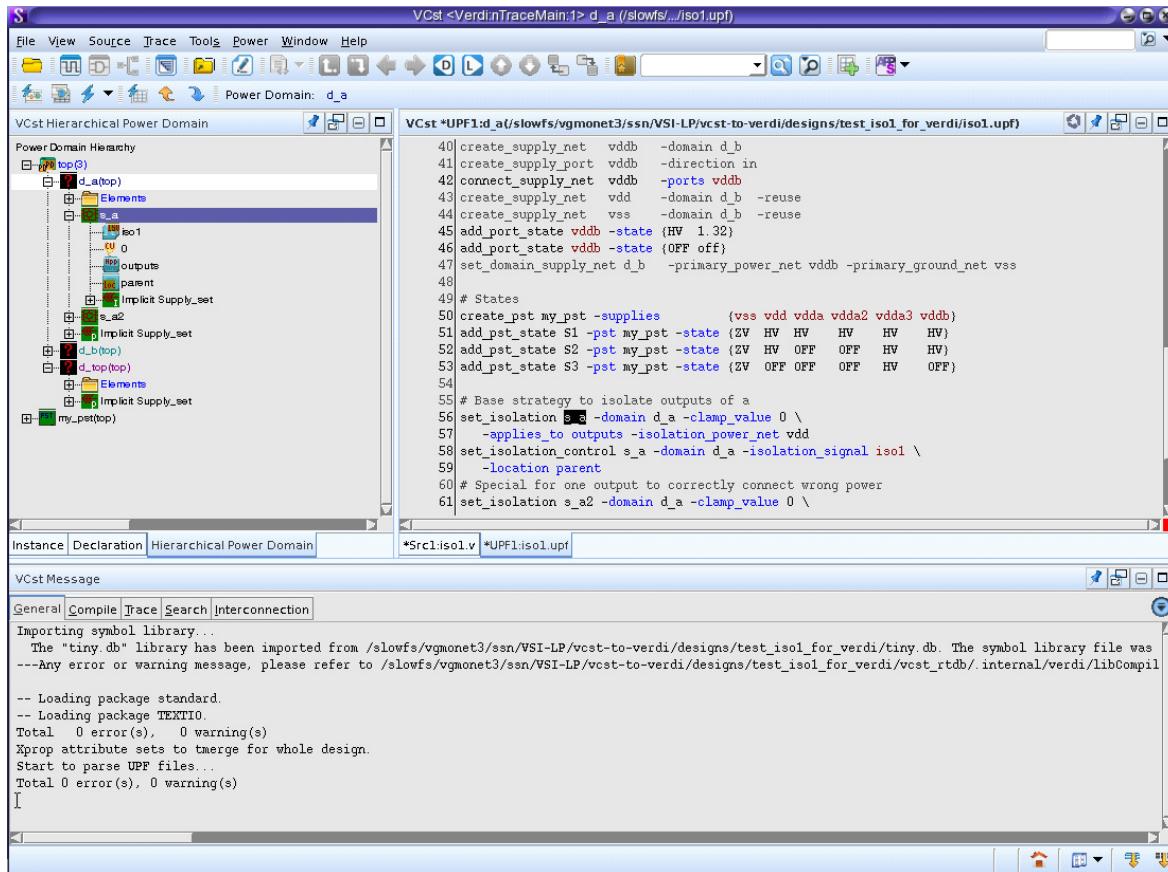
```
%vc_static_shell -f <tcl_file>
%vc_static_shell> view_activity
```

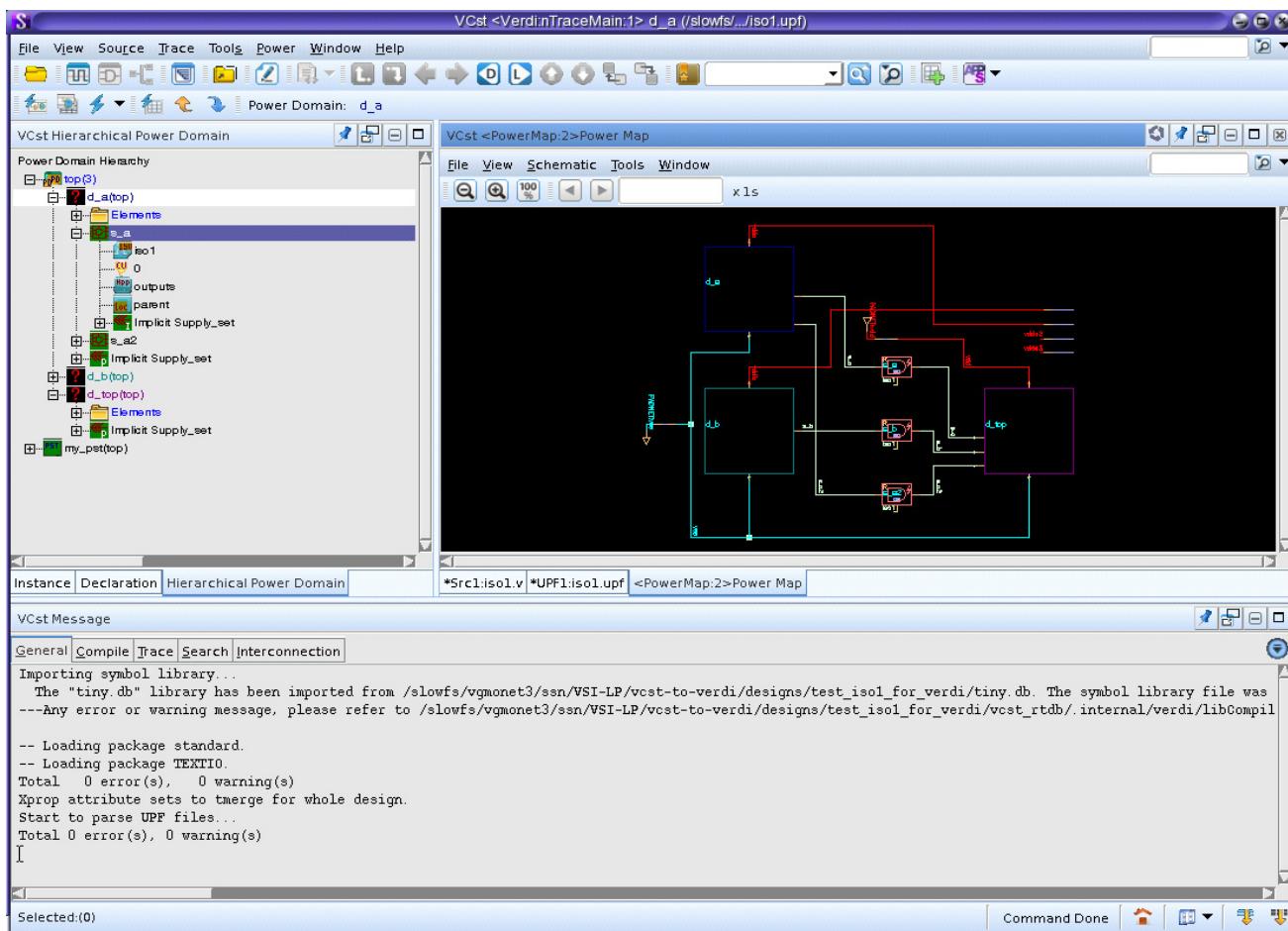
[Figure 4-19](#), [Figure 4-20](#), [Figure 4-21](#), [Figure 4-22](#), and [Figure 4-23](#) show the VC Static Activity View, Verdi Schematic View, Verdi Source View, Verdi UPF view and the Verdi UPF Schematic view when you use the VC Static First Mode.

**Figure 4-19** VC Static Activity View

**Figure 4-20 Verdi Schematic View**

**Figure 4-21 Verdi Source View**

**Figure 4-22 Verdi UPF View**

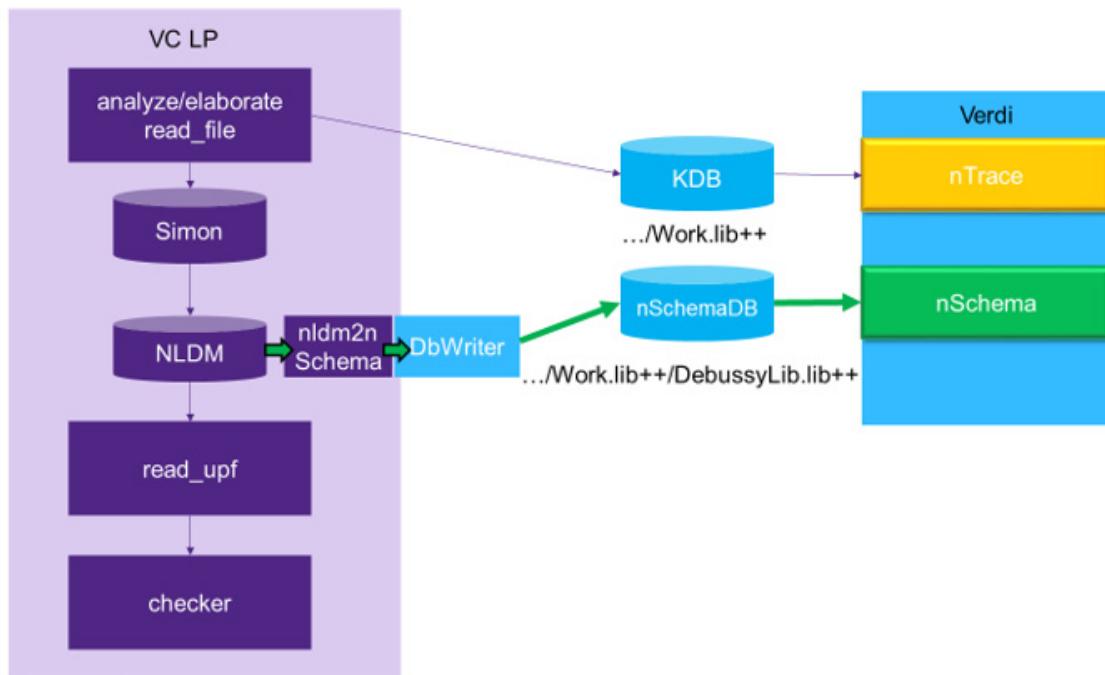
**Figure 4-23 Verdi UPF Schematic View**

### 4.6.3 Support for NLDM Based nSchema Flow

The NLDM based nSchema flow is supported for VC LP. When this flow is enabled, the VC Static NLDM database is converted to Verdi's nSchema database format, and Verdi shows the schematic based on the VC Static NLDM database.

Figure 4-24 shows the database framework of VC LP Verdi integration.

**Figure 4-24 Database framework of VC LP-Verdi integration**



When the NLDM based Schema flow in VC LP is enabled:

- ❖ The database writer for VC LP converts NLDM into nSchema DB.
- ❖ The Tcl interface in VC LP opens the Verdi nSchema to show violation patch (with src/dest signals, and locators to some signals in the path).
- ❖ When Verdi-nSchema triggers event for object selection, VC LP sends NLDM based information to Verdi-nSchema to display in the property view.

#### 4.6.3.1 Use Model

To enable the NLDM2nSchema flow in VC LP, add the following application variable before `analyze`, `elaborate`, `read_file` commands in the VC LP Tcl script:

```
set_app_var enable_nldm_nschema true
```

By default, the `enable_nldm_nschema` application variable is set to false.

There are no changes in the way you invoke Verdi nSchema from VC LP. You can click the **New Schematic Path** link in the **Detail** pane for a selected violation in the Activity view, or click a signal's hyperlink and choose **Schematic** link from the menu.

For violation schematic, you can toggle on the "Add Locators" to show locators in the schematic.

**Figure 4-25 Open Schematic from Activity view**

**Activities**

- static (VCst)
- ification (51112)
- Power (51112)
  - Open (51112)
    - error (47789)
    - UPF-Stage (46070)
    - Design-Stage (1032)
  - ISO\_ELSINPUT\_STATE (180)
    - ISO\_OUTPUT\_STATE (32)
    - LS\_INST\_MISSING (176)
    - RET\_INST\_NOSAVRES (644)
  - PG-Stage (687)
    - warning (3323)
    - UPF-Stage (1130)
    - Design-Stage (2193)
  - Waived
  - Compressed

**Violations: ISO\_ELSINPUT\_STATE**

Msg ID	Instance	Cell	CellPin	PivotInstance	Strategy	StrategyNode	LogicSource
0	test_so10_UPF_ISO	LVLLHCD2	I	test_so10_UPF_ISO			GPRs/GPR4_reg_reg_18_Q
1	test_so21_UPF_ISO	LVLLHCD1HVT	I	test_so21_UPF_ISO			GPRs/GPR5_reg_reg_10_Q
2	test_so33_UPF_ISO	LVLLHCD4HVT	I	test_so33_UPF_ISO			GPRs/GPR9_reg_reg_19_Q
3	B_17_UPF_ISO	LVLLHCD4	I	Multiplier/GENPP/iso_Y_17_UPF_ISO			GPRs/B_reg_reg_17_Q
4	B_17_UPF_ISO	LVLLHCD4	I	B_17_UPF_ISO			GPRs/B_reg_reg_17_Q
5	test_so14_UPF_ISO	LVLLHCD4	I	test_so14_UPF_ISO			GPRs/GPR5_reg_reg_30_Q
6	test_so18_UPF_ISO	LVLLHCD1	I	test_so18_UPF_ISO			GPRs/GPR4_reg_reg_1_Q

**Details: Violation ID#47202**

**Error - ISO\_ELSINPUT\_STATE**

Electrically unprotected path requiring change in supply connection for enabled level shifter test\_so21\_UPF\_ISO. Either input supply on for instance but source supply off, or input supply off for instance but sink supplies on. [More info... \(Help\)](#)

New Schematic Path: [ ] Add Locators [ ]

Create a Filter Template

Instance	test_so21_UPF_ISO	Click link to view schematic
Cell	LVLLHCD1HVT	
CellPin	I	
PivotInstance	test_so21_UPF_ISO	
LogicSource	<input checked="" type="radio"/> GPRs/GPR5_reg	View Item In: Source Code Schematic
LogicSink	<input checked="" type="radio"/> ChipTop_U_comp	Locate in Schematic Add to Schematic
DomainSource	GPRs/test_so21	Copy to Paste-Selection
DomainSink	GPRs/test_so21	
DriverNode	GPRs/GPR5_reg_reg_10_Q	
DriverInfo	DRAWN BY DRC - Ground VCC	

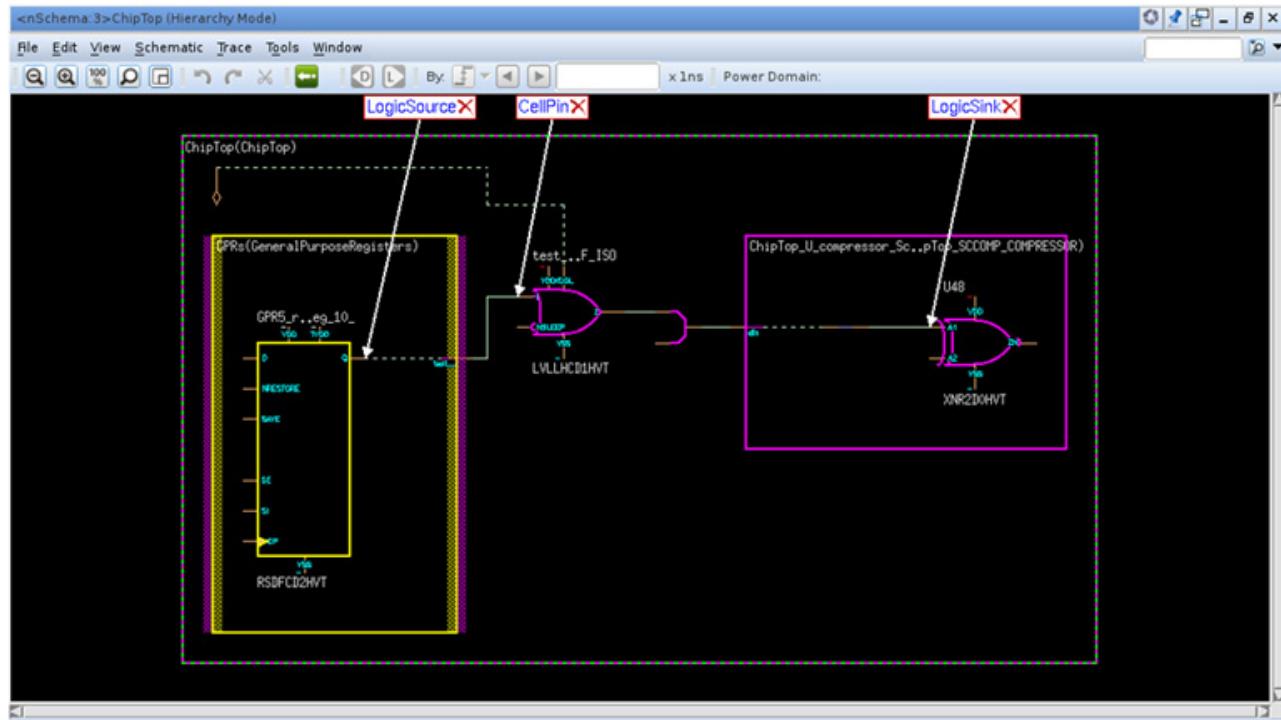
Search:  Live Cell  = ~ LVLLHCD1HVT

Or user can use Verdi toolbar icon to show module schematic for active scope in **Instance** tree, or use RMB menu of Source view to show flattened schematic for selected signal in Source view. The Instance tree and Source view are part of Verdi nTrace component.

#### 4.6.3.2 NLDM-based nSchema

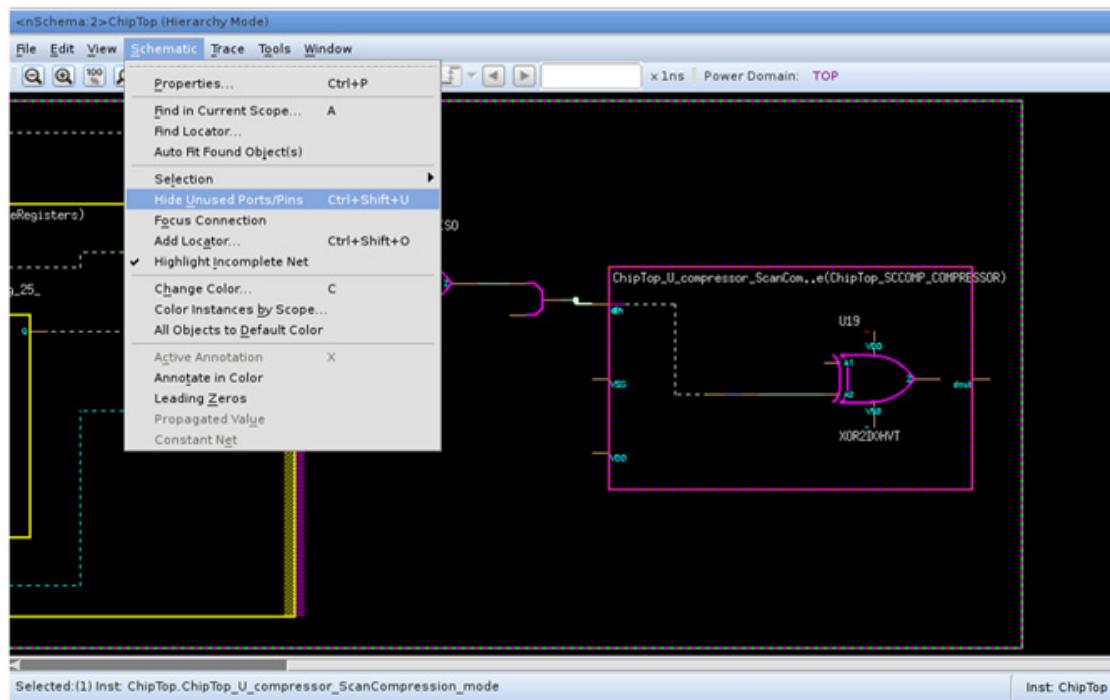
In the NLDM based nSchema flow, both module schematic and flattened schematic are based on NLDM.

- The different Power Domains are shown in different colors with power domain boundary boxes as shown [Figure 4-26](#).

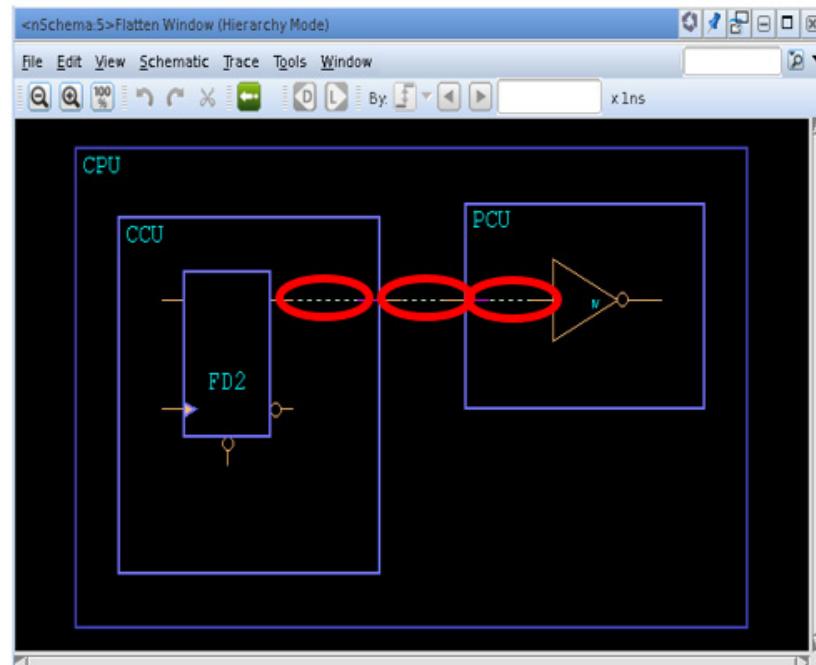
**Figure 4-26 NLDM-based nSchema for VC LP****Note**

UPF Schematic is still based on Verdi's own database.

- ❖ You can hide and unhide unused module ports and pins as shown in [Figure 4-27](#).

**Figure 4-27 Hide Unused Ports/Pins**

- ❖ You can view the partially visible nets whose connection is not complete in current view, and when you double-click on the partial net, the net's connectivity is expanded.

**Figure 4-28 Viewing Partially Connected Nets**

#### 4.6.4 Using Verdi Power Commands

Using the Verdi Power commands, you can browse the design and power aware source code and perform interactive debugging of designs. The various Power Aware Debug functions are docked to existing frame locations.

By default, the **Power Domain Tree** frame showing the power domain list or hierarchy is docked to the same frame as the design browser as either a **Flatten Power Domain** tab (CPF file only) or **Hierarchical Power Domain** tab (UPF file only).

The **Power Mode Table** and **Power State Table** frames are docked to the same frame as the signal list frame (the middle frame) as new tabs after the **Power -> Show Power Mode Table** or the **Power -> Show Power State Table** toggle commands are turned on (the defaults for **Show Power Mode Table** and **Show Power State Table** are off).

The **Power Map** and **Partial Power Map** frames are docked to the same frame as the source code/schematic frame as a new tab after the related **Power -> New Power Map -> Full/Partial Power Map** commands are invoked.

The **Power** pull-down menu in the nTrace window is available after a CPF/UPF file is loaded from the **File -> Import CPF/UPF Files** command or the Verdi command line.

From the **Power** menu, you can perform the following:

**Table 4-7 Verdi Power Commands**

For more details on how to	See the relevant section in the <i>Verdi® and Siloti® Command Reference Guide</i> .
Turn-on or turn-off the display of the power state/mode table frame.	see section <i>Power State/Mode Table Frame</i>
View the <b>Flatten Power Domain/Hierarchical Power Domain</b> tabs	see section <i>Power Domain Tree Frame</i>
Turn-on or turn-off the display of the full power scope name in the <b>Power Domain Tree</b> , <b>Power Mode Table</b> , and <b>Power State Table</b> frames	see section <i>Full Scope Name</i>
Open a <b>Scope Lists</b> form where scopes to report the power impacted signals can be specified	see section <i>Report Impacted Signals by Scope</i>
Open the <b>Check Power Sequence</b> form where the power rules to be checked may be selected and the check results may be reviewed	See section <i>Check Power Sequence</i>
Open the <b>List HDL Signals</b> form where all HDL signals used in the CPF/UPF files are listed	See section <i>List HDL Signals</i>
View all the currently loaded power domains and the specify colors for each power domain.	See section <i>Highlight Power Domain</i>
View the <b>Power Map</b> frame displaying a schematic view of power aware objects	See section <i>Power Map and Partial Power Map Frames</i>

You can download the *Verdi® and Siloti® Command Reference Guide* from [SolvnetPlus](#).

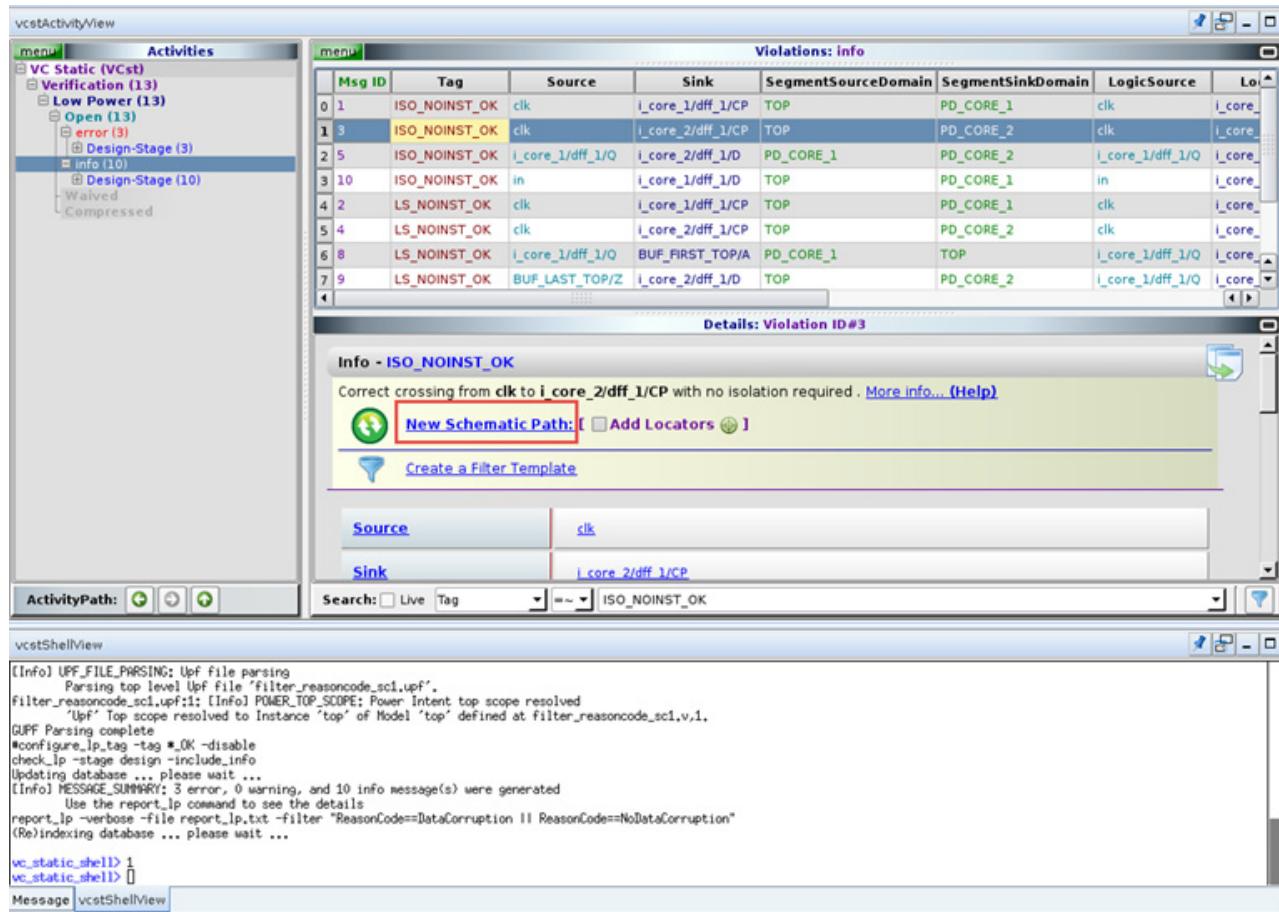
## 4.6.5 Opening Multiple nSchema Windows

VC Static can open multiple nSchema windows for each *violation report* and *node* in LP. You can also add any instance or signal to the active window. Following are the steps to open schematic windows.

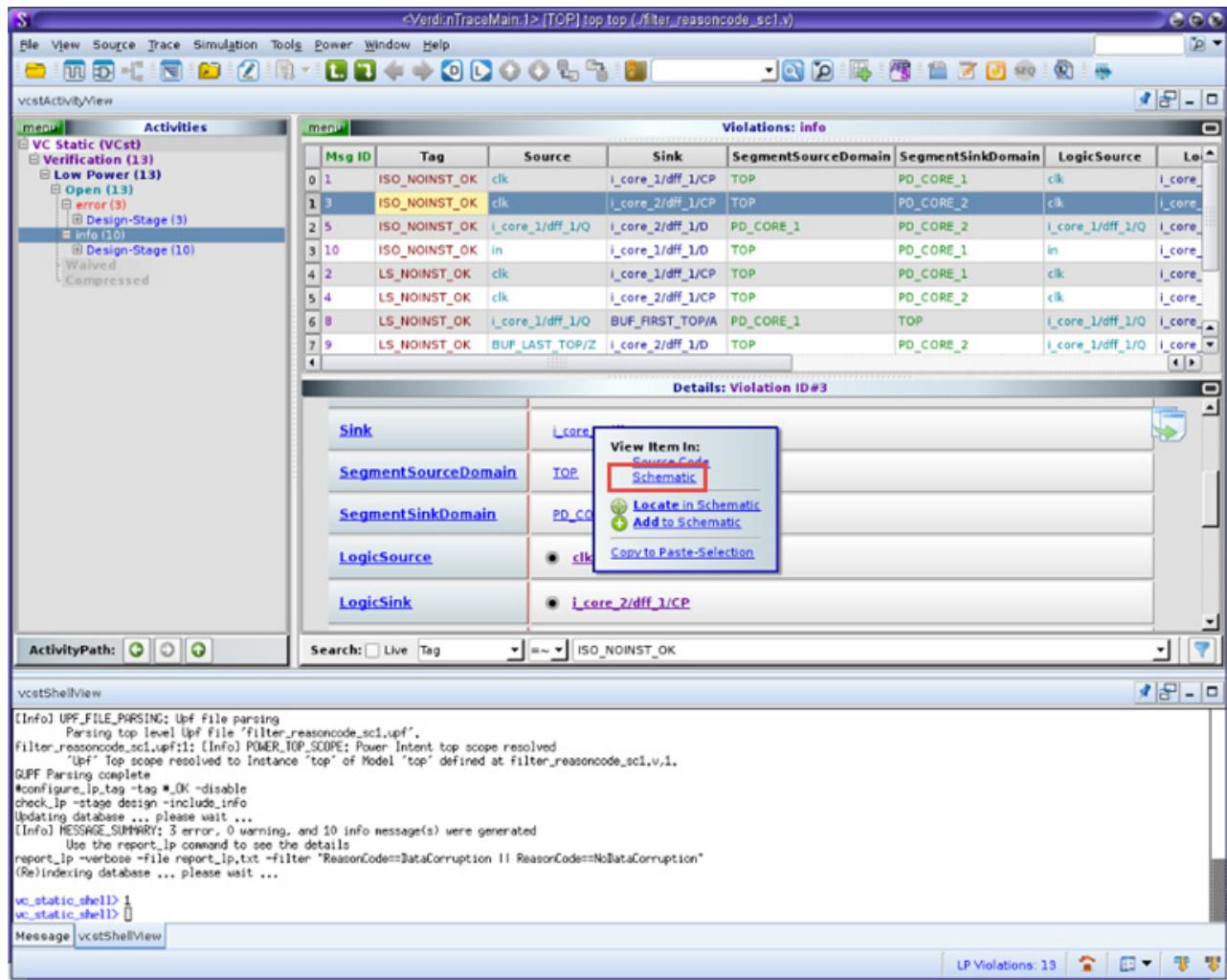
### 4.6.5.1 VC LP Schematic Entries

- ❖ If you click **New Schematic Path** for each *violation report*, a new Verdi-nSchema window is opened. it opens the current active flatten window. if there is no nSchema flatten window or the active one is not a flatten window, then a new window is opened.

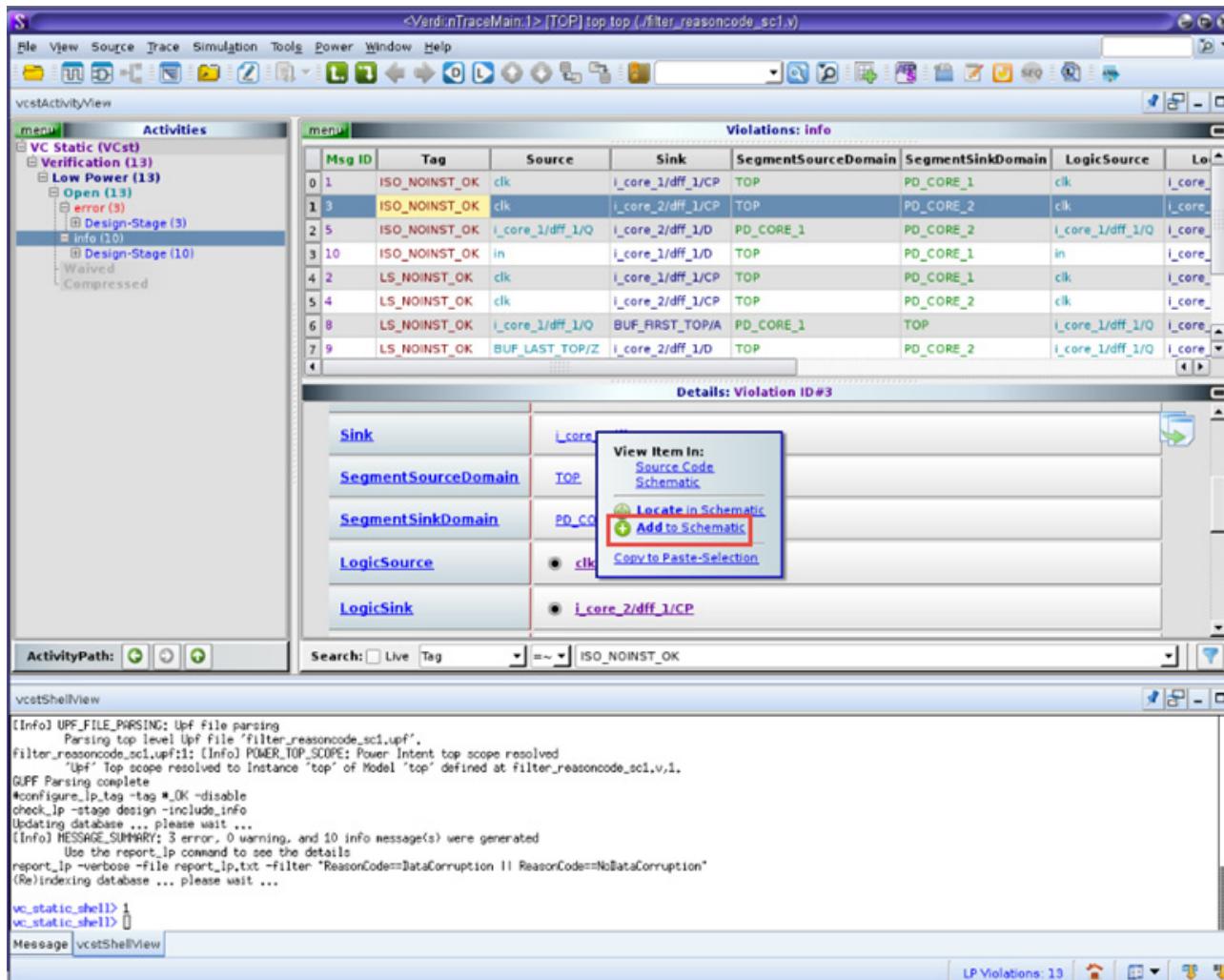
**Figure 4-29 New Schematic Path**



- ❖ If you click **schematic** of each *node*, then a new Verdi-nSchema window is opened.

**Figure 4-30 Schematic Path**

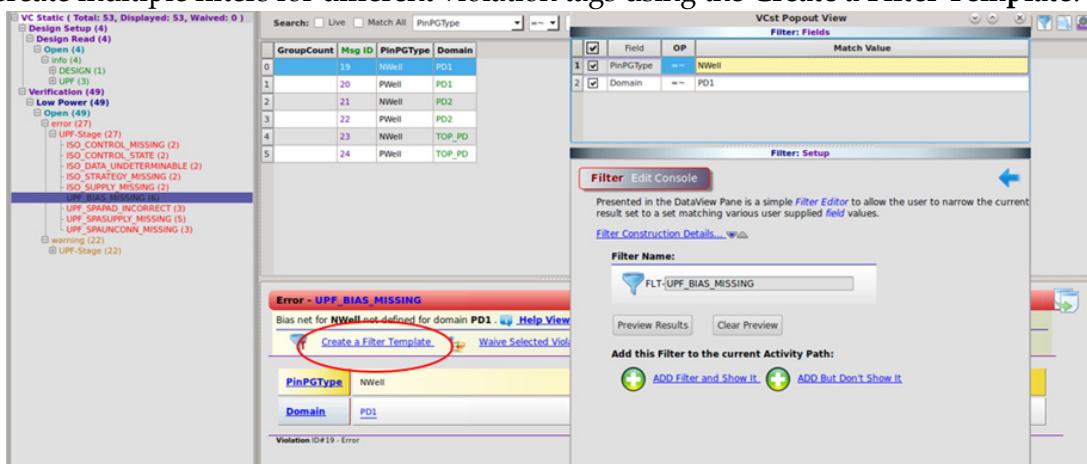
- ❖ If you click **Add to Schematic** of each *node*, the current active flatten window is opened. The last nSchema window you clicks on is the active window, the **active window** appears in the window title. If there is no flatten window or the active one is not flatten window, then a new window appears.

**Figure 4-31 Add to Schematic Window**

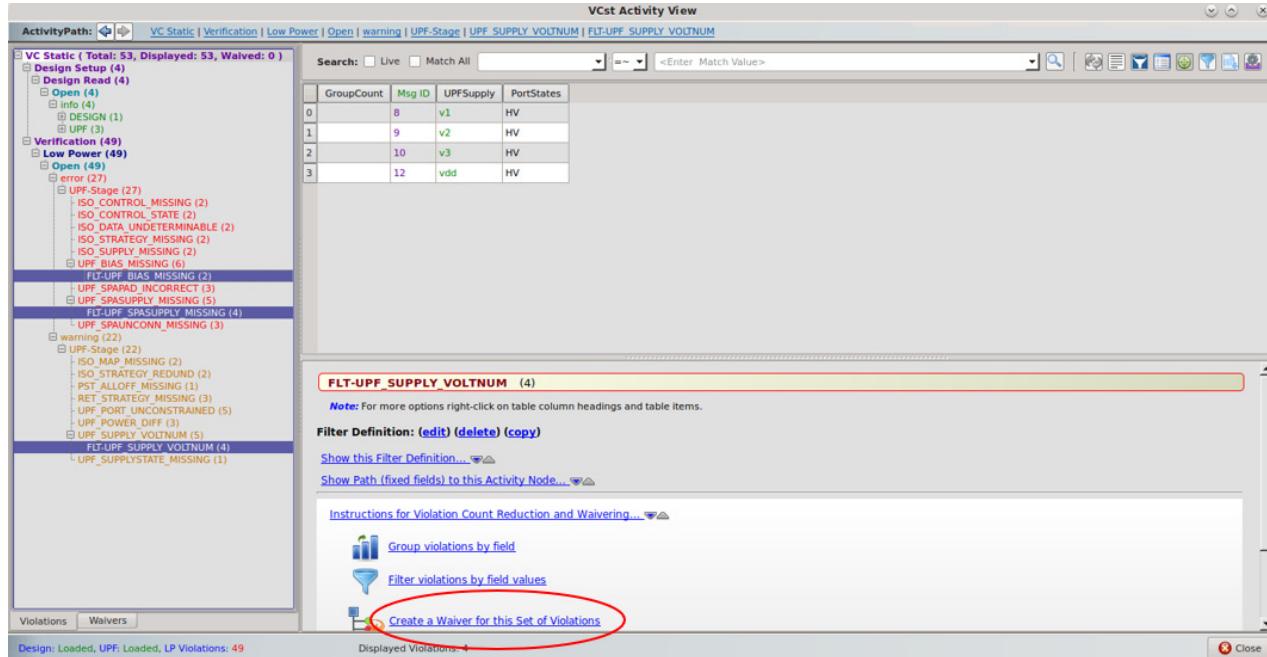
#### 4.6.6 Creating Multiple Filters

You can waive multiple filters at the same time in the view\_activity GUI.

1. Create multiple filters for different violation tags using the **Create a Filter Template**.



2. Select multiple filters using the left mouse click on the Activity tree, and then Ctrl + Select additional filters in the tree view.
3. After selecting multiple filters, select **Create a Waiver for this Set of Violations** in the Info view to convert all the selected filters into corresponding waivers. The waiver names are auto generated based on original filter names as expected.



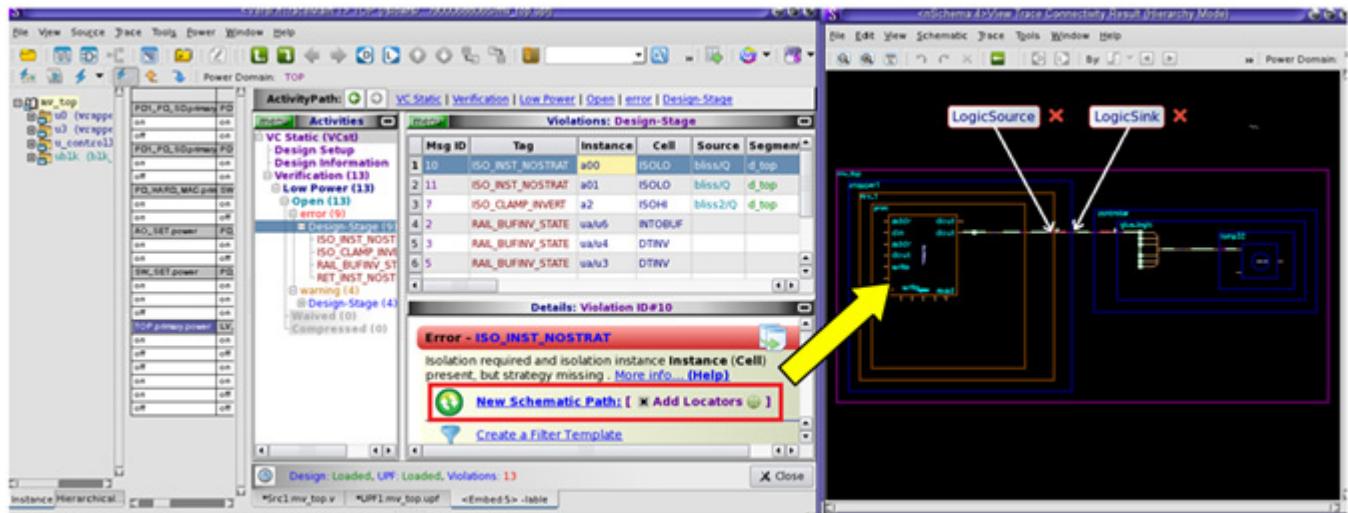
## 4.6.7 Using the Locator Function in Verdi Schematic Viewer

Verdi schematic viewer nSchema supports the locator function as VC static native schematic gui. You can add a locator from the VC Static Activity View menu entry to nSchema window. You can also add locator in nSchema window for any instance or pins.

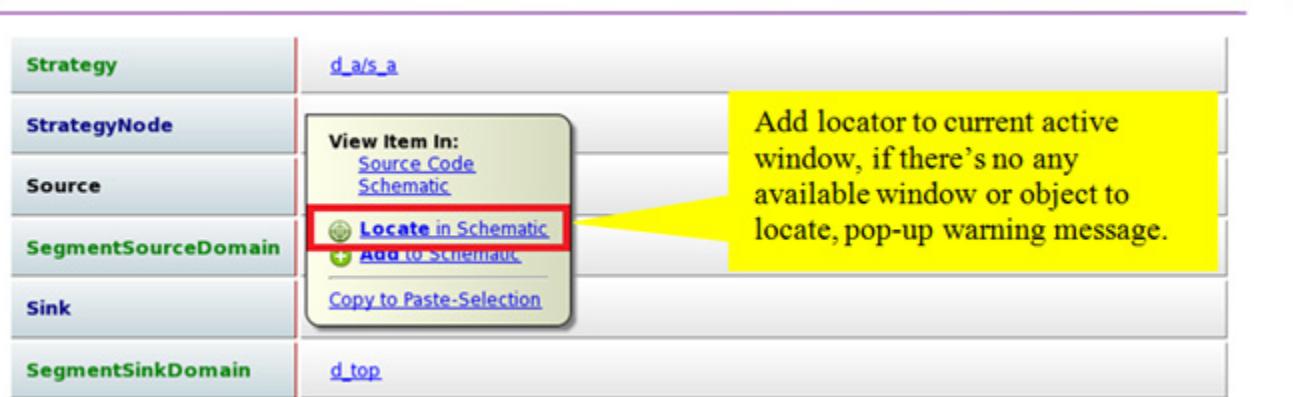
### 4.6.7.1 Adding a Locator from VC Static Activity View Menu Entry to nSchema Window

You can add locator from VC Static Activity View menu entry to nSchema window in the following ways:

1. Click **Add locator** when the new schematic window is open as shown in [Figure 4-32](#):

**Figure 4-32 Adding Locator when New Schematic Window is Open**

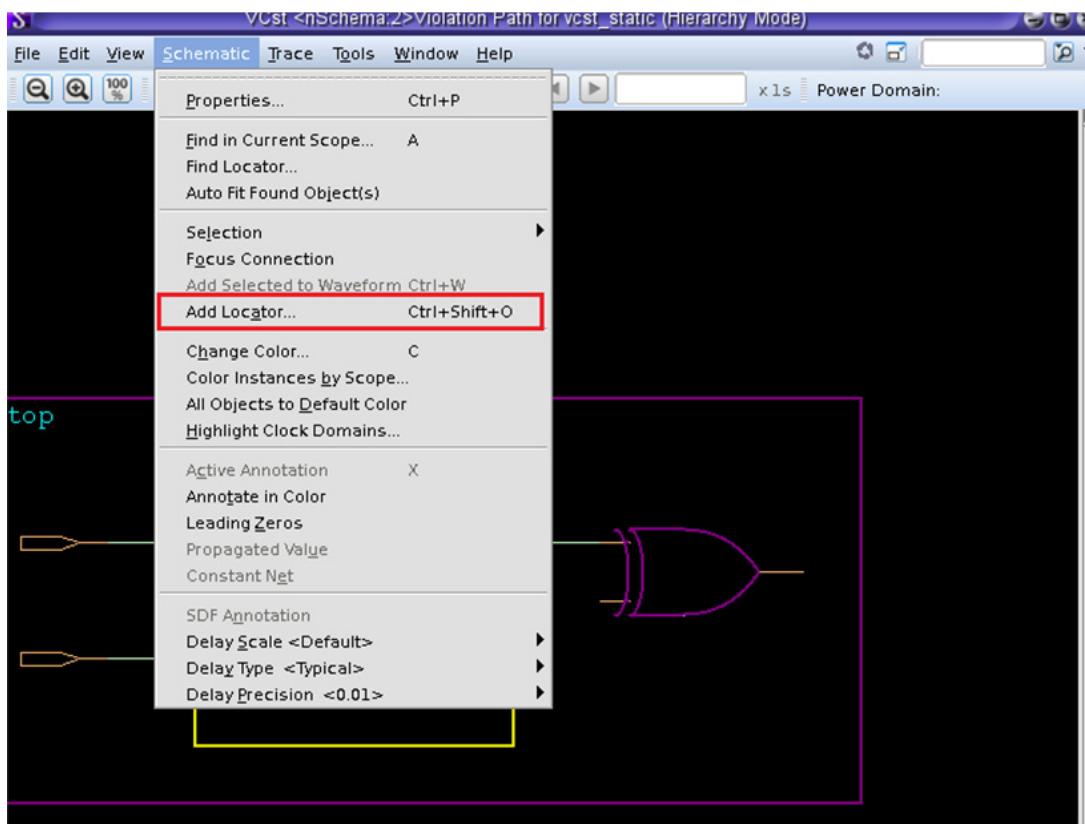
2. Add a locator using the **Locate in Schematic** option from the from VC Static Activity View menu entry as shown in [Figure 4-33](#)

**Figure 4-33 Locator from VC Static Activity View Menu Entry**

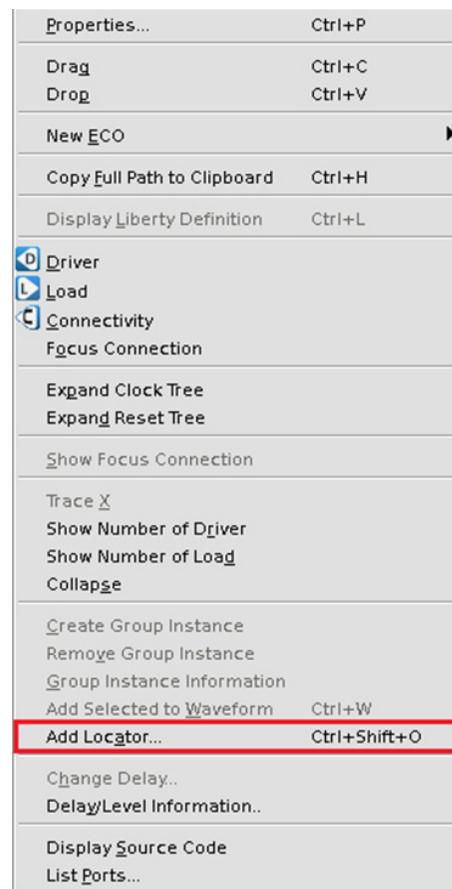
Verdi-nSchema always add a locator to the current active flatten window. If the current flatten window does not contain the object or there is no any flatten window, then Verdi pops-up the following message:

- ◆ Failed to find any available nSchema window, please open new schematic window for this object first.
- ◆ Failed to find any object in current active nSchema window, please open schematic window for this object first.

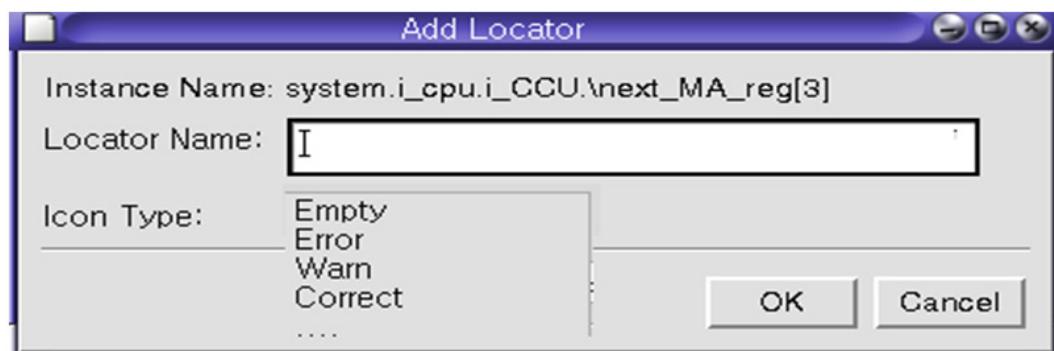
3. You can add locator for any primitive instance/instance port in nSchema window for selected object from the **Schematic-> Add Locator...** menu as shown in [Figure 4-34](#) or from the RMB (Right Mouse Button) menu **Add Locator...** as shown in [Figure 4-35](#).

**Figure 4-34 Adding locator in the nSchema window****Note**

This is enabled when you select a object in the current window. The menu opens the **Add Locator Form** window, you can set the locator name in this form.

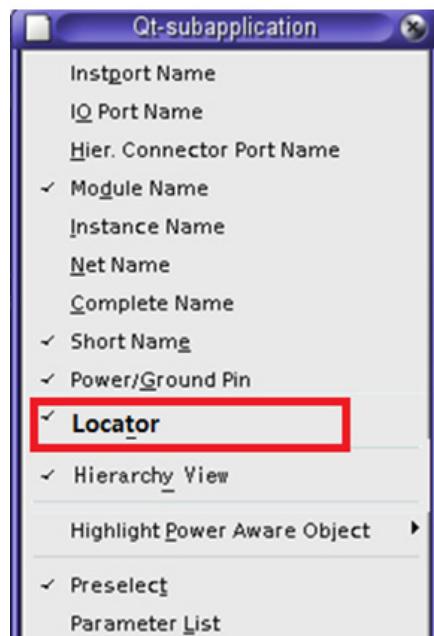
**Figure 4-35 Add Locator from RMB Menu**

The **Add Locator Form** is as shown in [Figure 4-36](#).

**Figure 4-36 Add Locator Form**

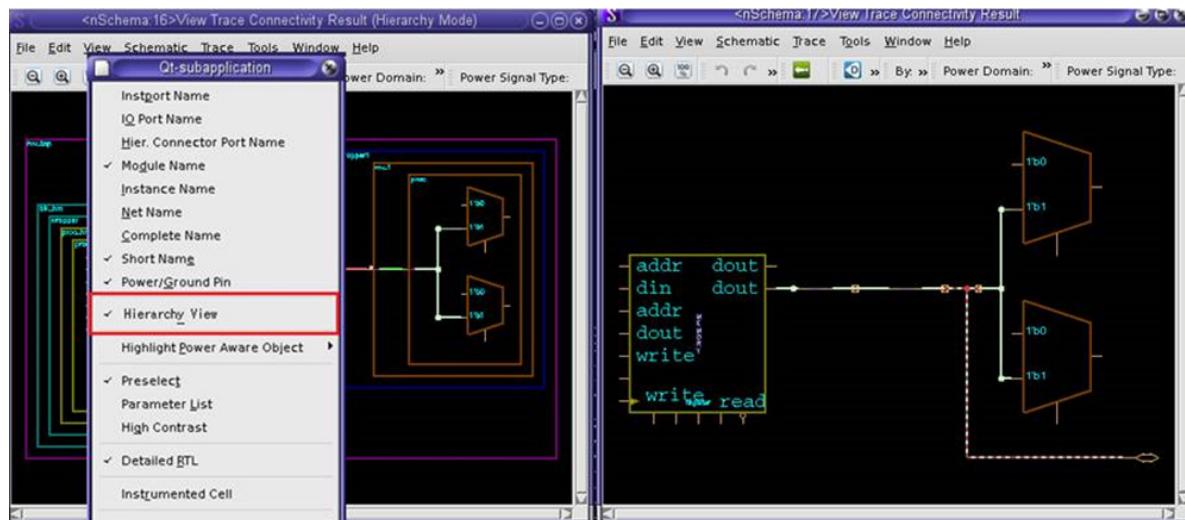
#### 4.6.7.2 Hiding/Displaying Locators

You can use the **View->Show Locator** option to display or hide locator in the nSchema window as shown in [Figure 4-37](#).

**Figure 4-37 Hiding/Showing Locator Option**

#### 4.6.7.3 Viewing Hierarchy

You can use the **View->Hierarchy View** option from the **Window** menu to switch views between hierarchy flatten and flatten mode.

**Figure 4-38 Hierarchy View Option**

#### 4.6.7.4 Finding Locator Form

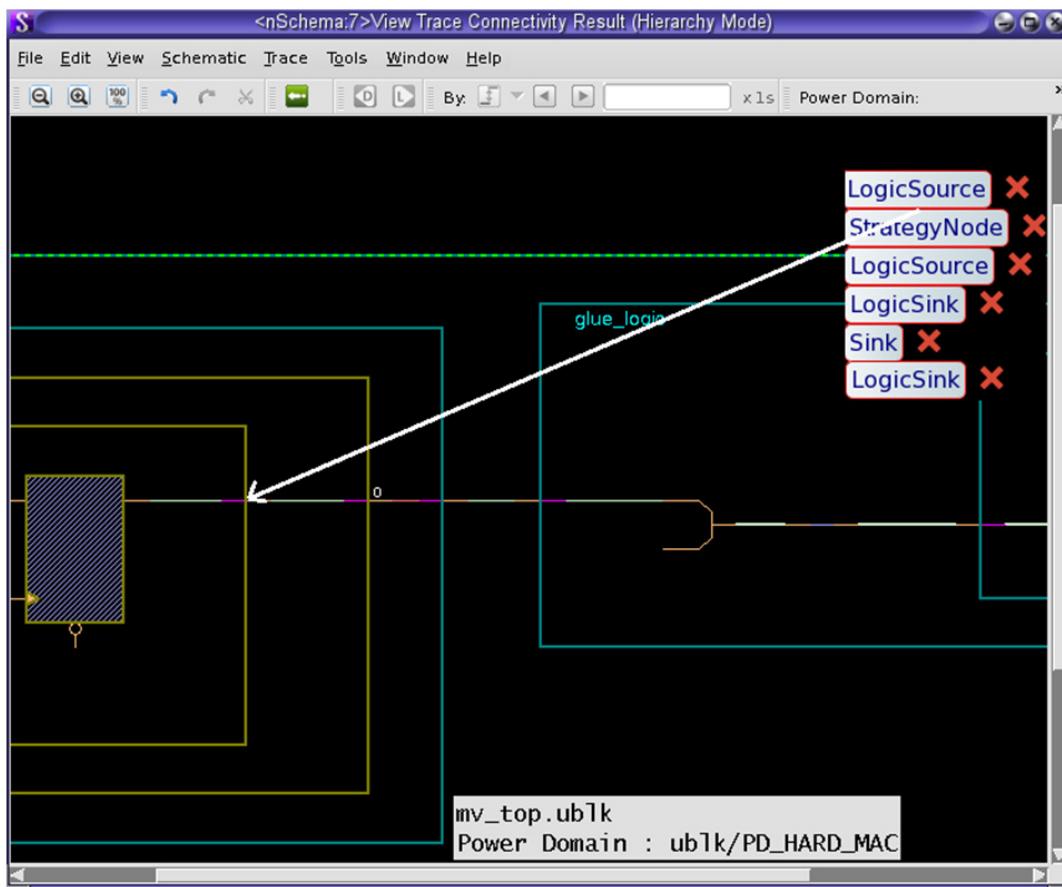
nSchema supports one locator find form as shown in [Figure 4-39](#) to search and highlight one locator. Once you select an item in this form, nSchema window jumps to the corresponding design object, similar to behavior when you double-click on the locator in the drawing area.

**Figure 4-39 Find Locator Form**

#### 4.6.7.5 Aligning Locator Form to Right

You can align all the locators on the right using the **Align Locator Right** menu as shown in [Figure 4-40](#). When you use this option nSchema lays all the locators on the right without overlapping. The locator lines is not drawn in this condition, until you double-click one locator to jump to the corresponding object as shown in [Figure 4-41](#).

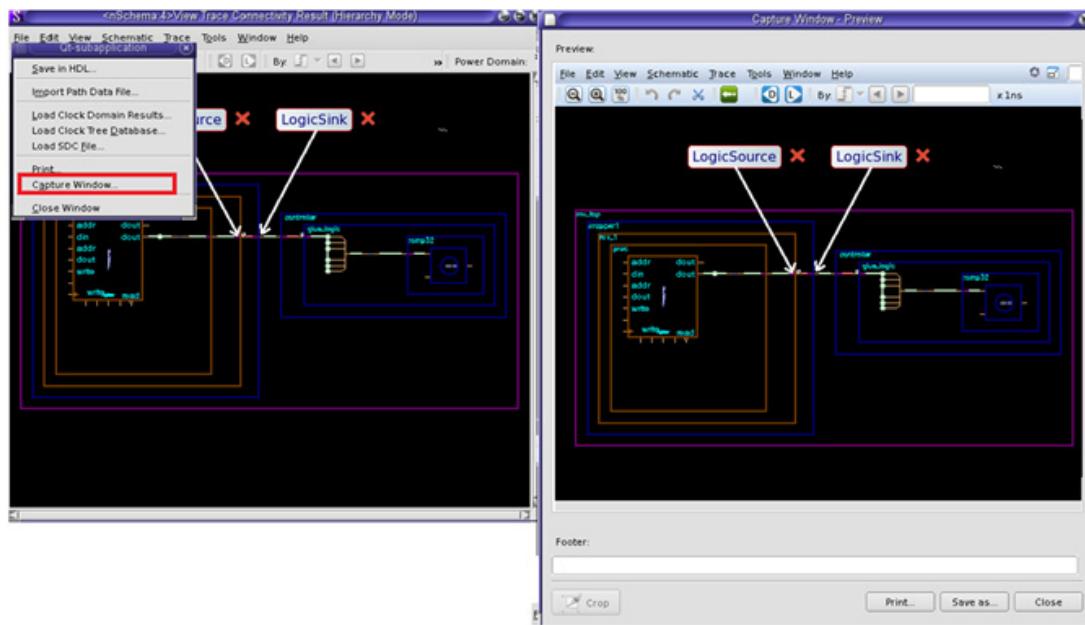
**Figure 4-40 Align Locator Right Menu**

**Figure 4-41 Align Locator Right**

#### 4.6.7.6 Capture Window applications

The Capture Window include the locators as shown in [4.6.7.7](#).

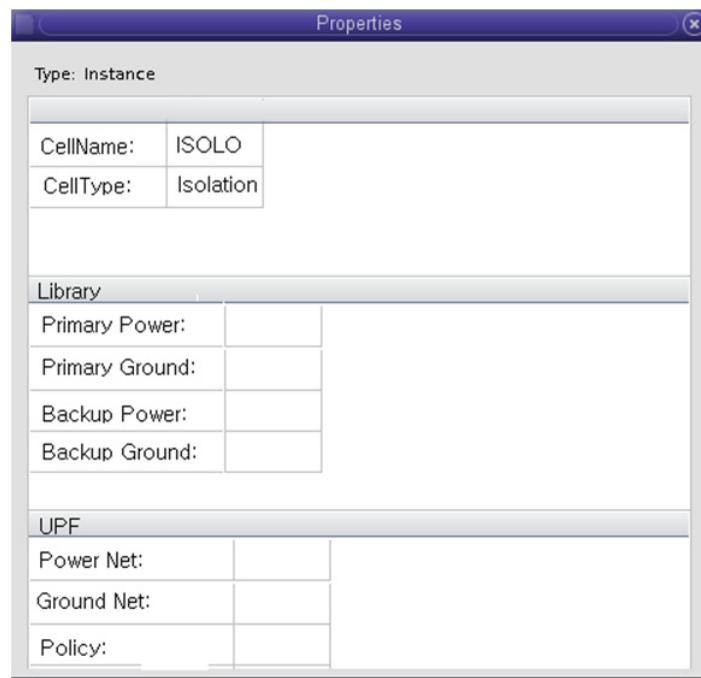
#### 4.6.7.7 Capture Window



#### 4.6.7.8 Property Window

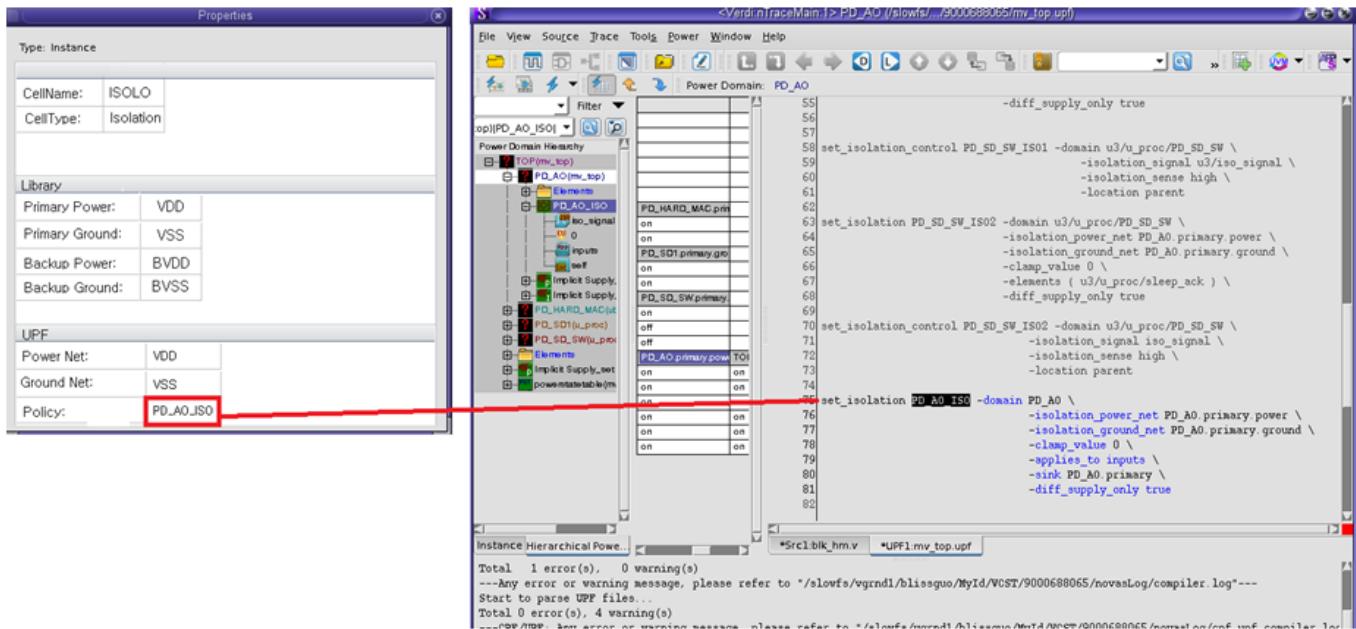
nSchema supports property window as shown in [Figure 4-42](#) to display library and UPF related power information for selected cell/pin. The property window can be opened from the **Property...** RMB (Right Mouse Button) menu. All the properties information is obtained from VC Static.

**Figure 4-42** Property Window



You can DnD (Drag and drop) the design object and power design object from this property window to Verdi's other components for further debugging as shown in [Figure 4-43](#).

**Figure 4-43** . DnD (Drag and drop) the Power Strategy to Verdi-nPowerManager



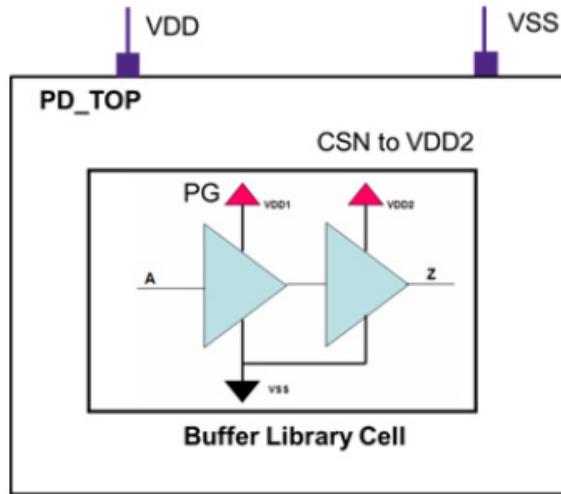
#### **4.6.8 Colorize by Root Supply**

Currently, in the Verdi schematic viewer, you can colorize cells based on the power domain using **Highlight By Power Domain** option. As the number of power domains increases in the designs, it becomes difficult to differentiate power domains with different supplies.

To resolve this issue, an option **Highlight By Root Supply** is introduced in the Verdi GUI. Using this option, you can colorize the cells based on the root supply instead of power domain.

This enables you to easily distinguish between the cells with different supplies which helps in debugging in GUI.

For example, consider the design as shown in [Figure 4-44](#), a buffer library cell is instantiated in a Power domain PD\_TOP whose primary domain supply is VDD. The buffer instance can have a supply different from VDD through use of CSN to VDD2 or direct PG connection to VDD1.

**Figure 4-44 Example Design**

In the following example, three buffers are connected back-to-back with different power supply and all the three buffer reside in TOP domain. The first buffer is connected with TOP domain power supply (VDD\_TOP), the second buffer is powered by PD1 power supply (VDD\_CORE1) and third buffer is powered by PD2 (VDD\_CORE\_2).

#### Design File Snippet:

```
//top module
MY_BUF buf_top (.A(in1), .Z(w1), .VDD(VDD1), .VSS(VSS) );
core1 CORE1 ( .clk(clk), .reset(reset), .in1(w1), .out1(w2), .VDD1(VDD1), .VSS(VSS) );
core3 CORE3 ( .in1(w2), .in2(in2), .out3(out), .VDD1(VDD1), .VDD3(VDD3), .VSS(VSS) );

//Inside module core1
module core1 (clk, reset, in1, out1, VDD1, VSS);
input clk, reset, in1, VDD1, VSS;
output out1;
MY_FF ff_i1 ( .D(in1), .CP(clk), .CD(reset), .Q(out1), .VDD(VDD1), .VSS(VSS));
endmodule

//Inside module core3
module core3 (in1, in2, out3, VDD1, VDD3, VSS);
input in1,in2;
input VDD1, VDD3, VSS;
output out3;
MY_LS_CELL ls_i2 ( .A(in1), .Z(w4), .VDD(VDD1), .VDDH(VDD3), .VSS(VSS) );
MY_OR_CELL or_i1 ( .A(w4), .B(in2), .Z(out3), .VDD(VDD3), .VSS(VSS) );
endmodule
```

[Figure 4-45](#) shows the schematic where cells are colorize by power domain:

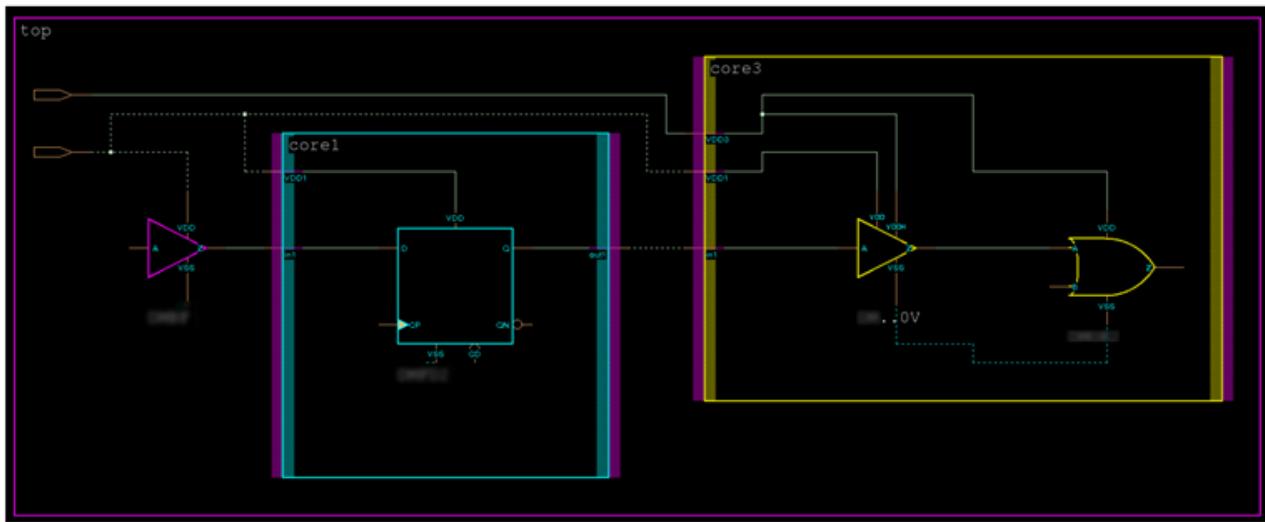
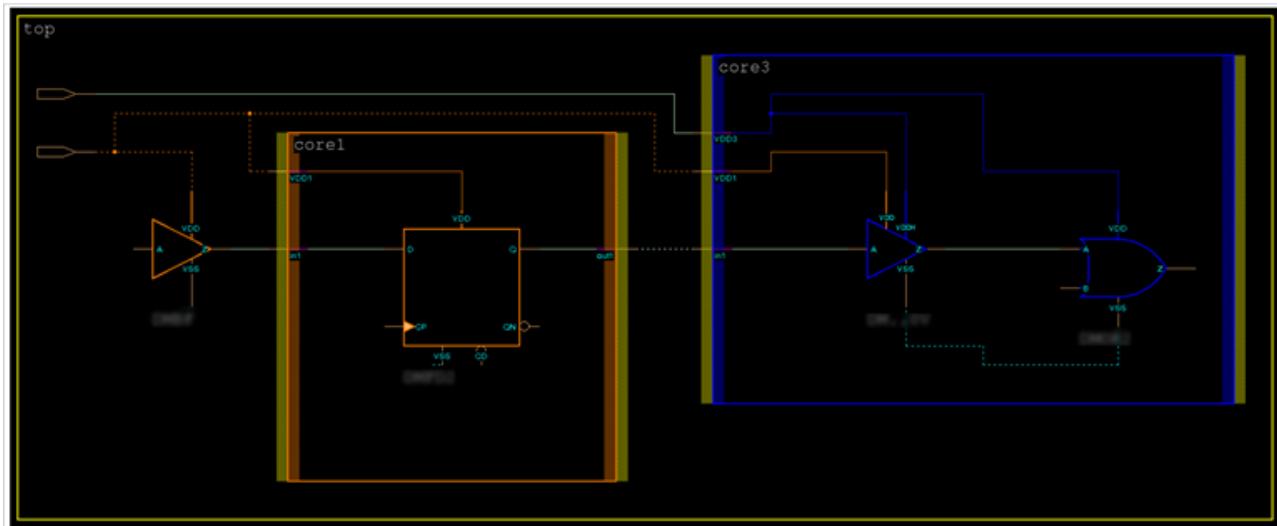
**Figure 4-45 Schematic Where Cells are Colorized by Power Domain**

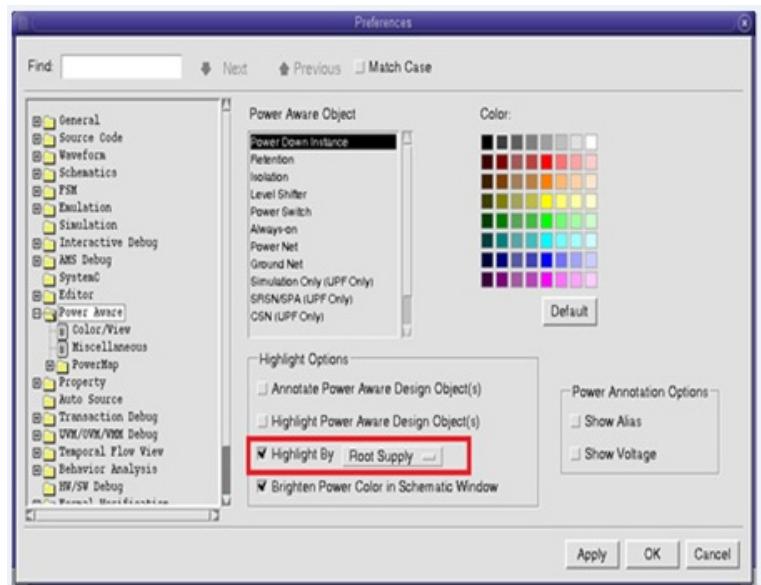
Figure 4-46 shows the schematic where cells are colorized by Root Supply:

**Figure 4-46 Schematic Where Cells are Colorized by Root Supply**

Perform the following in the Verdi GUI mode to colorize cells by root supply:

1. In the Verdi GUI, click Tools > Preferences.
2. In the left box, select to the Power Aware > Color/View option.

3. Select Power Net, and in the **Highlight Options** box, select **Root Supply** from the **Highlight by** drop-down menu.



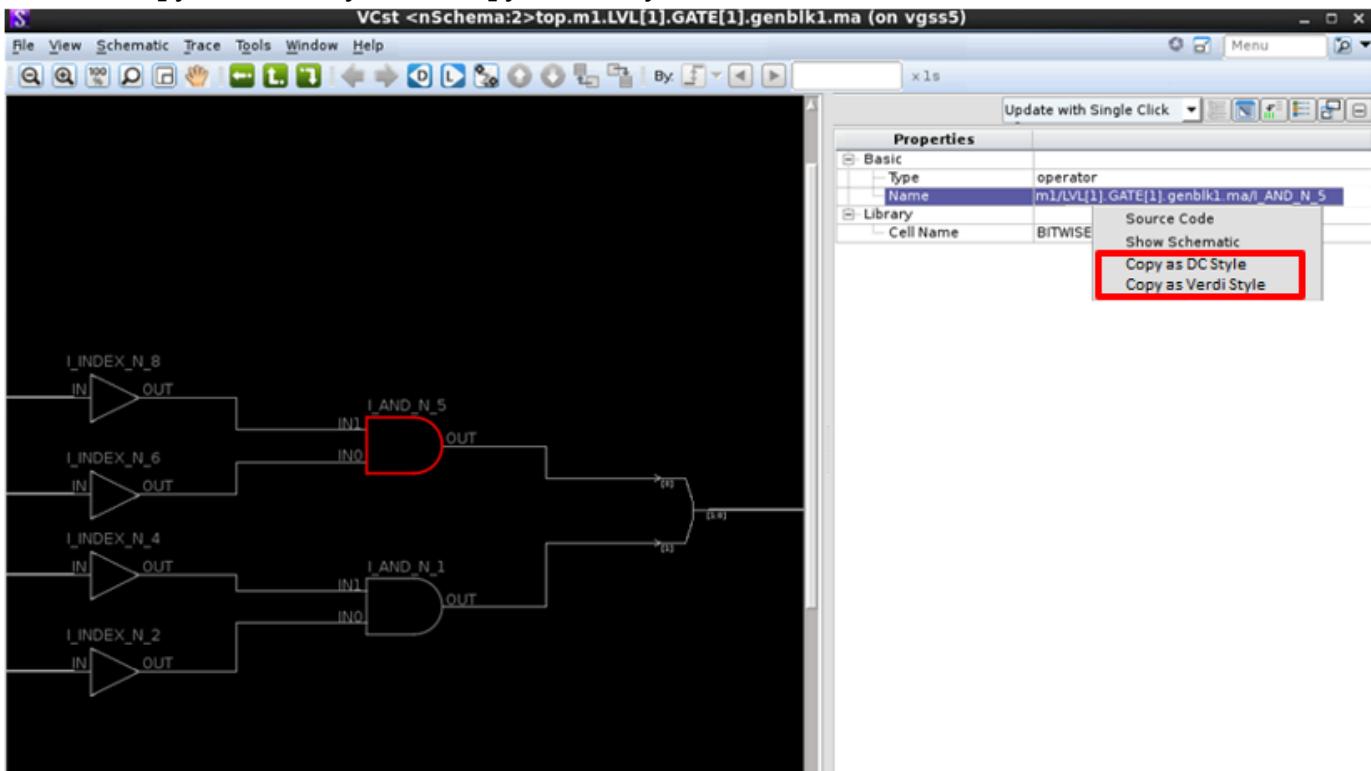
4. Click **Apply**.

By default, **Highlight by Power Domain** is selected.

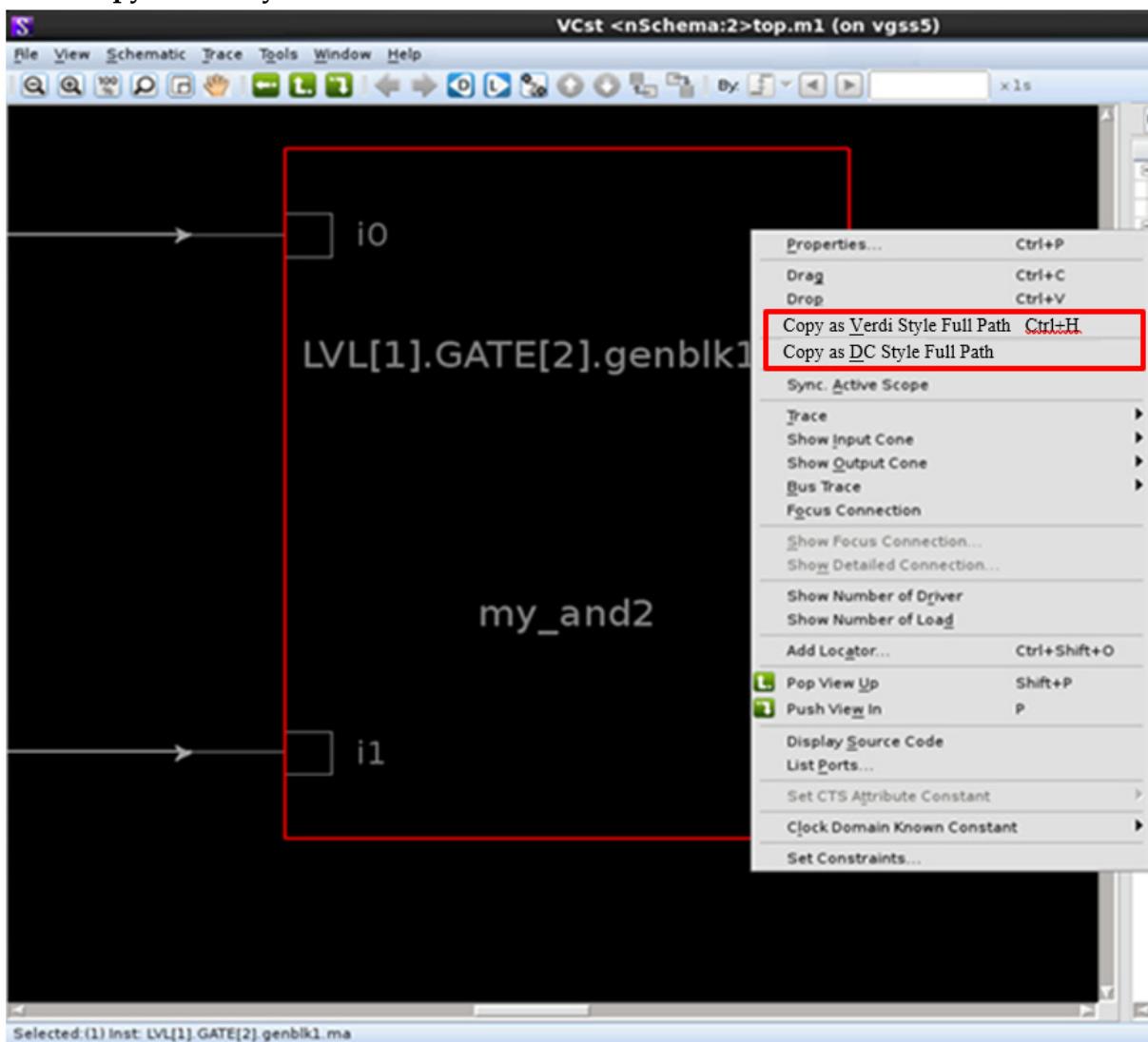
#### 4.6.9 Support for Consistent Name Resolution in VC Static and Verdi

The naming of objects across the VC Static and Verdi nSchema GUI are made consistent.

- ❖ You can copy the object name as Verdi Style or DC style using the nSchema context menu options **Copy as Verdi Style** and **Copy as DC Style**.

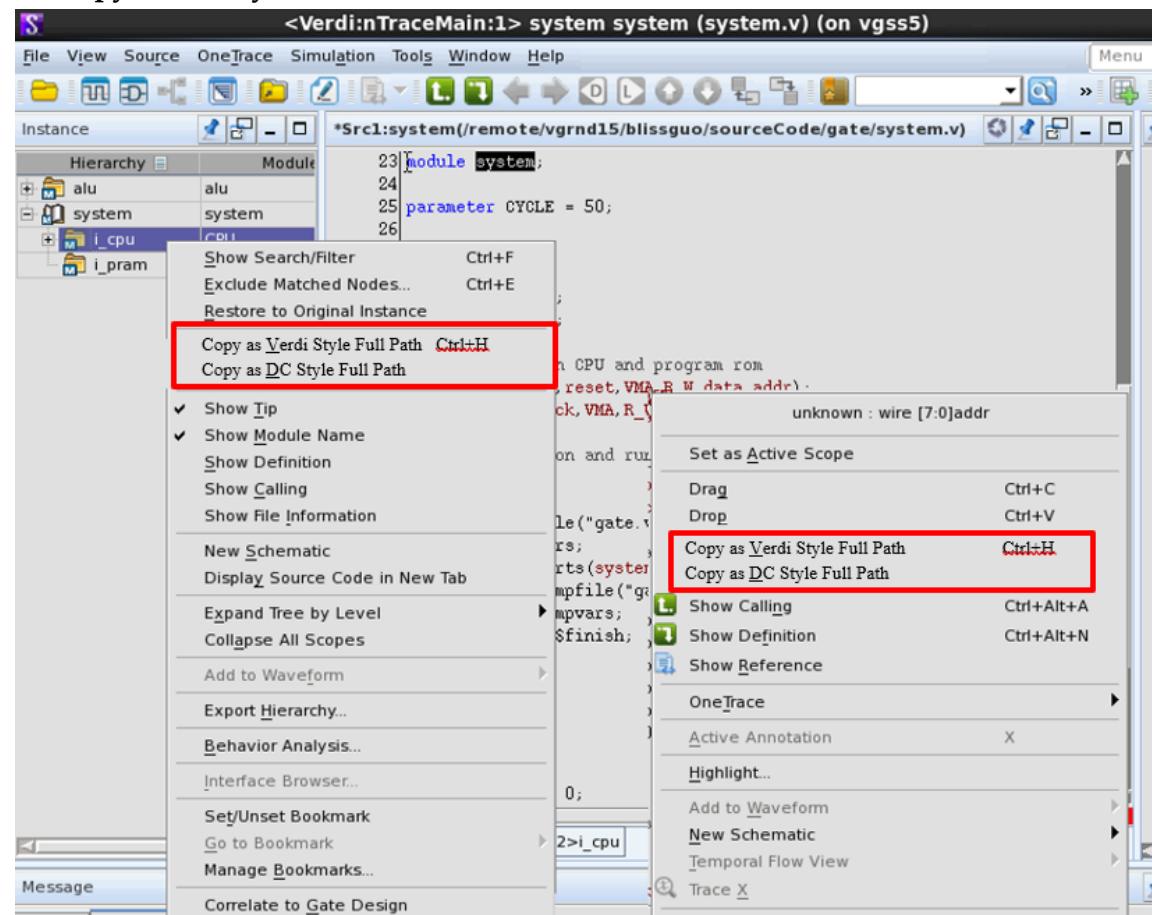


- ❖ You can also copy the full path of the object in Verdi style or DC style, using the RMB menu options in the nSchema window, nTrace context menu on hier. tree, and the source code viewer.
  - ◆ **Copy as Verdi Style Full Path**
  - ◆ **Copy as DC Style Full Path**

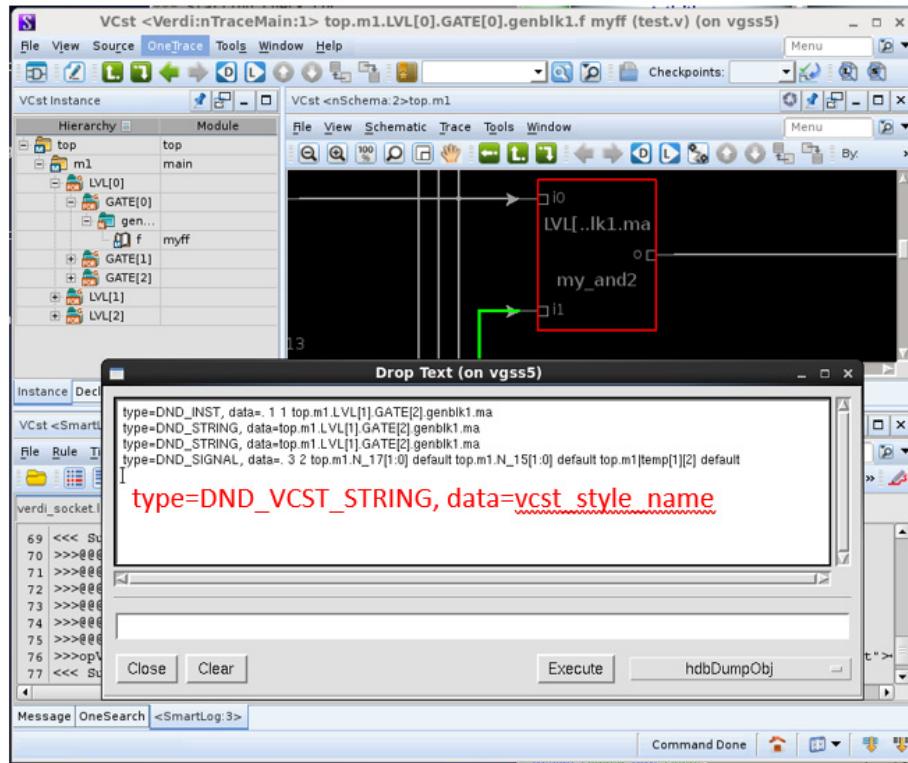


- ❖ You can also copy the full path of the object in Verdi style or DC style, from the nTrace context menu on **Hierarchy** tree, and the source code viewer.
  - ◆ **Copy as Verdi Style Full Path**

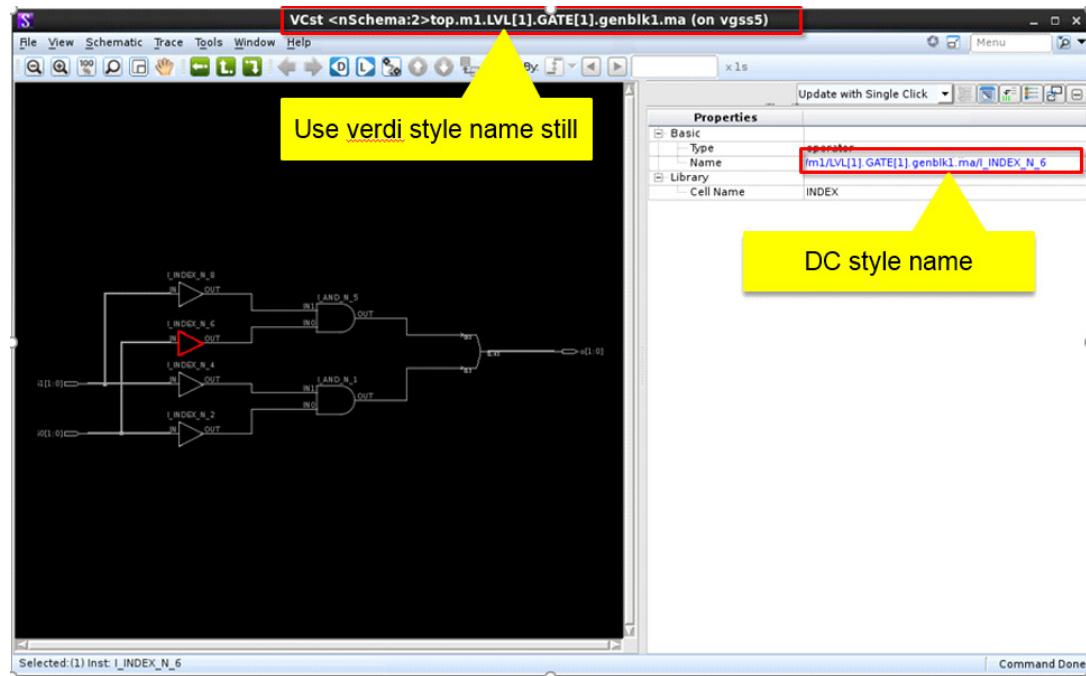
◆ Copy as DC Style Full Path



- ❖ When you drag nSchema and nTrace in the **Drop Text** box, the DND\_VCST\_STRING string is wrapped in the **Drop Text** box.

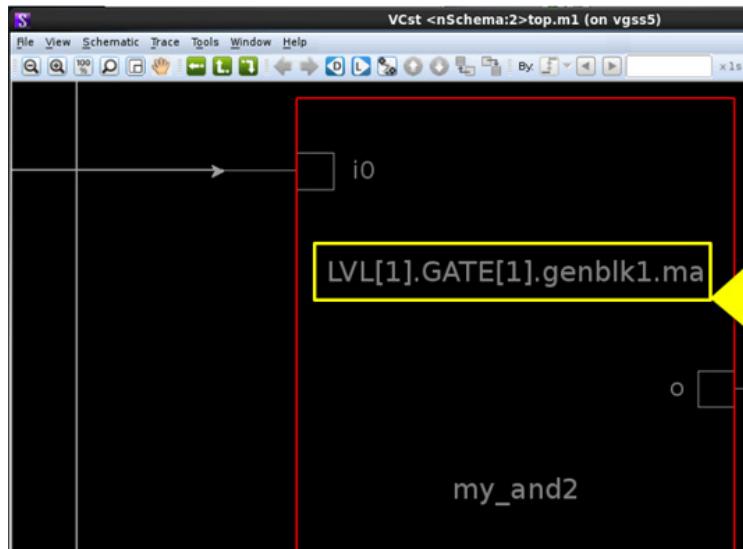


- ❖ The nSchema shows consistent name as VC Static (property window).

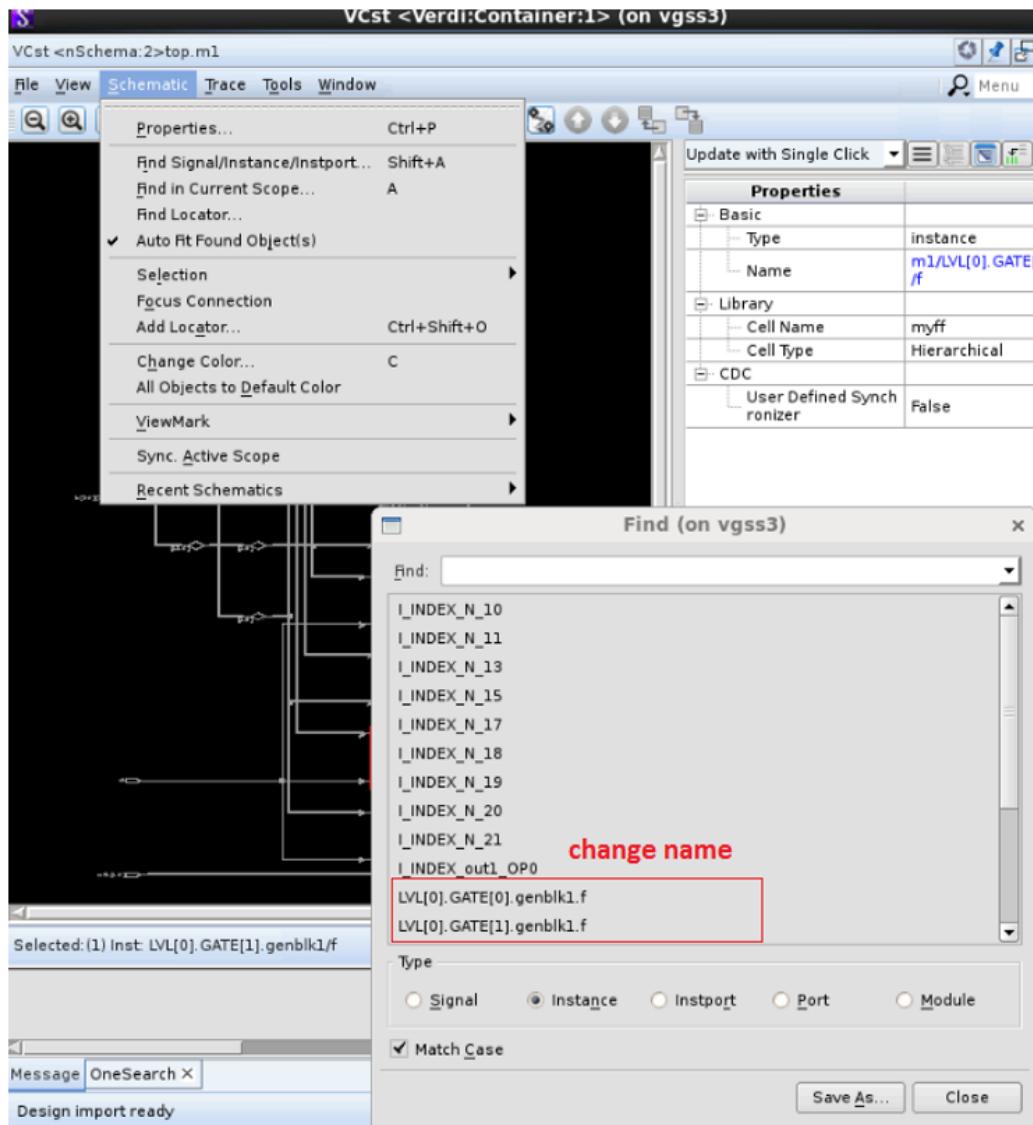


 **Note**

Always use “.” as gen-blk delimiter, since this name is used to DnD to Verdi and other components. If it is changed while drawing, there will be performance overhead.



- ❖ If you have changed the name of an object in VC Static using the `change_name` command, the nSchema shows the updated name. This is supported only in the **Find in Current Scope...** form. The content listed in the **Find in Current Scope...** form shows the changed name, and you can type the changed name into the **Find** box to search.



**Note** This is not currently supported in the **Find Signal/Instance/Instport...** form.

#### 4.6.10 Using VC Static Native GUI

To open the VC Static Native GUI, set the following application variable to false.

```
%vc_static_shell> set_app_var enable_verdi_debug false
```

By default, this application variable is set to true.

##### 4.6.10.1 Viewing the Activity

In VC LP, the primary debug entry point is the Activity View. The command `view_activity` opens this view which guides you through the activities of setting up the design environment, compiling, running

checks, viewing, debugging the violations, and finally categorizing the violations. It is also the primary springboard for linking to the other debug views like the source, UPF and schematic viewer.

The view is organized much like a modern web application with a tight focused context-specific activity flow. The view is divided into two or three panes with the left most pane showing the current activity which drives the other panes.

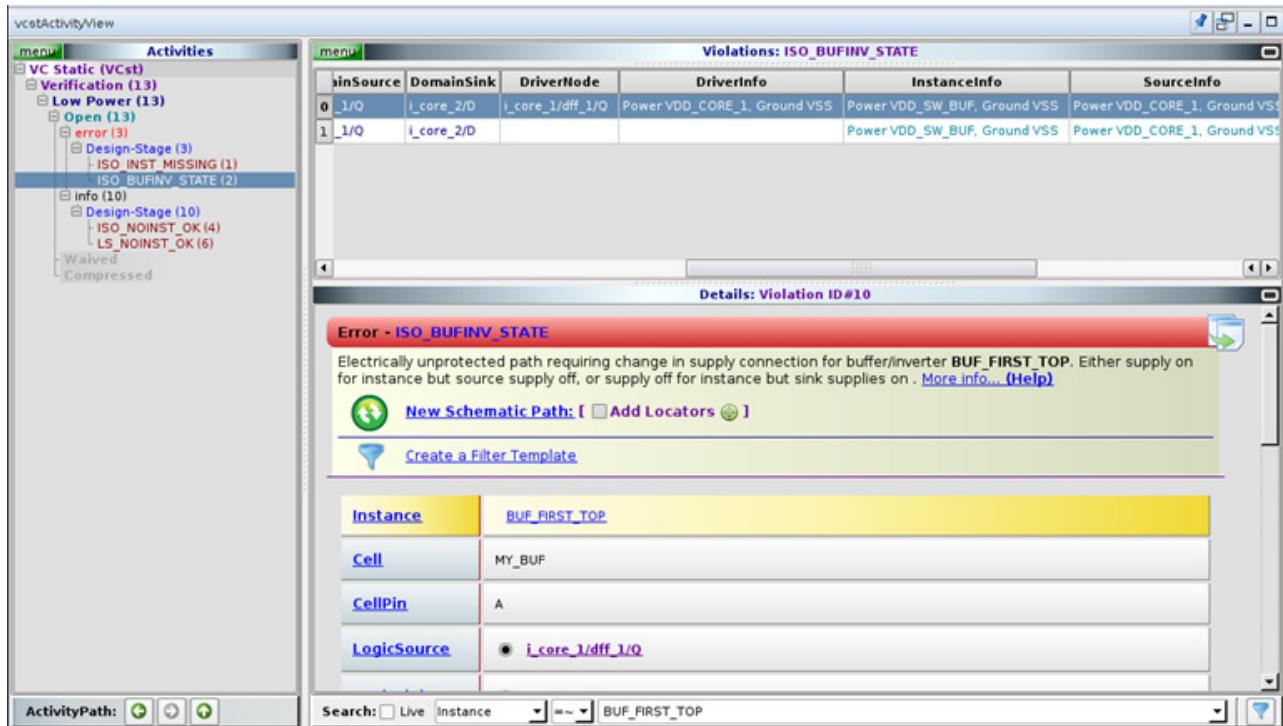
The upper pane as shown in [Figure 4-47](#), if present, is a table view pane used to show results or to allow data entry depending on the current selected activity.

Information, instructions/help along with debug links are presented in the Info pane shown at lower right in [Figure 4-47](#). Again, the contents are sensitive to the current selected activity.

The top left of the view provides the Activity Path control which allows you to step forward and backward in the history of selected activities. An absolute path is also given to directly jump to one of the ancestor activities.

This can also be achieved by clicking on the **i** icon in the lower left of the status bar.

**Figure 4-47 The Activity View**



## Syntax

```
%vc_static_shell> view_activity -help
Usage: view_activity      # Show the Activity 'Hybrid' View
      [-vid <id>]          (show given verification result message ID.)
```

### 4.6.10.1.1 Help Viewer

The `view_help` command opens the general Help viewer. From here, you can link directly to the various documentation available or see command and message man pages.

## Syntax

```
%vc_static_shell> view_help -help
Usage: view_help      # Show the Help Viewer
       [<string>]           (show help for the given command/msg label.)
```

#### 4.6.10.2 Viewing the Schematic

The `view_schematic` command opens a view which contains the graphical schematic of the design. You can view either the entire module at a time or just specified paths of connectivity. When viewing a specified path, you can expand the viewed schematic by double-clicking on unconnected pins in order to reveal additional connections.

Normally, viewing the schematic is driven by clicking on various paths from within the `view_activity` view, but the `view_schematic` command can also accept modules and paths from the VC LP command line.

The `view_schematic` command, without any arguments, will display a top level symbol of all the input/output ports.

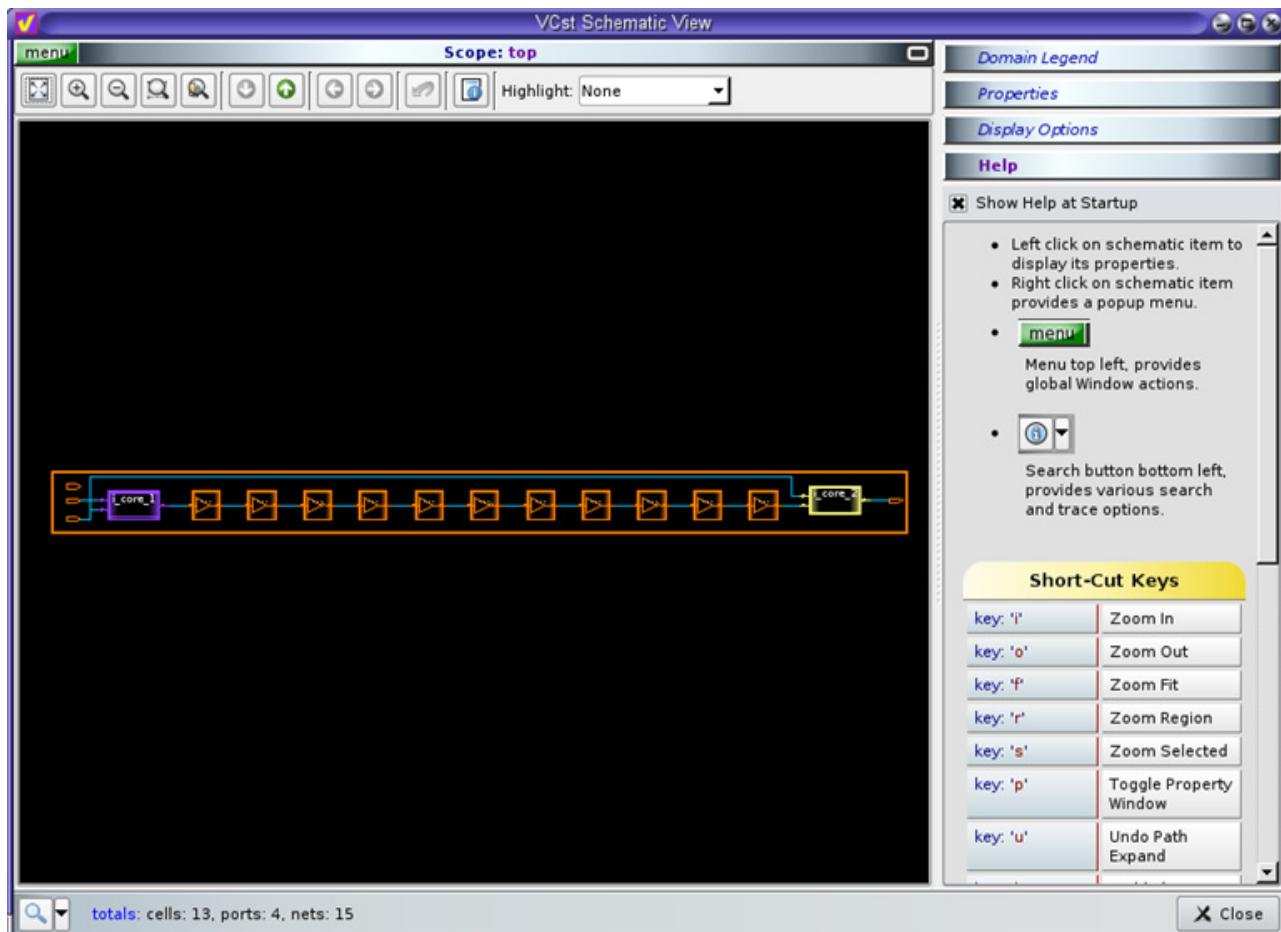
- ❖ If the `-module` option is used, the view will contain the entire contents of the design module.
- ❖ The `-src` and `-dest` options can be used individually or together to display the fan-out, fan-in or path respectively.
- ❖ The `-inst` option will display only the cell specified with all its pins unconnected. These pins can be expanded by double-clicking on each pins.

#### Syntax

```
vc_static_shell> view_schematic -help
Usage: view_schematic      # Display the Schematic for a Module or Trace/Path
       [-block]           (Makes command input block while this command is active.)
       [-run_finish <cmds>]   (Set of commands to run when this command finishes in non-
                                blocking mode.)
       [-cancel]          (Notify command to cancel its operation.)
       [-module <module name>]
                           (View: Display entire module)
       [-locate <name>]     (View: Create Locator for item with name)
       [-src <pin/port name>] (Trace: From source pin/port)
       [-dest <pin/port name>]
                           (Trace: To destination pin/port)
       [-instance <inst/net name>]
                           (Trace: Display instance)
       [-add]
       [-type <path type>]  (show a cdc aware schematic path of specified type:
                                Values: clock, comb_data, reset,
                                seq_data, seq_reset)
       [-title <title>]      (View: Window title)
       [-clear]              (View: Clear/remove all locators)
       [-new]                (View: Create schematic in new view)
       [-save <file name>]    (View: Save view to file)
       [-thru <net/pin/instance name>] (View: Display schematic passing through these
                                         objects)
       [-nschema <nscschema number>] (View: Add the schematic in corresponding nscschema
                                         window having same id)
```

#### Use Module:

```
%vc_static_shell> view_schematic -module top
```

**Figure 4-48 The Schematic View**

#### 4.6.10.3 Viewing the Source

The `view_source` command opens the source view which includes a hierarchy browser and a source viewer

The `view_source` command opens the VC LP Source Viewer. If

`-inst` option is used, the view selects the scope of the object passed to `-inst` in the hierarchy view and shows the object in the source view.

The find bar at the bottom of the view can be shown/hidden using the I button at the bottom left of the view. The find function is applied to the last view which was clicked. If you want to find text in the source view, click the source view, then use the find bar to find the text. If you want to find text in the hierarchy view, click the hierarchy view and use the find bar.

The hierarchy view can also filter out the hierarchy using the regular expression. For this, use the filter bar at the top of the hierarchy view.

#### Syntax

```
%vc_static_shell> view_source -help
Usage: view_source      # Show the Source View
       [-block]           (Makes command input block while this command is active.)
       [-run_finish <cmds>] (Set of commands to run when this command finishes in non-
                                blocking mode.)
```

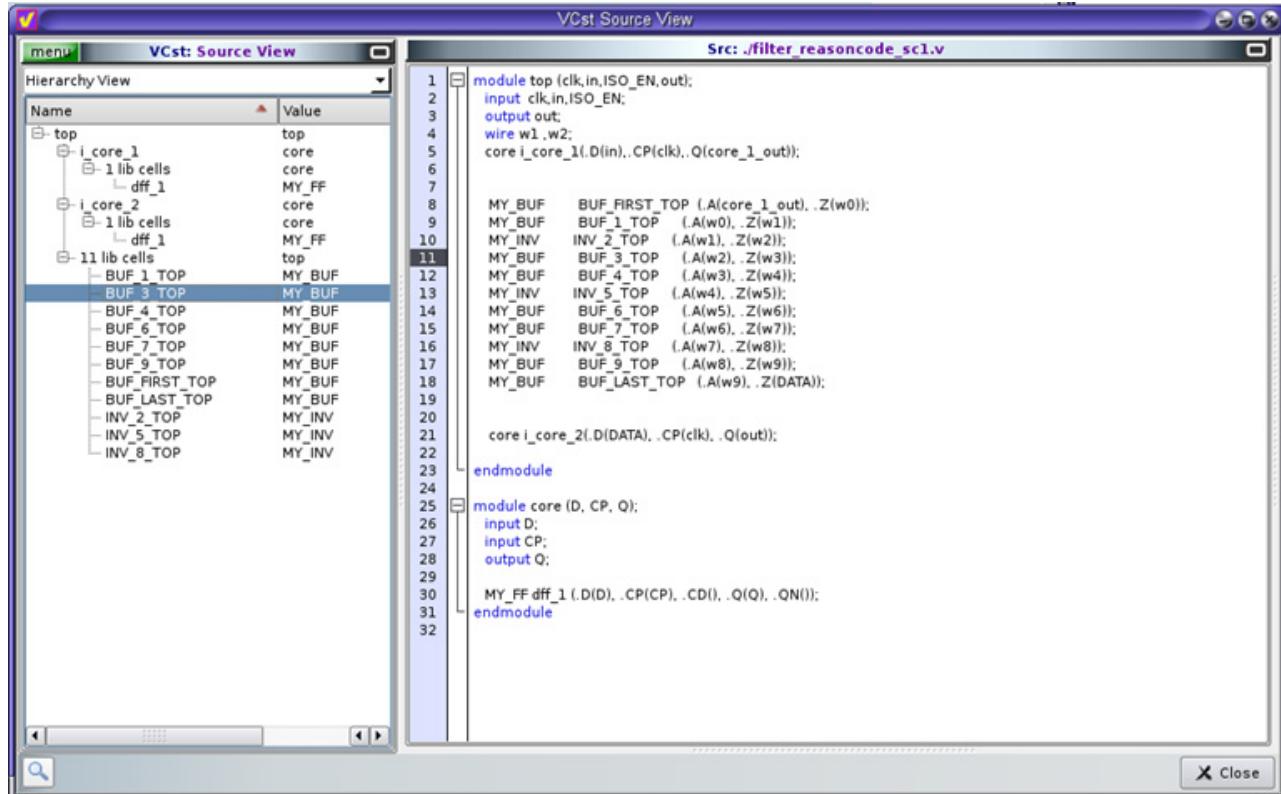
[-cancel] (Notify command to cancel its operation.)  
 [-line <string>] (File name:Line Number)

## Use Model

Here is an example of VC LP Source View obtained with the following command:

```
%vc_static_shell> view_source
```

**Figure 4-49 View Source**



### 4.6.10.4 Viewing the UPF

The `view_upf` command opens the source view which includes a hierarchy browser and a source viewer.

The `view_upf` command is similar to the `view_source` command. It opens the UPF view which contains a UPF tree browser and a UPF file viewer.

#### Syntax

```
%vc_static_shell> view_upf -help
Usage: view_upf      # Show the UPF View
      [-block]          (Makes command input block while this command is active.)
      [-run_finish <cmds>] (Set of commands to run when this command finishes in non-
                           blocking mode.)
      [-cancel]         (Notify command to cancel its operation.)
      [-domain <string>] (Domain name)
      [-strategy <string>] (Strategy name)
      [-state <string>] (State name)
      [-supply <string>] (Supply name)
      [-net <string>] (Net name)
      [-mapcell <string>] (Mapcell name)
```

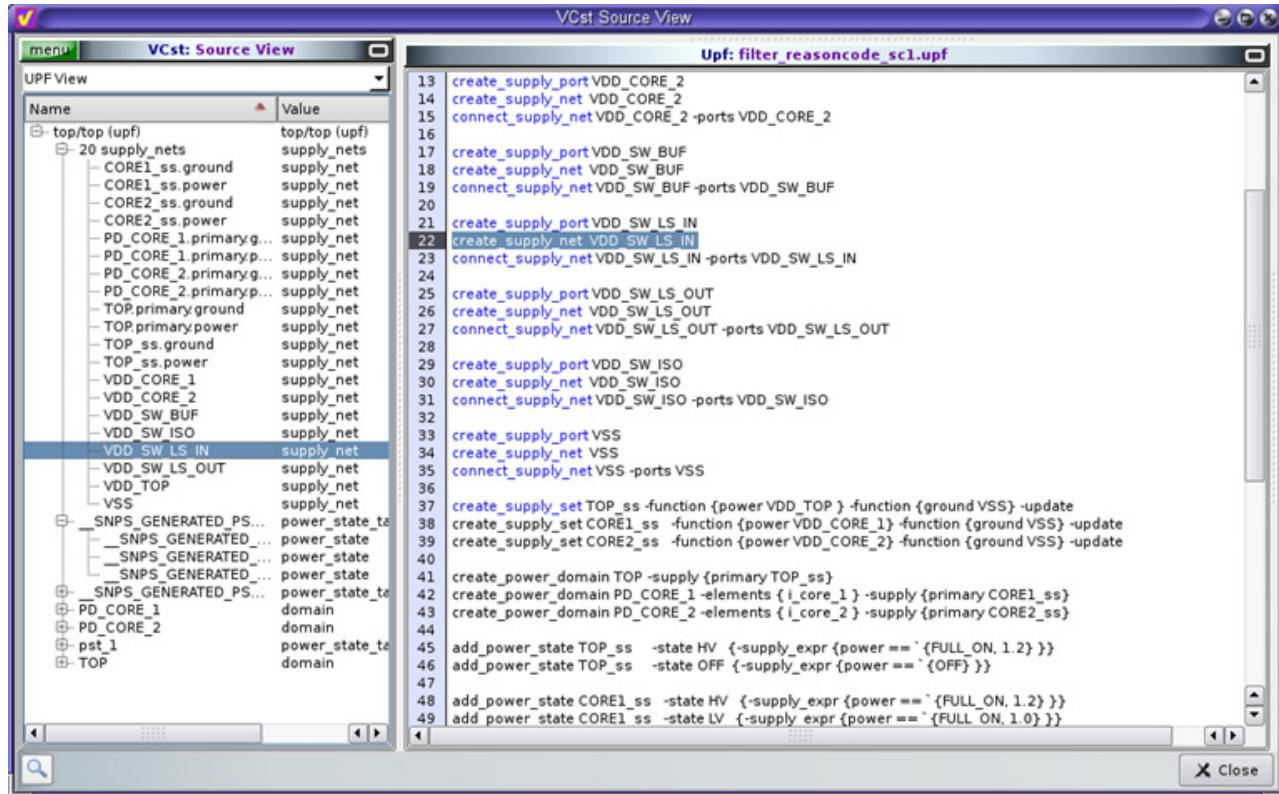
[ -line <string> ] (File name:Line Number)

## Use Model

Here is an example of a UPF view obtained by the following command:

```
%vc_static_shell> view_upf -supply VDD_SW_LS_IN
```

**Figure 4-50 The UPF View**



### 4.6.10.5 Support for Multiple Waiver Files

VC LP had introduced multiple waiver file support, and with this support you can maintain more than one waiver file, and can set one of them as default waiver file where the newly created waivers go in by default.

You can add multiple waiver files using GUI or Tcl commands (see section [“Setting Waiver Files and Default Waiver File using Tcl Commands”](#)).

There are two options to set waiver file in GUI:

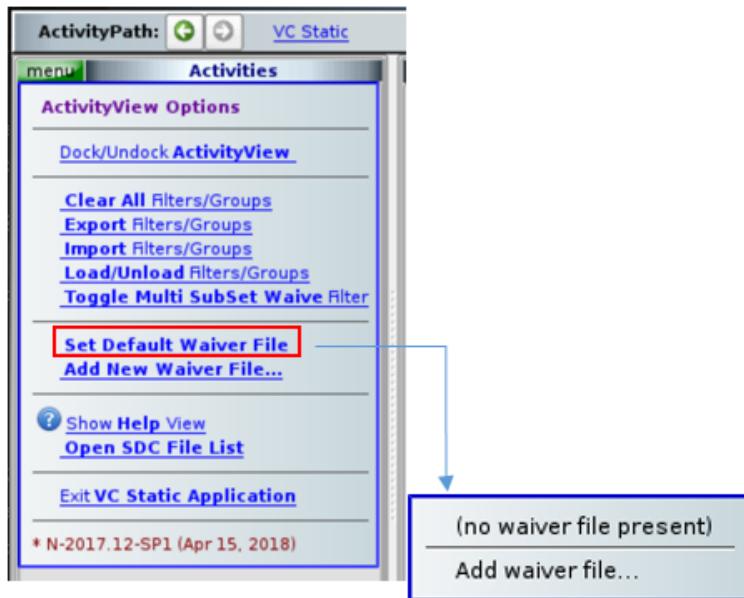
- ❖ “Using the Set Default Waiver File Option”
- ❖ “Using the Add New Waiver File Option”

#### 4.6.10.5.1 Using the Set Default Waiver File Option

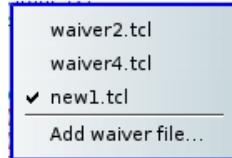
To set default waiver file:

1. Select Menu > Set Default Waiver File.

If there are no existing waiver files, the pop-up menu appears as shown in the following screen.



The popup appears as follows when waiver files are already present, and one of them is selected as 'default'.

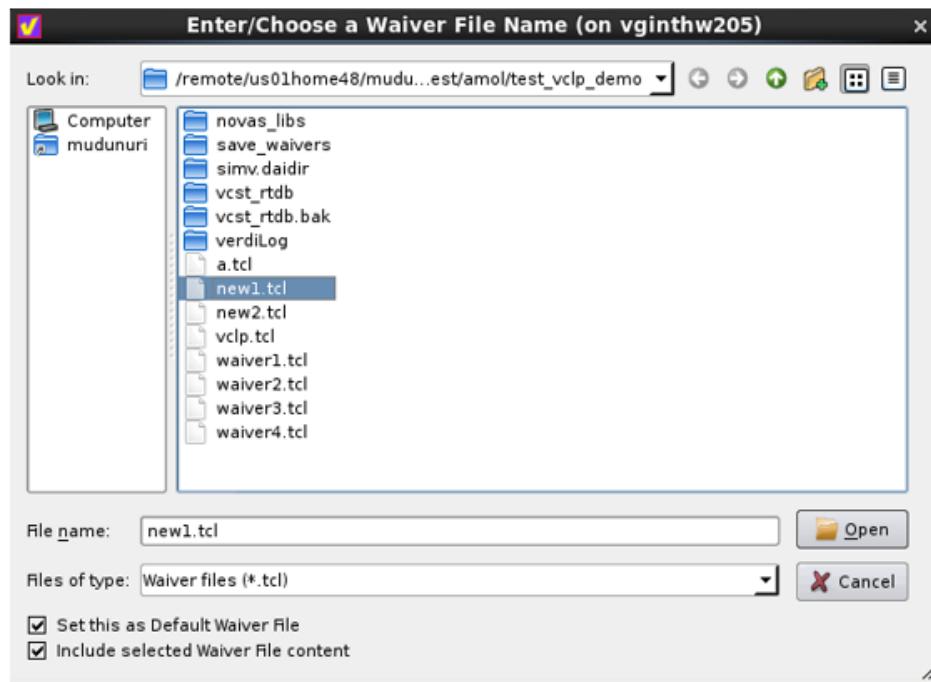


As displayed, the popup menu shows the 'tick mark' corresponding to the current selected default waiver file. To change the default waiver file, select **Menu > Set Default Waiver File** and select on another file name. The 'tick mark' is updated to the selected file, and the setting is applied.

2. If no waiver files are present, click the **Add waiver file** option to add waiver file.

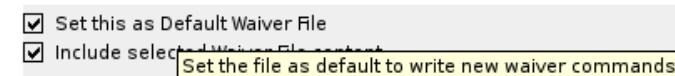
The **Enter/Choose a Waiver File Name** dialog box appears.

You can choose an existing waiver file from the list of available waiver files. If you want to create a new file, type a new name (Ex: my\_file1.tcl) in the 'File name' box.

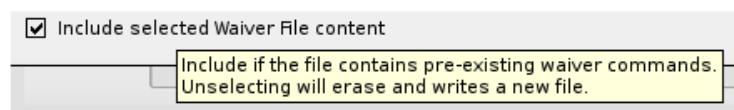


The **Set this as Default Waiver File** and **Include selected Waiver File content** options are selected by default.

- ◆ The **Set this as Default Waiver File** option sets the selected file as default waiver file.



- ◆ The **Include selected Waiver File content** option includes the pre-existing content of the selected waiver file, and processes the commands. When you unselect this option, the content of the file is erased, and re-written with the new waivers added to that file.



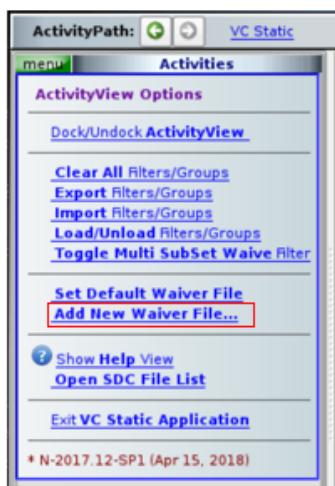
### Note

Unselecting both above check boxes will serve no purpose, and it ignores the selected file. You should have unchecked the **Include selected waiver file content** option when adding empty file or creating a new file.

#### 4.6.10.5.2 Using the Add New Waiver File Option

To set the waiver file:

1. Select Menu > Add New Waiver File.



The Enter/Choose a Waiver File Name dialog box appears. See [Step 2](#) for more details.

#### 4.6.10.5.3 Setting Waiver Files and Default Waiver File using Tcl Commands

Use the `manage_waiver_file` command to manage file based waivers. This command enables you to add waiver files while performing VC LP runs.

##### Syntax

```
%vc_static_shell> manage_waiver_file -help
Usage: manage_waiver_file      # Read waivers from a file
       -add                  (Add waiver file)
       <file_name>           (Waiver file name)
```

##### Note

An error is issued if the file does not exist or is not readable.

You must set the `default_waiver_file` application variable to set a file as the default waiver file.

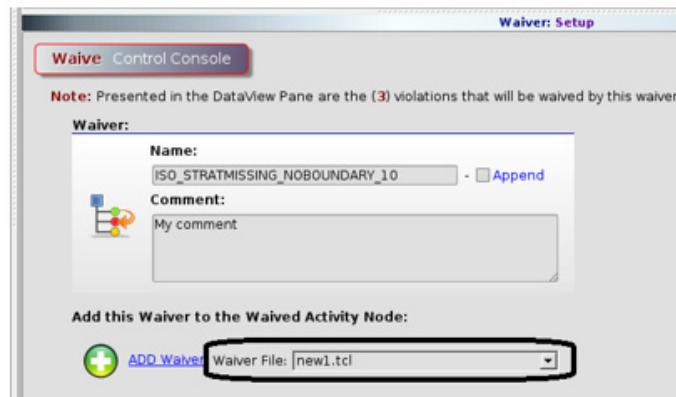
##### Example

Adding the following commands in the Tcl script adds three files, and sets `new1.tcl` as default waiver file.

```
manage_waiver_file -add waiver2.tcl
manage_waiver_file -add waiver4.tcl
manage_waiver_file -add new1.tcl
set_app_var default_waiver_file new1.tcl
```

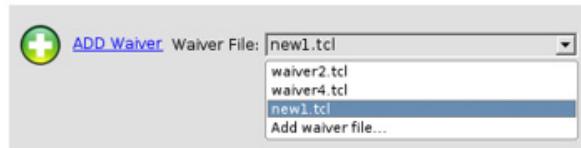
#### 4.6.10.5.4 Selecting the Waiver File While Creating Waivers

You can also select the waiver file while creating waivers. For most waiver creation flows (for example, **Create a waiver** or **create a waiver based on existing filter**), the following waiver setup page is displayed. The waiver File drop-down list is added for such cases.



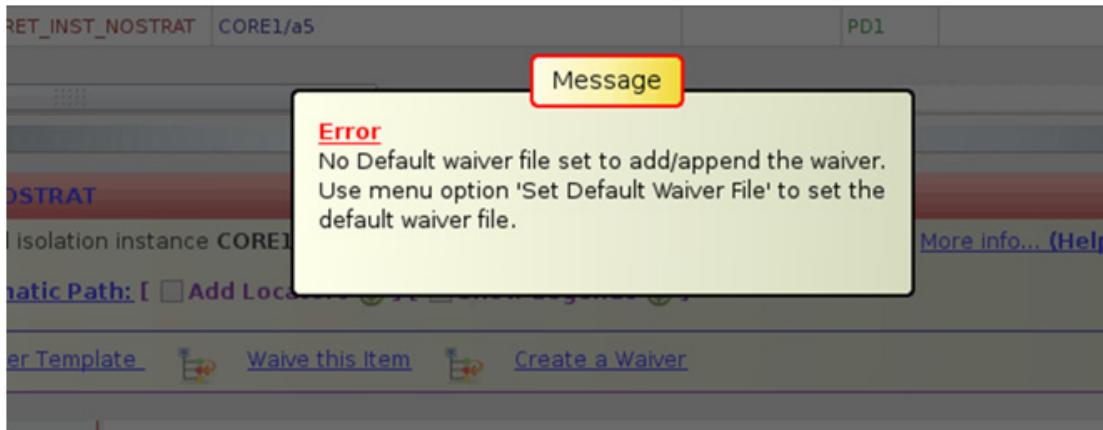
You must select the default waiver file name from the **Waiver File** list to change the current set default waiver file name. If any file is not selected from the **Waiver File**, then waiver is added to the waiver file set as the default waiver file, however, you can click the drop-down list, and can set it to a different listed file.

The **Waiver File** list also shows **Add waiver file** option, which can be used to set additional file directly without going to the main menu.

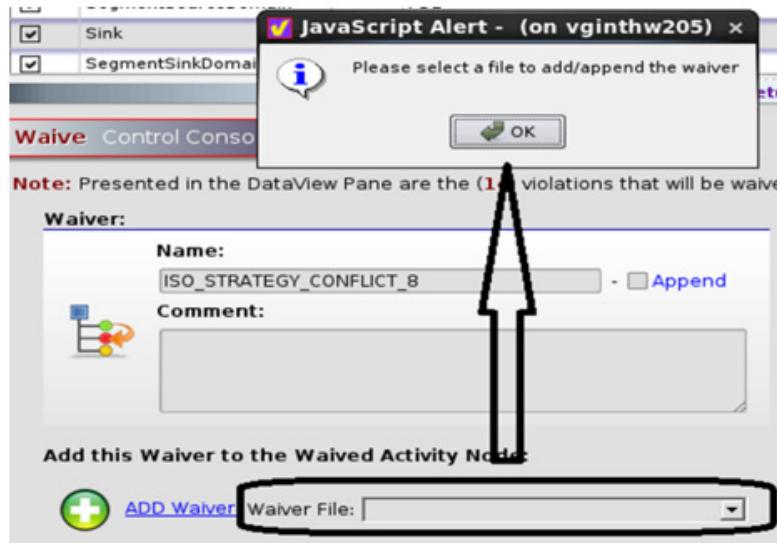


The following messages are reported if you have not setup any waiver file, and try to add waivers in the UI

- ❖ If you are using one click waiver (**Waive this Item**), the following error pops up to set the default waiver file.



- ❖ If you are using the two-click waiver creation (example, 'Create a waiver'), or when you create a filter and then generate a waiver based on that filter, the waiver setup page comes up without any waiver files in corresponding 'Waiver File' drop-down list. If you continue to create waiver by selecting the **ADD Waiver** link, the error message as shown in the following figure appears.



### Note

When you add the waiver file using above methods except unselecting "**Include selected Waiver File content**", the waivers contained in the file are added to the database with the file name details. The waiver file management is applied only to waivers added, changed, or deleted in the `view_activity` GUI. The save and restore flow works with no requirement to repeat any commands.

### Limitations:

- ❖ The opened message count on GUI is not reduced, when you use the `manage_waiver_file` Tcl command.

#### 4.6.10.6 Support for Removing Waivers from the `view_activity`

Previously, you could remove a loaded waiver file using the `manage_waiver_file -remove <waiver_file>` command.

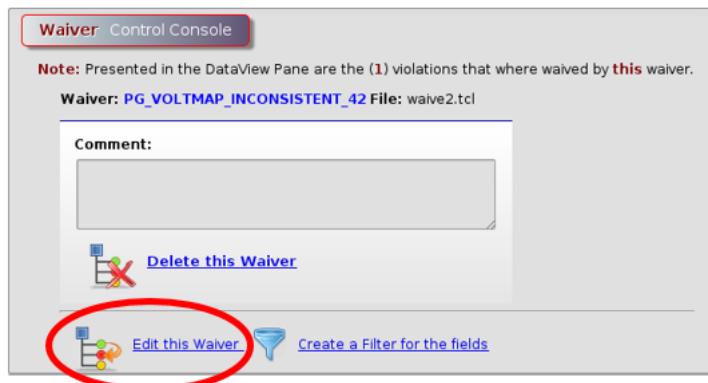
```
manage_waiver_file -remove waiver1.tcl
```

The following option is added to remove a loaded waiver.

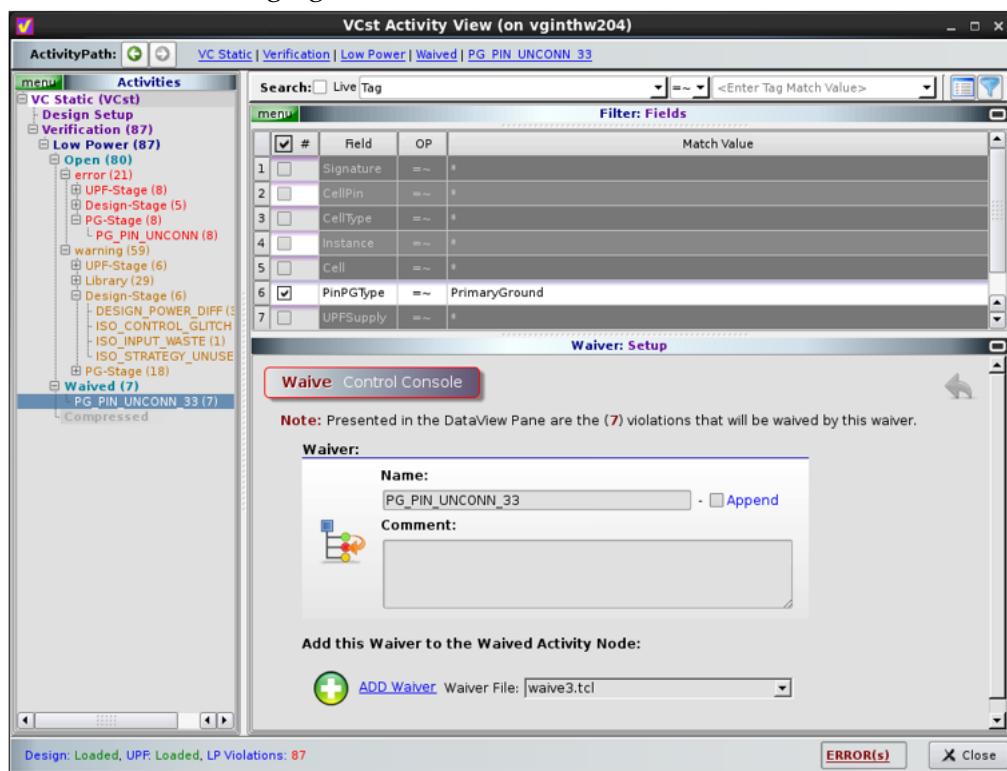
1. Select **Menu->Remove Waiver File...**  
A dialog box appears showing all the existing loaded waiver files.
2. Select the corresponding check box to the file/files to be removed, and click the **Remove** button.

#### 4.6.10.7 Support for Edit this Waiver Option in view\_activity

The Edit this Waiver option is added into Waiver Control Console as shown in the following figure.



When you click the **Edit this Waiver** option, the regular waiver creation screen (where user can select fields) appears as shown in the following figure.



The fields are populated from the existing waiver file. The non-existing fields in the selected waiver will be disabled, and have "\*" as the **Match** value. You can specify Match value to waive particular message sets. After editing, you can click **ADD Waiver** option, the existing waiver file is overwritten. The **Waiver file** is also populated from the existing waiver.

#### 4.6.10.8 Configuring Filter Fields for Tags

. Using the `configure_waiver_filter_field` command, you can customize filter fields for certain tags in waiver window. If this filter command is not set, all fields are enabled in waiver window for each tag.

```
configure_waiver_filter_field -tag <tagname> -field <fieldname> <-enable|-disable>
```

Specify the tag and field names, and use the -enable or -disable options to enable or disable tags.

```
configure_waiver_filter_field -tag ISO_INST_MISSING -field {Source:PinName} -enable
```

All the fields must include subfields till leaf level as below:

```
configure_waiver_filter_field -tag ISO_INST_MISSING -field {SourceInfo:PowerNet:NetName} -enable
```

```
configure_waiver_filter_field -tag ISO_INST_MISSING -field {SourceInfo:PowerNet:NetName} -enable
```

```
configure_waiver_filter_field -tag ISO_INST_MISSING -field {SinkInfo:GroundNet:NetName} -disable
```

```
configure_waiver_filter_field -tag PG_SUPPLY_NOPORT -field {SupplyPort:PortName} -enable
```

You can only see the enabled fields for each tag in waiver window.



#### 4.6.11 Using VC Static Console

VC Static Console enables you view the VC Static output incrementally and read and traverse log files in a readable view. VC Static Console loads a log file with a specific format that shows a structured presentation of the log contents.

VC Static Console offers the following features:

- ❖ Identify and filter out the desired log messages
- ❖ Auto-completion of commands and options
- ❖ Open the Smart Log viewer
- ❖ Open the source of the UPF file loaded in the design

#### 4.6.11.1 Searching with Strings

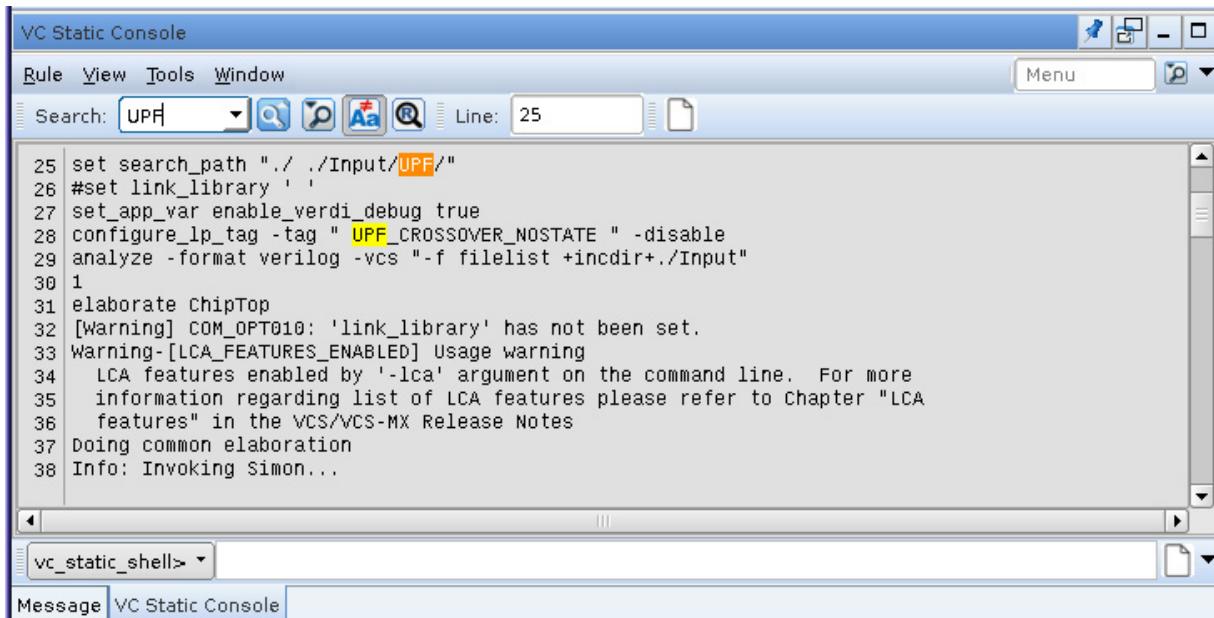
You can search the log file to highlight a string and filter out the blocks that do not contain the matched string.

To search the log file with strings:

- ❖ In the **Search** box, type the string and click  or press **Enter**.

As illustrated in [Figure 4-51](#), only the blocks with messages that meet the searched criteria are displayed.

**Figure 4-51 Search with Strings**



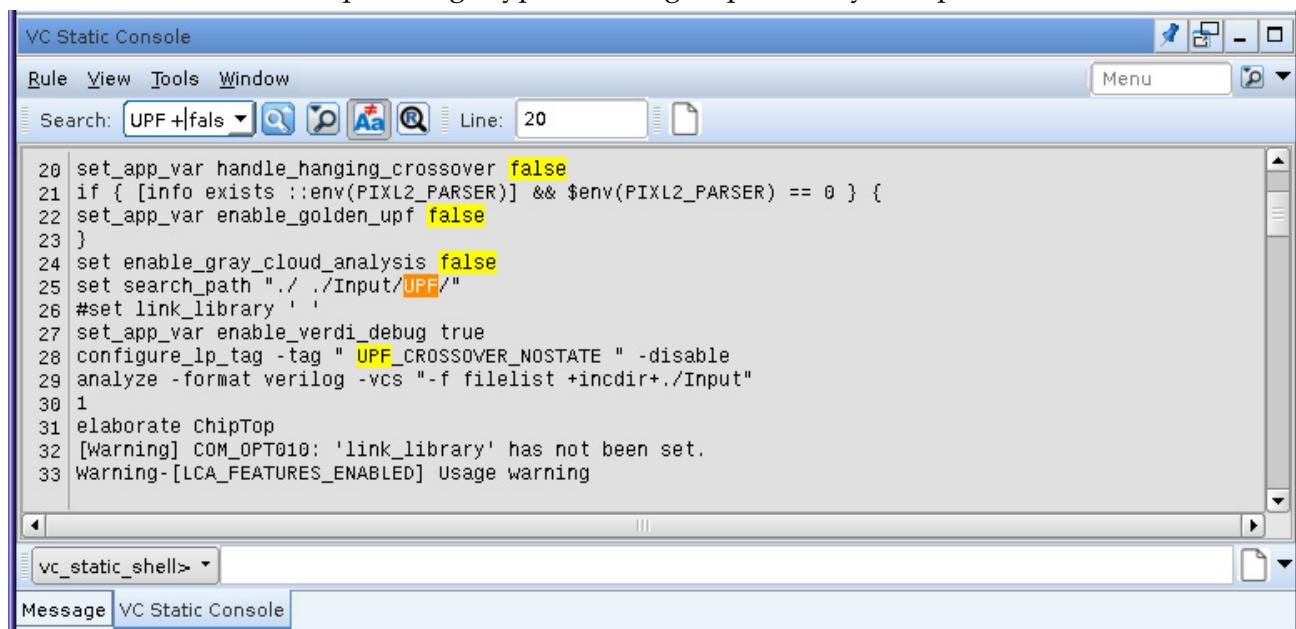
The screenshot shows the VC Static Console interface. The title bar reads "VC Static Console". The menu bar includes "Rule", "View", "Tools", "Window", and "Menu". The toolbar contains icons for search, print, and other functions. A search bar at the top has the text "Search: UPF" and a magnifying glass icon. To the right of the search bar is a "Line: 25" field. The main pane displays a log file with the following content:

```
25 set search_path "./ ./Input/UPF/"
26 #set link_library ''
27 set_app_var enable_verdi_debug true
28 configure_lp_tag -tag " UPF_CROSSOVER_NOSTATE " -disable
29 analyze -format verilog -vcs "-f filelist +incdir+./Input"
30 1
31 elaborate ChipTop
32 [Warning] COM_OPT010: 'link_library' has not been set.
33 Warning-[LCA_FEATURES_ENABLED] Usage warning
34   LCA features enabled by '-lca' argument on the command line. For more
35   information regarding list of LCA features please refer to Chapter "LCA
36   features" in the VCS/VCS-MX Release Notes
37 Doing common elaboration
38 Info: Invoking Simon...
```

Below the main pane, there is a smaller window titled "vc\_static\_shell>". The bottom of the screen shows tabs for "Message" and "VC Static Console", with "Message" currently selected.

- ❖ Continue clicking  or press **Enter** to search the result in another line. Click  to go back to the previously searched result.

- ❖ To search with multiple strings, type the strings separated by a + operator.



The screenshot shows the VC Static Console interface. The main window displays a script or UPF file with several lines highlighted in yellow. The search bar at the top contains the text "UPF +fals". The highlighted lines include:

```
20 set_app_var handle_hanging_crossover false
21 if ( [info exists ::env(PIXL2_PARSER)] && $env(PIXL2_PARSER) == 0 ) {
22 set_app_var enable_golden_upf false
23 }
24 set enable_gray_cloud_analysis false
25 set search_path "./ ./Input/UPF/"
26 #set link_library ''
27 set_app_var enable_verdi_debug true
28 configure_lp_tag -tag " UPF_CROSSOVER_NOSTATE " -disable
29 analyze -format verilog -vcs "-f filelist +incdir+./Input"
30 1
31 elaborate ChipTop
32 [Warning] COM_OPT010: 'link_library' has not been set.
33 Warning-[LCA_FEATURES_ENABLED] Usage warning
```

The status bar at the bottom shows "vc\_static\_shell>" and the tabs "Message" and "VC Static Console" are visible.



If the plus character (+) is the prefix of a string, the string following the + character with the whole pattern is recognized as a key word of the inclusion string.

#### 4.6.11.2 Locating a Specified Line Number

The line numbers are displayed in the VC Static Console.

- ❖ In the **Line** box, specify a line number and press Enter.

The cursor appears at the specified line and highlights the same.

#### 4.6.11.3 Opening the Loaded UPF Files

You can view the source of the UPF files loaded in the design. By default, all the loaded UPF files contains a hyper-link in the VC Static Console. You can click the hyper-link to view the source of the files.

**Figure 4-52 Viewing UPF Source Files**

The screenshot shows the VC Static Console interface with two main panes. The top pane displays UPF source code for 'ChipTop.upf' with syntax highlighting. The bottom pane shows the output of a 'read\_upf' command, detailing simulation statistics and analysis results.

```
*Src1:/slowfs/vgcs15/nikitam/chiptop_rtl/Input/UPF/ChipTop.upf
1 ## CREATE POWER DOMAIS
2 #####
3 create_power_domain TOP
4 create_power_domain MULT -elements Multiplier
5 create_power_domain INST -elements InstDecode
6 create_power_domain GPRS -elements GPRs
7 create_power_domain GENPP -elements Multiplier/GENPP
8
9
10 ## TOPOLEVEL CONNECTIONS
11 #####
12 # VDD
13 create_supply_port VDD
14 create_supply_net VDD -domain TOP
15 create_supply_net VDD -domain GENPP -reuse
connect supply_net vnn -ports vnn

*Src1:ChipTop.upf vcstActivityView

VC Static Console
Rule View Tools Window
Search: Line: 47
47 SIMON CPU Time(s) :0.03
48 SIMON Total Time(s) :0.08
49 Peak Memory(MB) :331
50 =====
51 Info: Simon call complete
52 Info: Exiting after Simon Analysis
53 Info: Simon VCS Finished
54 Verdi KDB elaboration done and the database successfully generated: 0 error(s), 0 warning(s)
55 1
56 read_upf ./Input/UPF/ChipTop.upf
57 GUPF Filename = ./Input/UPF/ChipTop.upf

vc_static_shell>
Message VC Static Console <SmartLog:4>
```

#### 4.6.11.4 Executing Commands from the VC Static Console

You can execute commands from the VC Static Console. The VC Static Console supports auto-completion of command and commands' options.

- ❖ Type the VC Static commands in the **vc\_static\_shell>** box and press Enter, or click the **vc\_static\_shell>** to execute the commands.

Use the **Tab** key to enable auto-completion of the commands and commands' option.

The command string and returned result are displayed in the interactive console frame.

The screenshot shows the VC Static Console window. The main pane displays a command history with several UPF-related commands. Below the main pane, there is a message area labeled 'vc\_static\_shell>' containing 'check\_l'. A dropdown menu is open over this area, showing three options: 'check\_lint' and 'check\_lp'. The status bar at the bottom indicates 'Message VC Static Console'.

```

56 read_upf ./Input/UPF/ChipTop.upf
57 GUUF Filename = ./Input/UPF/ChipTop.upf
58 Supplemental UPF Filename =
59 [Info] UPF_FILE_PARSING: Upf file parsing
60     Parsing top level Upf file 'Input/UPF/ChipTop.upf'.
61 Input/UPF/ChipTop.upf:1: [Info] POWER_TOP_SCOPE: Power Intent top scope resolved
62     'Upf' Top scope resolved to Instance 'ChipTop' of Model 'ChipTop' defined at Input/RTL/ChipTop.v,1.
63 GUUF Parsing complete
64 1
65 set dump_crossover true
66 [Warning] DUMP_CROSSOVER_SET: Application variable dump_crossover has been set to true. Runtime might be increased.
67             The application variable dump_crossover has been set to true. The tool runtime might be increased as this will dump many files for debug purpose
68 true
69 infer_source

```

- ❖ To get the historical typed-in commands, use the up "↑" and down "↓" keys in the command entry.

#### 4.6.11.5 Displaying Search Results

You can get a report of the search results in a separate window. The window only displays the lines in which a matched term is found. You can further filter the results using **Search Result** pane.

- ❖ To open the search results, click .

The Search Results pane appears as shown in [Figure 4-53](#).

**Figure 4-53 Displaying Search Results**

The screenshot shows the VC Static Console window with a search term 'upf' entered in the search bar. The main pane displays the command history. A new 'Search Result' pane is open on the right side, showing a table of search results. The table has two columns: 'Line' and 'Content'. The results listed are:

Line	Content
23	set_app_var enable_golden_upf false
27	set search_path "././Input/UPF/*"
28	./ ./Input/UPF/

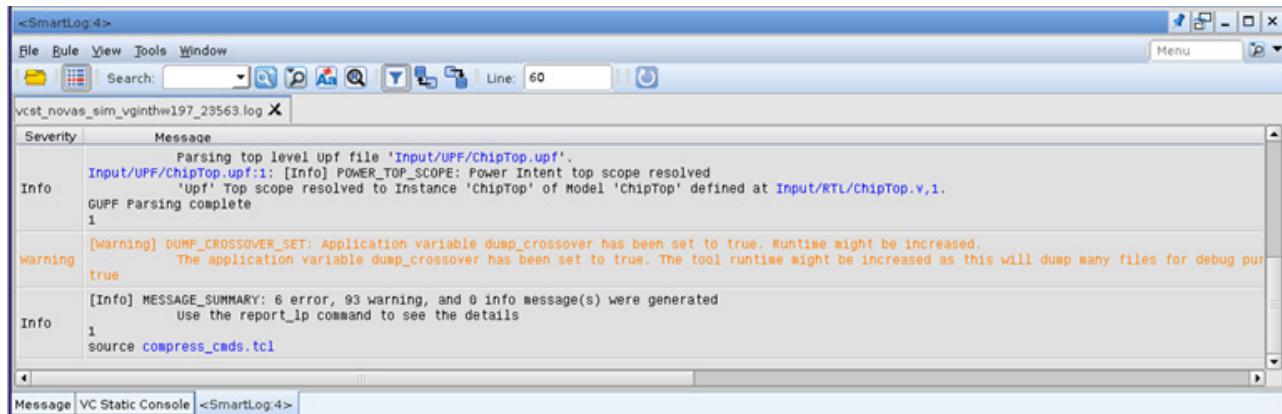
To filter results in the **Search Result** pane:

- ❖ To search for a word in the results, type a word in the **Filter** box.  
The results are filtered for the specified word.
- ❖ To filter for the results in a set of lines, in the **Range(Start-End)** box, type the range.  
The lines in the range that contains the searched word are displayed.

#### 4.6.11.6 Opening Smart Log

You can open the Verdi Smart Log viewer from the VC Static Console.

- ❖ To open the Verdi Smart Log, in the VC Static Console toolbar, click .
- The **SmartLog** window appears as shown in [Figure 4-54](#).

**Figure 4-54 Verdi SmartLog**

For more details on how to use the Verdi Smart Log, see section *Smart Log Tutorial* in the [Verdi® User Guide and Tutorial](#).



# 5 Handling Special LP Scenarios

This chapter is organized into the following sections:

- ❖ “[Special VC LP features](#)”
- ❖ “[Advanced Low Power Cells](#)”
- ❖ “[Advanced UPF Variations](#)”

## 5.1 Special VC LP features

### 5.1.1 Signal Corruption Checks

Signal corruption checks are reported on global control signals which typically traverse multiple power domains, in large designs. Clocks, resets and power control signals are termed as global control signals.

The basis for signal corruption checks is that global signals must not be generated and propagated through relatively OFF domains, compared to sinks. They must also not be isolated as this can cause potential functional problems.

Signal corruption checks (architectural violations) are essential, but not sufficient to prove a design bug. Only a directed test for dynamic low power simulation can confirm the problem. VC LP’s static architectural checks help identify potential logic structures that can cause functional problems. The importance of signal corruption checks at netlist and the pg netlist stage is high, as this is where buffers, inverters etcetera, are put on the control signal path for timing and other reasons. This can cause electrical function violations, but not enough simulations are done post RTL to catch these issues.

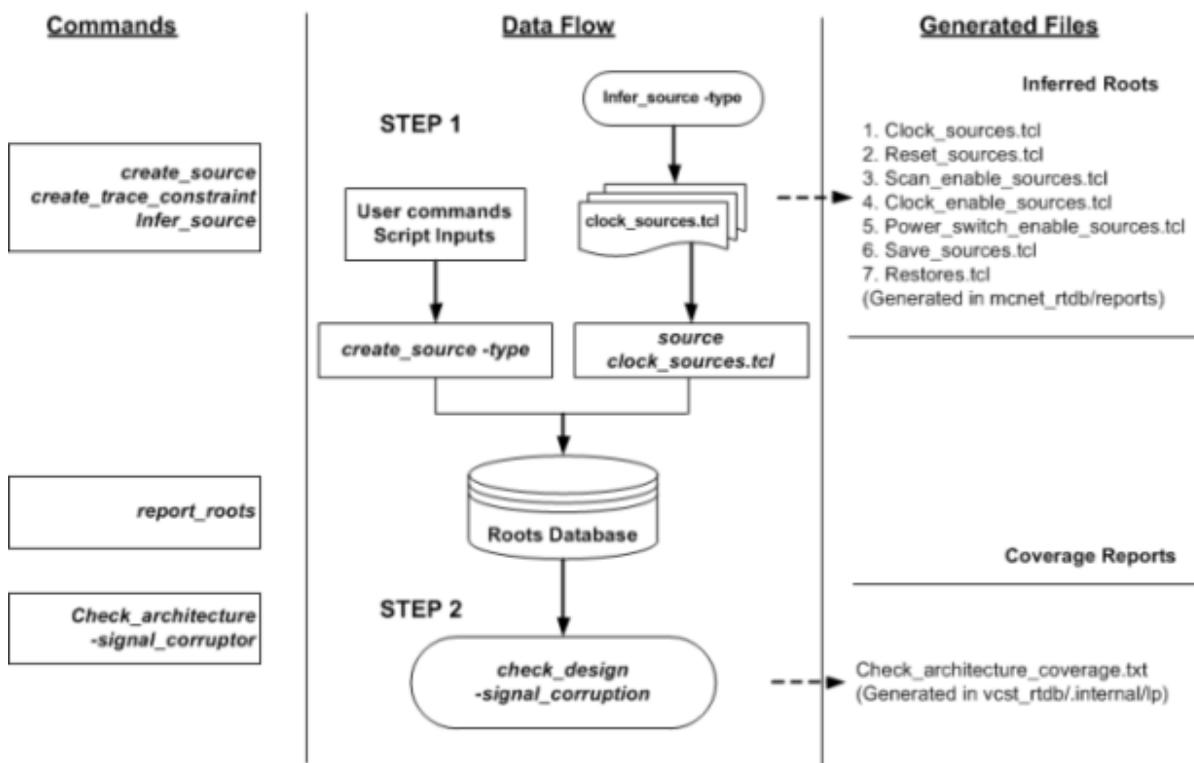
Signal corruption checks require control signal roots to be specified to perform signal corruption analysis. The following types of control signals are supported:

- ❖ Clock
- ❖ Reset
- ❖ Scan Enable
- ❖ Isolation Enable
- ❖ Power Switch Enable
- ❖ Fine-grain Switch Enable
- ❖ Save
- ❖ Restore
- ❖ Clock Enable

The control signals fall into the following broad categories:

- ❖ Explicitly user specified
- ❖ Auto inferred

**Figure 5-1 Signal Corruption Checks Data Flow**



### 5.1.1.1 Specifying the List of Global Signals on Which Signal Corruption Checks must be Performed

- ❖ Explicitly User Specified Roots

You explicitly specify the root of the design on which signal corruption checks must be performed using the `create_source` command.

#### Syntax

```
vc_static_shell> create_source -help
Usage: create_source      # create sources
       -type <str>          (source type:
                           Values: clock, clock_enable,
                           finegrain_switch_enable, iso_enable,
                           power_switch_enable, reset, restore,
                           save, scan_enable, scan_enable_enable,
                           sdc_clock)
       -signal <list>        (signal list)
```

#### Use Model

Creating a top module input as a clock type root.

```
%vc_static_shell> create_source -type clock -signal {CLK}
```

Creating a hierarchical instance, input as a clock type root.

```
%vc_static_shell> create_source -type clock -signal {M_1/CLK}
```

The `-intermediate` option in the `create_source` command enables back trace to the ultimate root, and VC LP takes that ultimate root for architectural checks.

### Example

```
resetA > some logic or buffers > resetB: some logic or buffers > leaf cell
```

When the following Tcl command is used:

```
create_source -type reset -signal resetB
```

VC LP takes `resetB` as startpoint for arch check

```
create_source -type reset -signal resetB -intermediate
```

VC LP back traces from `resetB` to `resetA`, and take `resetA` as start point for arch check. Here, the back trace happens similar to `infer_sorce`. It can back trace through combo logic. The trace is saved in the file `vcst_rtdb/reports/intermediate_source_summary.txt`.

### Notes

By default, VC LP does not trace back to actual source when hierarchical module boundary is provided as argument to the `create_source` command. VC LP creates hierarchical module boundary as the source. The `lp_trace_back_hier_source` application variable enables/disables the actual source tracing, by doing fanin traversal from hierarchical module boundary given in `create_source` command.

When this application variable is set to true, for intermediate ports given in `create_source` command, VC LP traverses backward to identify the actual driver and creates that actual driver as source.

In addition, clock roots can be supplied using the following commands:

- ◆ `create_clock`

```
%vc_static_shell> create_clock -help
Usage: create_clock      # Defines a clock
        -period <period_value>  (Specifies clock period value)
        [-name <clock_name>]    (Specifies clock name)
        [-waveform <edge_list>]
                                (Specifies edge list)
        [-add]                  (Add new options to the original clock)
        [-comment <string>]     (Specifies comment)
        [-refclk]                (Use clock as reference clock [formal])
        [-initial <value>]      (Specify clock initial value (0/1 - default 0)
at time 0 [formal])
        [source_objects]         (nets, ports and pins on which this clock is
defined)
```

- ◆ `create_generated_clock`

```
%vc_static_shell> create_generated_clock -help
Usage: create_generated_clock      # Specifies the relationship between a master
clock and generated clock
        [-name <clock_name>]    (Specifies clock name)
        -source <master_pin>     (Specifies source clock object from which this
clock is derived (port or pin object))
        [-divide_by <divide_factor>]
                                (Specifies frequency division factor (integer))
        [-multiply_by <multiply_factor>]
                                (Specifies frequency multiplication factor
(integer))
```

```

[-duty_cycle <high_pulse_percentage>]
    (Specifies duty cycle (of high pulse width), if
frequency multiplication is used)
    [-edges <edge_list>]   (Specifies a list of positive integers that
represents the edges from the source clock that are to form the edges of the
generated clock)
    [-edge_shift <edge_shift_list>]
        (Specifies a list of floating-point numbers
that represents the amount of shift, in library time units, that the specified
edges are to undergo to yield the final generated clock waveform)
    [-invert]           (Inverts the generated clock signal)
    [-preinvert]        (Creates a generated clock based on the
inverted clock signal)
    [-add]              (Add new options to the original clock)
    [-combinational]    (Combinational signal)
    [-comment <string>] (Specifies comment)
    [-pll_output <output_pin>]
        (Specifies the output pin of the PLL which
is connected to the feedback pin)
    [-pll_feedback <feedback_pin>]
        (Specifies the feedback pain of the PLL)
    [-master_clock <clock_name>]
        (Specifies master clock)
collection_of_objects (Collection of objects)

```

If you have an SDC (industry standard Synopsys design constraints file) for the design, the same can be supplied to VC LP using the `read_sdc` command:

```
%vc_static_shell> read_sdc
```

From the SDC file, VC LP reads only the following commands. The rest of all SDC file is just parsed and ignored.

```

%vc_static_shell> create_clock
%vc_static_shell>create_generated_clock
%vc_static_shell> set_false_path
%vc_static_shell> set_clock_uncertainty
%vc_static_shell> create_clock_group

```



### Note

If the `read_sdc` file has complex Tcl query commands (`get_cells`) used in `create_clock` and so on, it slows down the tool's performance.

#### ❖ Automatically Infer Global Signals

VC LP automatically infers certain types of signal sources. It automatically infers sources for low power control signals, including isolation enable signal, power switch enable signal, and retention register control signals. In addition, you may perform guided inference of clock, reset, clock enable and scan enable signals.

To perform auto inference, use the `infer_source` command.

The `infer_source` command extracts the roots and dumps them into multiple Tcl files consisting of the `create_source` commands in `vcst_rtdb/reports` directory. You can then have the flexibility of editing and sourcing the file.

```

%vc_static_shell> infer_source -help
Usage: infer_source      # infer sources
       [-type <list of types>]

```

```
(source types:  
Values: clock, clock_enable,  
finegrain_switch_enable, iso_enable,  
power_switch_enable, reset, restore,  
save, scan_enable, scan_enable_enable,  
sdc_clock)  
[-exclude_pin_list_file <pin_list_file>]  
(Specifies exclude pin list file name when type is  
clock_enable)  
[-j <num threads>] (number of threads to be used in parallel mode)
```

By default, VC LP did not trace through macro cells when inferring signal sources using the `infer_source` command. As a result, VC LP does not report CORR\* violations for buffers/inverters which are in between the macro buffers/inverters and the actual signal source.

When the `macro_bufinv_as_stdcells` application variable set to true, VC LP traces through macro buffers/inverters when generating signal sources, and VC LP reports the CORR\* violations for buffers/inverters which are in between the macro buffers/inverters and the actual signal source.

By default, the `infer_source` command does not traverse through the clock pins which have `isolation_data_pin` attribute. The signal corruption violations are not reported for paths which are terminating at clock pins having `isolation_data_pin` attribute. In addition to such pins, signal corruption checks and `infer_source` traversal is skipped for paths ending at clock and reset pins of zero pin retention cells.

By default, the `infer_source` command does not trace through the mux select line. When the `lp_infer_source_through_mux_select` application variable is set to true, the `infer_source` traces back through the mux select line irrespective of any other application variable setting.

### Note

This support is limited to macro cells which qualifies as an buffer or inverter.

By default, the `macro_bufinv_as_stdcells` application variable is set to false.

- ❖ You can infer sources in a multi-threaded mode for all the types of signals (`clock`, `clock_enable`, `finegrain_switch_enable`, `iso_enable`, `power_switch_enable`, `reset`, `restore`, `save`, `scan_enable`, `scan_enable_enable`, `sdc_clock`).

Use the `infer_source -j <num_thread>` option to infer sources parallel.

### Example

```
infer_source -j 8
```

In the multi-threaded mode, the performance in terms of the run time is improved as compared to the serial run.

### Note

You must set the `lp_enable_constant_propagation` application variable to true in serial flow to match the QOR.

#### 5.1.1.2 Creating Root for Guided Type Signals

VC LP uses design cell attributes (db cell pin attributes) to backward trace the fanin cone to extract the actual source of the control signals for guided type signals.

For example, in tracing clock type root, VC LP finds all the cells that have clock type attributes and locates their clock pins. Then VC LP traces backwards through all logic to find a source. It uses the information about logic function from the library files to trace backwards through the combinational gates. Tracing stops

at a primary input, black box output or sequential output such as the Q pin of a register. The final list of sources represents the union of all such unique stopping points.

### 5.1.1.3 Creating Root for UPF Automatic Infer Type Signals

VC LP uses UPF (specified control signals) to backward trace the fanin cone to extract the actual source of the control signals for guided type signals. In this case, the UPF must contain isolation, power switch and retention commands in it.

For example, in tracing isolation enable type root, VC LP finds all isolation enable signal details from the isolation strategy mentioned in the UPF. Then, VC LP finds all the cells that have isolation enable type attribute and locates their isolation enable pins that are connected to that UPF mentioned isolation signals. VC LP then traces backwards through all logic to find a source. It uses the information about timing arcs from the library files to trace backwards through combinational gates. Tracing stops at a primary input, black box output, or sequential output such as the Q pin of a register. The final list of sources represents the union of all such unique stopping points.

Another example is the isolation cells present in the design. However, the UPF mentioned isolation enable signal does not connect to the isolation enable pin in design. Hence, VC LP does not create any automatic UPF root.



#### Note

A rerun of the infer\_source command with the same type overwrites existing files.

### 5.1.1.3.1 Support for timing\_arc\_function Application variable

Some customers have special cells which has no function attribute in one or more of the cell output pins, but have timing arc definitions. For such cells, VC LP derives connectivity using the timing arc functions. This support is disabled by default. To enable this support, set the timing\_arc\_function application variable to true.

Under the application variable, connectivity checks and corruption checks are triggered based on the relations derived from the timing arcs. The polarity values of the signal paths propagating through the cells using timing arc functions are reflected in VC LP violations (for example, ISO\_CONTROL\_INVERT).

When this application variable is set to true, the Tcl commands (for example, all\_fanin, all\_fanout and trace path commands) trace through cells using the timing arc information.

#### Notes:

- ❖ The timing\_arc\_function application variable does not overwrite existing function attributes.
- ❖ The timing\_arc\_function application variable suppresses the functionality of the synth\_all\_macro\_pins\_with\_func application variable. Hence, these application variables must not be used together.

### 5.1.1.4 Reviewing Inferred Sources

Clock tracing, in particular, can be very complex. VC LP does not automatically use the result of its clock tracing. When you use guided tracing, you should review the results in the report file. If the list of the inferred sources is correct, you can use the Tcl source command to read it into VC LP. If the list is almost correct, you may copy the file to your own directory (so it is not written over by the next run), edit the file, and then source your edited file in future VC LP runs.

VC LP uses information about timing arcs to trace backwards through the gates. In some cases, the timing arc information in the library may be missing or incorrect. To correct this, use the create\_trace\_constraint command. This has many options; but its primary use is to create a temporary timing arc. Here is an example of using the command:

```
create_trace_constraint -trace_through \
signal_type clock -module BBOX -input_port I \
output_port ZN
```

This indicates that when tracing clocks, a temporary timing arc must be used to trace between the indicated input and output ports of the module. In case the library has macros or other complex cells with no function definition, this command can be used to bridge the gap when inferring sources.

### 5.1.1.5 Removing Root

To remove the user specified root, use the `remove_source` command.

#### Syntax

```
%vc_static_shell> remove_source -help
Usage: remove_source      # remove source
      [-type <str>]          (match type:
                                Values: clock, clock_enable,
                                finegrain_switch_enable, iso_enable,
                                power_switch_enable, reset, restore,
                                save, scan_enable, scan_enable_enable,
                                sdc_clock)
      [-signal <signal list>]  (signal list)
```

### 5.1.1.6 Tracing Constraints

Tracing stops at a primary input, black box output or sequential output such as the Q pin of a register when the automatic infer source command is used. If you want to bypass such elements and find the next global source, use the `create_trace_constraint` command.

#### Syntax

```
%vc_static_shell> create_trace_constraint -help
Usage: create_trace_constraint      # create trace constraint that tracing engine should
honor
      [-halt_at]          (halt at)
      [-trace_through]    (trace through)
      [-signal_type <signal_type>]
                            (signal type:
                            Values: clock, iso_enable, power_ground,
                            reset, save_restore, scan_en,
                            switch_enable)
      [-module <str>]       (module name)
      [-instance <str>]     (instance name)
      [-cell_type <cell_type>]
                            (cell type:
                            Values: always_on, buffer, coarse_grain,
                            complex_combo, inverter, is_a_flip_flop,
                            is_a_latch, is_clock_gating_cell,
                            is_isolation, is_level_shifter, is_pad,
                            multiplexer, simple_combo)
      [-input_port <str>]    (input port)
      [-output_port <str>]   (output port)
      [-function <str>]     (function)
      [-port <str>]         (port name)
      [-port_type <str>]    (port type:
```

Values: clock, control, data, power,  
preset, reset, scan\_in, select)



**Note**  
The tracing constraints specified for the signal type switch\_enable are applied for power\_switch\_enable as well as finegrain\_switch\_enable.

Use the `create_trace_constraint` command for the following cases:

1. Applying constraint on module.

```
create_trace_constraint -trace_through \
    signal_type clock -module BBOX -input_port I \
    output_port ZN
```

In this case, constraint is applied in all the black box module instances.

2. Applying constraint on specific instance.

```
create_trace_constraint -trace_through \
    signal_type clock -instance BBOX_inst -input_port I \
    output_port ZN
```

In this case, the constraint is applied only to the BBOX\_inst instance.

3. To halt tracing

```
create_trace_constraint -halt_at \
    signal_type clock -module BBOX -input_port I
```

In this case, back-tracing stops at I, input of a black box module.

#### 5.1.1.6.1 Placing of a Constraint

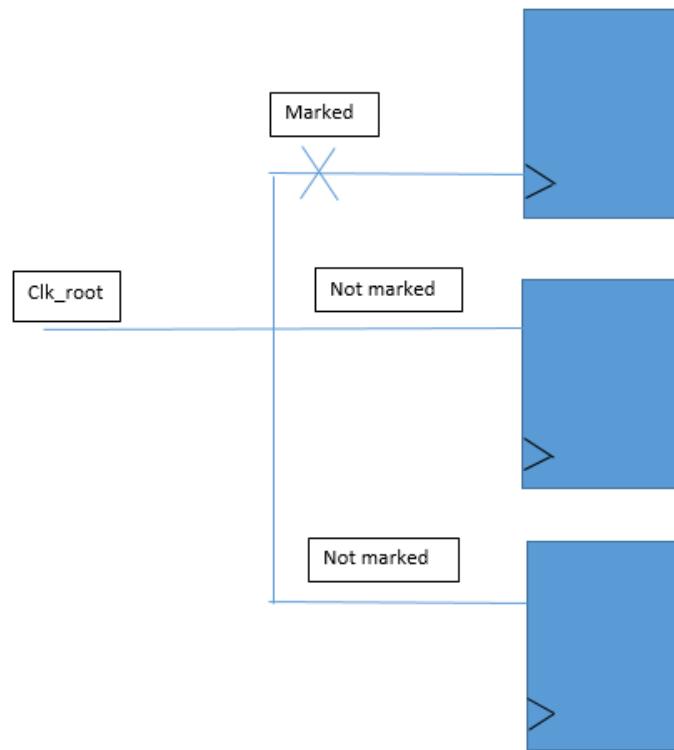
You must write a constraint after the UPF and design compilation and before creating a critical signal source or inferring a critical signal source.

#### 5.1.1.7 Support for the enable\_arch\_check\_optimization Application Variable

While performing architecture checks, VC LP identifies the `clock_enable` and `scan_enable` roots and traces these paths. These paths might contain logic circuit that are not meaningful for architectural checks. You can optimize the architectural checks by setting `enable_arch_check_optimization` application variable to true. When you set this application variable to true, VC LP stops tracing `scan_enable` and `clock_enable` roots that might not be meaningful for architectural checks.

For example, while performing the `infer_source` fanin tracing from `clock_enable` and `scan_enable` pins, when this application variable is set to true, all the input pins are not marked for tracing for a multi fanout net as shown in [Figure 5-2](#). When architecture checks are run for these `clock_enable` and `scan_enable` roots, VC LP traces those fanouts pins that are marked. As the number of marked pins are lesser, this improves the performance for architecture checks.

When this application variable is set to false, in case of a multi fanout net, all input pins are traced resulting in lesser performance benefits for `infer_source` and architecture checks, when compared to the performance benefits achieved when this application variable is set to true.

**Figure 5-2 Example of the enable\_arch\_check\_optimization Application Variable**

### 5.1.1.8 Running Signal Corruption Checks

The following application variable must be enabled to work with the architecture flow:

- ❖ Set the ignore\_reset\_reaching\_dpin\_corruption application variable to false [Default set in Tool]

If you do not want to perform signal corruption checks upon reaching D-pin of a flip-flop while traversing from a reset root, then set this variable to true.

The check command is used to invoke signal corruption checks.

- ❖ Use the lp\_ignore\_signal\_corruption command to ignore signal corruption checks for user provided nodes.

#### Syntax

```
vc_static_shell> lp_ignore_signal_corruption -help
Usage: lp_ignore_signal_corruption    # Ignore signal corruption checks
      -type <str>                  (source type:
                                      Values: *, clock, clock_enable,
                                              finegrain_switch_enable, iso_enable,
                                              power_switch_enable, reset, restore,
                                              save, scan_enable, scan_enable_enable,
                                              sdc_clock)
      [-from <signal>]             (Starting node of a path)
      [-to <signal>]               (End node of a path)
      [-through <signal>]          (Node a path goes through)
      [-regexp]                    (Patterns are regular expressions)
```

<code>[-verbose]</code>	(Reports wildcard expansions)
<code>[-force]</code>	(Force wildcard expansions)
<code>[-hierarchical]</code>	(hierarchical wildcard expansions)

### Note

When there is more than 1 possibility (to/through/from) available for the `lp_ignore_signal_corruption` command, use the `-force` option. The command can handle 100K possibilities. For example, `lp_ignore_signal_corruption -through /static// -to /static//*` can have 100 possible instances. `lp_ignore_signal_corruption -to /static*` has 1000 possible instances. Number of possibilities  $1000 \times 100 = 100K$ . This limitation is only if a command has more than one option (to/through/from) specified.

- ❖ Signal Corruption checks get enabled when you run either `check_lp -stage design` or `check_lp -stage design -signalcorruption` command.
- VC LP reports the following two violation messages when it finds architecture issues in the design:
- ◆ CORR\_CONTROL\_STATE
  - ◆ CORR\_CONTROL\_ISO
- ❖ When you set the `annotate_iso_in_corr_control_state` application variable to true, the CORR\_CONTROL\_ISO tag is not reported. Instead, the CORR\_CONTROL\_STATE\_WITHISO and CORR\_CONTROL\_STATE\_NOISO violations are reported. The default value of this application variable is false.

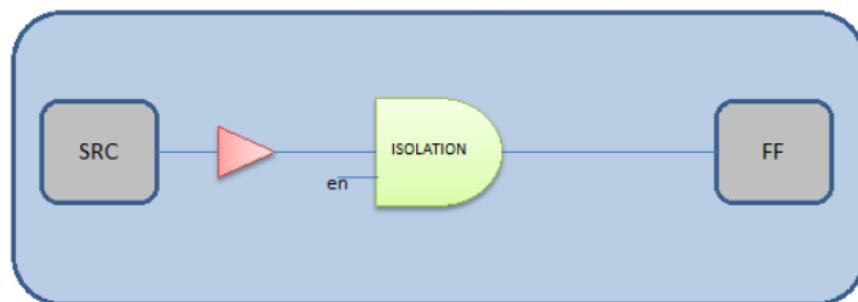
### Note

The `annotate_iso_in_corr_control_state` application variable support is a custom feature available with beta support.

- ◆ CORR\_CONTROL\_STATE\_WITHISO

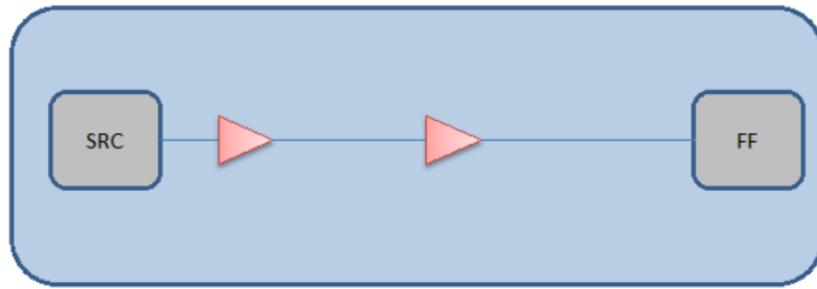
The signal passes through corrupting objects, but also passes through an isolation cell which guards the corrupting objects.

**Figure 5-3 CORR\_CONTROL\_STATE\_WITHISO**



- ◆ CORR\_CONTROL\_STATE\_NOISO

The signal passes through corrupting objects, but there is no isolation cell to guard the corrupting objects.

**Figure 5-4 CORR\_CONTROL\_STATE\_NOISO**

### 5.1.1.9 Support for Parallel Signal Corruption Checks

You can perform parallel signal corruption checks in the design stage. To enable parallel signal corruption checks, use the `-j <num-threads>` in the `check_lp -stage design -family signalcorruption` command. Signal corruption checks have been parallelized for all types of control signals.

```
check_lp -stage design -family signalcorruption -j 8
```



#### Note

It is recommended to always use the `check_lp -stage design -j <num-threads>` command with `-family signalcorruption`.



#### Note

You must set the `lp_enable_constant_propagation` application variable to true in serial flow to match the QOR.

### 5.1.1.10 Reviewing Results and Coverage Report

VC LP creates a coverage report when you run the `check_lp -stage design -signalcorruption` command. It generates the `check_architecture_coverage.txt` file in the `vcst_rtbd/reports` area.

```
Clock
-----
Total number of roots      : 2799
Total number of pins        : 9031671
Total number of pins covered: 9031671
```

Here,

- ❖ The total number of roots indicate the total number of roots created using either user specification or automatic inferring.
- ❖ Total number of pins indicate the number of design cells that have this type of attributes on a pin.
- ❖ Total number of pins covered indicate the number of pins covered by the roots created.

### 5.1.1.11 Debugging/Fixing Signal Corruption Checks

Placing protection logic (ISO/LS) may not always be functionally correct. For this purpose, VC LP reports architectural checks. In some designs/power architectures, you can have isolation gates on global signals. Their functional impact is analyzed and taken care of.

In such cases, you do not need to run architectural checks in VC LP on those signals. Alternately, you can simply ignore the reported violations.

After reviewing the global signals supplied for checks, identify global signals for which the signal corruption violations are reported. Check whether they are:

- ❖ Functional paths
- ❖ Non-functional paths

Functional paths are those that have to be kept alive to be able to propagate the signal from source/root to the sinks. These paths are typically buffered with always-on buffers and no isolation cells are used.

Non-functional paths are those that need NOT carry the signal from source to sink. On these paths, it is OK to use normal (shutdown) buffers and isolation to avoid electrical issues.

Based on the architectural intent of the design, review the signal corruption violations in the above context, confirm through simulations, and fix or waive the violations as required. You may have to revisit the global paths and check if they can be routed differently to avoid crossovers and thereby implement protection logic on its path.

#### 5.1.1.12 Checking Progress of Architectural Checks

When you perform architectural checks on large designs in VC LP, the architecture checks traversal takes too much time, and some times the design may hang or get stuck at some traversal. Using the `enable_arch_check_process` application variable, you can check whether the design is still being checked, or if it is hanging or stuck.

The `enable_arch_check_process` is disabled by default. You can enable this application variable by setting it to a positive integer value, that is, any value greater than 0.

```
%vc_static_shell> set_app_var enable_arch_check_progress N
```

Where N is a positive integer value greater than 0, which indicates that the status of the run must be displayed after the `check_lp` command traverses N number of paths.

For example,

```
%vc_static_shell> set_app_var enable_arch_check_progress 500
```

When this application variable is set to 500, the results are displayed after every 500 paths are traversed. You can know the status of run by viewing the results displayed in the output.

The following is an example output of the parsing messages indicating the progress that the check is ongoing, not hanging.

```
check_lp -stage design
[Info] SIG_CORR_CHECK_RUN: Running Signal corruption checks for '4' 'Clock' roots ...
Doing signal corruption check for 4 clock roots...
Doing signal corruption check for clock "Root:TST_tclk"...
clock reached / path traced = 428/448
Doing signal corruption check for clock "Root:cg_gbl_sclk"...
clock reached / path traced = 39/500
clock reached / path traced = 39/1000
clock reached / path traced = 39/1500
clock reached / path traced = 39/2000
clock reached / path traced = 39/2500
clock reached / path traced = 39/3000
clock reached / path traced = 39/3500
clock reached / path traced = 39/4000
clock reached / path traced = 39/4500
clock reached / path traced = 39/5000
clock reached / path traced = 39/5500
clock reached / path traced = 39/6000
clock reached / path traced = 189/6500
clock reached / path traced = 689/7000
```

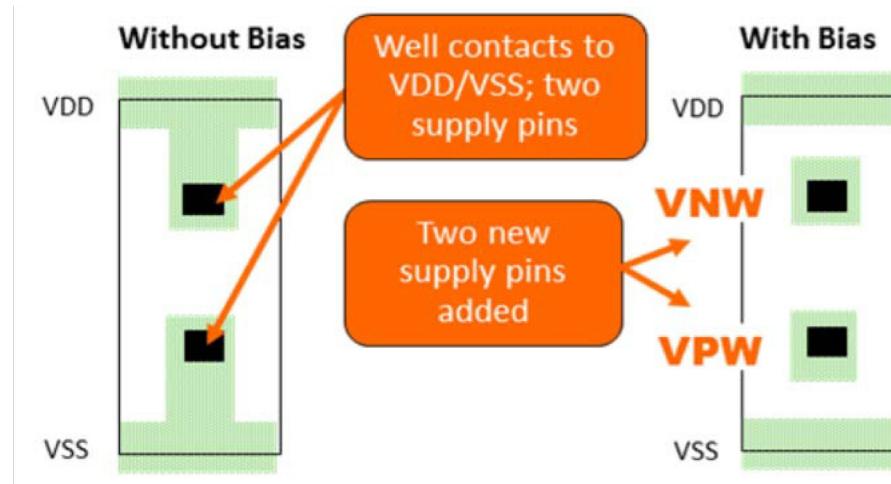
```
clock reached / path traced = 1189/7500
clock reached / path traced = 1689/8000
clock reached / path traced = 1751/8126
```

## 5.1.2 Bias Checks

In modern low power designs, some design teams control the nwell and pwell connections to cells and set them to voltages which differ from the supply rails. This can be done to improve speed or reduce leakage, depending on the voltage differences. However, if there are power states where the bias or supply rail is floating and the other is driven, the cell can have an electrical failure. This section describes the checks which VC LP performs to ensure electrical correctness of bias connections.

[Figure 5-5](#) shows the basic layout of a cell with bias connections:

**Figure 5-5 Cell With Bias Connections**



In liberty, extra supply pins are defined with pg\_type : [pn]well. Each normal supply pin is annotated with related\_bias\_pin.

For example:

```
cell (AND2) {
    pg_pin (VDD) {
        pg_type : primary_power;
        related_bias_pin : "VNW";
    }
    pg_pin (VNW) {
        pg_type : nwell;
    }
    pg_pin (VSS) {
        pg_type : primary_ground;
        related_bias_pin : "VPW";
    }
    pg_pin (VPW) {
        pg_type : pwell;
    }
}
```

### 5.1.2.1 Support for Negative Voltage for Bias Checks

VC LP supports negative supply for bias pins. If a negative voltage is associated with the add\_power\_state or add\_port\_state command for a bias network, VC LP parses the UPF without reporting any error or warning messages.

#### Example

```
create_supply_set SS_TOP -function {pwell vdd_pwell} -function {nwell vdd_nwell}
add_port_state vdd_pwell -state {vdd_pwell_01 -0.1}
add_port_state vdd_nwell -state {vdd_nwell_01 -1.0}
```

If a negative voltage is associated with a non-bias network (power/ground) as shown in the example below, then the UPF\_NONBIAS\_NEGATIVE\_VOLTAGE error message is reported.

#### Example

```
add_power_state TOP.primary -state HV {-supply_expr {{power == `FULL_ON, -1.08}}}
add_power_state TOP.primary -state LV {-supply_expr {ground == `FULL_ON, 0.0}}
```

[Error] UPF\_NONBIAS\_NEGATIVE\_VOLTAGE: Negative voltage supply to a non-bias network

*State 'HV' associated with a non-bias network has a negative voltage '{ FULL\_ON -1.08 }' which is illegal.*

Error: 0

*Use error\_info for more info. (CMD-013)*

### 5.1.2.2 Support for UPF\_STATEVOLTAGE\_TRIPLET Parser Warning Message

The UPF\_STATEVOLTAGE\_TRIPLET parser warning message is reported when the voltage value of supply state (not off) defined in the add\_port\_state or add\_power\_state UPF command is not written with triplet value format.

VC LP does not check for the UPF\_STATEVOLTAGE\_TRIPLET violation on the ground supply. The UPF\_STATEVOLTAGE\_TRIPLET violation reports the supply information.

The UPF\_STATEVOLTAGE\_TRIPLET message is disabled by default. To enable it, use the configure\_read\_tag -family UPF -tag UPF\_STATEVOLTAGE\_TRIPLET -enable command. Once it is enabled, the violation is reported under the report\_read\_violations command.

### 5.1.2.3 Support for lp\_upf\_disallow\_sdsn\_in\_bias Application Variable

With the lp\_upf\_disallow\_sdsn\_in\_bias application variable set to true, when set\_domain\_supply\_net set on a bias block (with SDA -enable\_bias true), VC LP reports the UPF\_COMMAND\_NOT\_ALLOWED\_IN\_BIAS\_BLOCK warning to resemble cross tool behavior

### 5.1.2.4 Missing Bias Specification

If the bias specification (Pwell or Nwell) is missing for the power domains present in the UPF then UPF\_BIAS\_MISSING is flagged.

This violation is disabled by default. You can enable it using the configure\_tag -app LP -tag UPF\_BIAS\_MISSING -enable command.

Tag	:	UPF_BIAS_MISSING
Description	:	Bias net for [PinPGType] not defined for domain
[Domain]		
PinPGType	:	NWell
Domain	:	PD2

### 5.1.2.5 Defining Bias Specification

You can define bias specifications using the following commands:

- ❖ `create_supply_set`
- ❖ `set_back_bias`
- ❖ `connect_supply_net`

#### 5.1.2.5.1 Defining Bias Specification Using the `create_supply_set` Command

In UPF 2.0, the following is the syntax to define bias nets as a part of supply sets.

```
create_supply_set ss_top -function {power VDDtop} -function {ground VSStop} -function
{nwell VDDtop_s} -function {pwell VSStop}
create_power_domain D1 -supply { primary ss_top }
```

In Verilog, bias pins are connected just like any other pins. However, it is very common for a PG netlist to be generated with no bias pin information. Bias pins are often assumed to be connected at the device level and the information may not be written into a PG netlist. In case the library has bias pin information but the PG netlist intentionally does not, you may need to run the `check_lp -stage pg` command with bias checking disabled to avoid a message about each instance bias pin being unconnected.

#### 5.1.2.5.2 Defining Bias Specification Using the `set_back_bias` Command

Typically PG connectivity information as specification (`connect_supply_net`) is not available in UPF for back bias pins. Though bias connectivity can be specified using supply set specification, VC LP also enables you to describe the same using the `set_back_bias` command. The `set_back_bias` specification is defined on a per-domain basis. It describes how bias pins are expected to be connected in the PG netlist. The `set_back_bias` commands can also be used to define domain level bias specification.

#### Syntax

```
%vc_static_shell> set_back_bias -help
Usage: set_back_bias      # Sets back bias nets for a Power Domain
       -domain <domain>          (Specify the domain to which the bias net should get
associated)
       -type <type>            (Specify the pg type of the bias pin to which the supply
net will get associated: nwell or pwell:
                           Values: nwell, pwell)
       -net <net list>          (specify the supply net defined in UPF that connect to
bias pg pin)
```

#### Use Model 1:

```
set_back_bias -domain TOP -type pwell -net VPW
set_back_bias -domain TOP -type nwell -net VNW
```

The above example means that for any cell in the power domain TOP, any pin with 'pwell' description should be connected to the supply Net VPW at the design top level and any pin with 'nwell' description should be connected to the supply Net VNW at the design top level.

#### Use Model 2:

The following is an example of the `set_back_bias` command used for the hierarchical domains.

```
set_back_bias -domain CORE1/PD1 -type pwell -net CORE1/VPW
set_back_bias -domain CORE1/PD1 -type nwell -net CORE1/VNW
```

The above example means that for any cell in the hierarchical power domain CORE1/PD1, any pin with 'pwell' description should be connected to the supply Net CORE1/VNW at the design level and any pin with 'nwell' description should be connected to the supply Net CORE1/VNW at the design level.

### Use Model 3:

The following is an example of the `set_back_bias` command used for multiple supply nets.

```
set_back_bias -domain TOP -type pwell -net "VPW VPW_BACK"
set_back_bias -domain TOP -type nwell -net "VNW VNW_BACK"
```

The above example means that for any cell in the power domain TOP, any pin with 'pwell' description should be connected to the supply Net VPW or VPW\_BACK at the design top level and any pin with 'nwell' description should be connected to the supply Net VNW or VNW\_BACK at the design top level.

#### 5.1.2.5.3 Defining Bias Specification Using `connect_supply_net`

Bias pins connection can be directly done using the `connect_supply_net` UPF command.

In the following example, VNW and VPW of the CORE1/ff\_i1 instance is connected to the VNW and VPW supply nets respectively.

```
connect_supply_net VNW -ports ff_i1/VNW
connect_supply_net VPW -ports ff_i1/VPW
```

The `connect_supply_net` command can also be used to define cell/instance level bias specification.

#### 5.1.2.6 VC LP Checks for Bias Connections

VC LP flags violations depending on the power states of the power and ground pins and their discovered bias pins.

When power state checking is performed on the UPF, certain pairs of bias nets are found. For each nwell net mentioned in a supply set, a pair is formed with its primary power net. For each pwell net mentioned in a supply set, a pair is formed with its primary ground net. For each pair, a check of the power states is needed. The following violations are reported at the `check_lp -stage upf -family powerstatetable` stage.

- ❖ PST\_BIAS\_NOSTATE (error)
- ❖ PST\_BIASOFF\_STATE (error)
- ❖ PST\_BIAS\_INSUFFICIENT (error)
- ❖ PST\_BIASON\_STATE (warning)
- ❖ PST\_BIAS\_EXCESS (warning)

Many pairs of bias nets are determined by the UPF. However, any cell may introduce a new pair of bias nets due to its connections. This means that during PG checking, new bias pairs may be discovered. Each new pair which is discovered must be checked. The same power state checking is done for discovered bias pairs and normal bias pairs declared in the UPF. However, a new message type is required because the name of one instance which caused the pair to be discovered must be included. The following messages are issued for this type of violation. The following violations are reported at the `check_lp -stage pg -family bias` stage.

- ❖ PG\_BIAS\_OPTIMIZE (warning)
- ❖ PG\_BIAS\_SHORT (error)
- ❖ PG\_BIAS\_NOSTATE (error)
- ❖ PG\_BIASOFF\_STATE (error)
- ❖ PG\_BIAS\_INSUFFICIENT (error)
- ❖ PG\_BIASON\_STATE (warning)
- ❖ PG\_BIAS\_EXCESS (warning)

- ❖ PG\_TIEDTO\_SHORT (error)

The following violations are reported at the `check_lp -stage upf -family bias` stage.

- ❖ UPF\_BIAS\_SHORT (error)

#### **5.1.2.6.1 Violations Flagged at `check_lp -stage upf -family powerstatetable` Stage**

- ❖ PST\_BIAS\_NOSTATE (error): Supply is used in a bias pair, but has no port states

This violation is flagged when the supply nets used in the bias pairs do not have any port states defined in the UPF. All further bias checks do not happen without port states.

This violation is disabled by default. You can enable it using the `configure_tag -app LP -tag PST_BIAS_NOSTATE -enable` command.

##### **Example**

```
create_supply_set SS_TOP -function {nwell VNW} -function {pwell VPW} -update
create_power_domain TOP -supply {primary SS_TOP} -update
```

- ❖ PST\_BIASOFF\_STATE (error): Bias net [UPFSupply] off, but supply net on for domain [Domain]

This violation is flagged when the `primary_power` or `primary_ground` supply of the domain is ON, but the related bias pins supply of the domain is OFF in a given PST state.

##### **Example**

In the following example, the `primary_power` VDD works at 1.2V and `related_bias_pin` VNW is OFF. The `primary_ground` VSS works at 0.0V and `related_bias_pin` VPW is OFF.

```
create_supply_set SS_TOP -function {power VDD} -function {ground VSS} -function {nwell VNW} -function {pwell VPW}
create_power_domain TOP -supply {primary SS_TOP}
create_psttop_pst -supplies {VDD VSS VNW VPW}
add_pst_state s1 -pst top_pst -state {VDD_12 VSS_00 VNW_OFF VPW_00}
add_pst_state s2 -pst top_pst -state {VDD_12 VSS_00 VNW_12 VPW_OFF}
```

- ❖ PST\_BIAS\_INSUFFICIENT (error): Bias net [UPFSupply] voltage insufficient relative to supply net for domain [Domain]

This violation is flagged when `related_bias_pin` supply voltage of the domain is insufficient compared to the `primary_power` or `primary_ground` supply voltage of the domain in given PST state.

##### **Example**

In the following example, the `primary_power` VDD works at 1.2V and `related_bias_pin` VNW works at 1.1V. The `primary_ground` VSS works at 0.0V and `related_bias_pin` VPW works at 0.1V.

```
create_supply_set SS_TOP -function {power VDD} -function {ground VSS}
create_power_domain TOP -supply {primary SS_TOP}
connect_supply_net VNW -ports ff_i1/VNW
connect_supply_netVPW -ports ff_i1/VPW
create_psttop_pst -supplies {VDD VSS VNW VPW}
add_pst_state s1 -pst top_pst -state {VDD_12 VSS_00 VNW_11 VPW_00}
add_pst_state s2 -pst top_pst -state {VDD_12 VSS_00 VNW_12 VPW_01}
```

- ❖ PST\_BIASON\_STATE (warning): Supply net off, but bias net [UPFSupply] on for domain [Domain]

This violation is flagged when `primary_power` or `primary_ground` supply of the domain is OFF but the related bias pins supply of the domain is ON in given PST state.

##### **Example**

In the following example, the primary\_power VDD is OFF and related\_bias\_pin VNW works at 1.2V. The primary\_ground VSS is OFF and related\_bias\_pin VPW works at 0.0V.

```
create_supply_set SS_TOP -function {power VDD} -function {ground VSS} -function {nwell VNW} -function {pwell VPW}
create_power_domain TOP -supply {primary SS_TOP}
create_psttop_pst -supplies {VDD VSS VNW VPW}
add_pst_state s1 -pst top_pst -state {VDD_OFF VSS_00 VNW_12 VPW_00}
add_pst_state s2 -pst top_pst -state {VDD_12 VSS_OFF VNW_12 VPW_00}
```

- ❖ **PST\_BIAS\_EXCESS** (warning): Bias net [UPFSupply] voltage excess relative to supply net for domain [Domain]

This violation is flagged when related\_bias\_pin supply voltage of the domain exceeds compared to the primary\_power or primary\_ground supply voltage of the domain in given PST state.

### Example

In the following example, the primary\_power VDD works at 1.2V and related\_bias\_pin VNW works at 1.3V. The primary\_ground VSS works at 0.1V and related\_bias\_pin VPW works at 0.0V.

```
create_supply_set SS_TOP -function {power VDD} -function {ground VSS}
create_power_domain TOP -supply {primary SS_TOP}
connect_supply_net VNW -ports ff_i1/VNW
connect_supply_net VPW -ports ff_i1/VPW
create_psttop_pst -supplies {VDD VSS VNW VPW}
add_pst_state s1 -pst top_pst -state {VDD_12 VSS_00 VNW_13 VPW_00}
add_pst_state s2 -pst top_pst -state {VDD_12 VSS_01 VNW_12 VPW_00}
```

#### 5.1.2.6.2 Violations flagged at check\_lp -stage pg -family bias stage

- ❖ **PG\_BIAS\_OPTIMIZE** (warning): Large area multi-well cell Instance (Cell) used where single well cell is sufficient

In some standard cells, there are two independent wells, each with its own well pin. These cells tend to have very large area because the two wells must be physically separated. In normal usage the two well pins are connected to different supply nets. However, if a large cell is used, and the two well pins are connected to the same net, then there is no benefit of using the large cell. This violation indicates that the cell has two well pins but they are connected to the same net. Changing the cell to one with a single well does not change the behavior, but reduces the area of the design.

### Example:

In following example, there are two nwell pins available and it is connected to same PG connection in design.

The following library file snippet, where there are two nwell pins defined:

```
cell (AOB_CELL) {
pg_pin(VNW) {
pg_type : nwell;
...
}
pg_pin(VNW1) {
pg_type : nwell;
...
} ... }
```

And the following is the corresponding design file snippet, where the cell has both the nwell pins defined for the same PG pin:

- AOB\_CELL aob\_i1 (.VDD(VDD) , .VSS(VSS) , .VNW(VNW) , .VNW1(VNW1) , .VPW(VPW) , .VPW1(VPW) ) ;
- ❖ PG\_BIAS\_SHORT (error): Bias net DesignNet shorted by non-insulated pin CellPin on Instance (Cell)
- In a standard cell design, the wells connect by abutment. If the bias pin for a cell is insulated, then its well does not connect by abutment. However, if an uninsulated bias pin is connected to a different supply, this creates a short between the supply and the main bias supply through the well abutment. This violation indicates that the bias pin is not insulated, and it causes a short from its connected supply to the main bias supply of the domain. This cause the design to function improperly, or not function at all.

#### **Example:**

In the following liberty snippet the same net is connected to VDD1 and VDD2.

```
set_domain_supply_net PD1 -primary_power_net VDD1 -primary_ground_net VSS
set_back_bias -domain PD1 -type nwell -net VDD1
connect_supply_net VDD2 -ports {u1/VNW}
```

- ❖ PG\_BIAS\_NOSTATE (error): Supply is used in a discovered bias pair, but has no portstates

This violation is flagged when supply nets discovered as bias pairs from the PG design does not have any port states defined in the UPF. All further Bias checks cannot happen without port states.

In some PG designs, there may be a large number of instances with a particular discovered bias pair; each such instance will be flagged.

This violation is disabled by default. You can enable it using the `configure_tag -app LP -tag PST_BIAS_NOSTATE -enable` command.

#### **Example**

Consider the following design:

```
module core1 (clk, in1, out1, VDD1, VSS, VNW);
  input clk, in1, out1, VDD1, VSS, VNW;
  output out1;

  SC22DFFQEXC1 ff_i1 ( .DATA(in1), .CLK(clk), .Q(out1), .VDD(VDD1), .VSS(VSS), .VNW(VNW)
);

endmodule
```

- ❖ PG\_BIASOFF\_STATE (error): Bias net [DesignNet] off, but supply net [UPFSupply] on for instance [Instance]

This violation is flagged when primary\_power or primary\_ground supply of the instance is ON but the related bias pins supply of the instance is OFF in given PST state.

For the protection cells, the PG\_BIASOFF\_STATE tag is reported in the following scenarios:

- ◆ There is a power state in which the output supply is on, but its bias supply is off.
- ◆ There is a power state in which the output supply is on, and the input supply is on, but the input bias supply is off.

When the `lp_protection_input_bias` application variable is enabled, the PG\_BIASOFF\_STATE violation will be reported in the following scenarios:

- ◆ There is a power state in which the output supply is on, but its bias supply is off.
- ◆ There is a power state in which the input supply is on, but its bias supply is off.

### Example

In the following example, the primary\_power VDD works at 1.2V and related\_bias\_pin VNW is OFF. The primary\_ground VSS works at 0.0V and related\_bias\_pin VPW is OFF.

```
create_supply_set SS_TOP -function {power VDD} -function {ground VSS} -function {nwell VNW} -function {pwell VPW}
create_power_domain TOP -supply {primary SS_TOP}
create_psttop_pst -supplies {VDD VSS VNW VPW}
add_pst_state s1 -pst top_pst -state {VDD_12 VSS_00 VNW_OFF VPW_00}
add_pst_state s2 -pst top_pst -state {VDD_12 VSS_00 VNW_12 VPW_OFF}
```

The default behavior of PG\_BIASOFF\_STATE for protection cells (ISO/LS/ELS) is that VC LP checks the supply states for RPP/RGP of output pin and its related bias pin. It does not consider the supply for input pin.

When lp\_protection\_input\_bias is set true, for protection cells, VC LP checks the supply states for RPP/RGP of input pin and its related bias pin. If input supply is ON and bias supply is OFF, PG\_BIASOFF\_STATE is reported.

- ❖ PG\_BIAS\_INSUFFICIENT (error): Discovered bias net [DesignNet] voltage insufficient relative to supply [UPFSupply] for instance [Instance]

This violation is flagged when related\_bias\_pin supply voltage of the instance is insufficient compared to the primary\_power or primary\_ground supply voltage of the instance in given PST state.

### Example

In the following example, the primary\_power VDD works at 1.2V and related\_bias\_pin VNW works at 1.1V. The primary\_ground VSS works at 0.0V and related\_bias\_pin VPW works at 0.1V.

```
create_supply_set SS_TOP -function {power VDD} -function {ground VSS}
create_power_domain TOP -supply {primary SS_TOP}
connect_supply_net VNW -ports ff_i1/VNW
connect_supply_net VPW -ports ff_i1/VPW
create_psttop_pst -supplies {VDD VSS VNW VPW}
add_pst_state s1 -pst top_pst -state {VDD_12 VSS_00 VNW_11 VPW_00}
add_pst_state s2 -pst top_pst -state {VDD_12 VSS_00 VNW_12 VPW_01}
```

- ❖ PG\_BIASON\_STATE (warning): Supply net [UPFSupply] off, but discovered bias net [DesignNet] on for instance [Instance]

This violation is flagged when primary\_power or primary\_ground supply of the instance is OFF but the related bias pins supply of the instance is ON in given PST state.

### Example

In the following example, the primary\_power VDD is OFF and related\_bias\_pin VNW works at 1.2V. The primary\_ground VSS is OFF and related\_bias\_pin VPW works at 0.0V.

```
create_supply_set SS_TOP -function {power VDD} -function {ground VSS} -function {nwell VNW} -function {pwell VPW}
create_power_domain TOP -supply {primary SS_TOP}
create_psttop_pst -supplies {VDD VSS VNW VPW}
add_pst_state s1 -psttop_pst -state {VDD_OFF VSS_00 VNW_12 VPW_00}
add_pst_state s2 -psttop_pst -state {VDD_12 VSS_OFF VNW_12 VPW_00}
```

- ❖ PG\_BIAS\_EXCESS (warning): Discovered bias net [DesignNet] voltage excess relative to supply [UPFSupply] for instance [Instance]

This violation is flagged when related\_bias\_pin supply voltage of the instance exceeds compared to the primary\_power or primary\_ground supply voltage of the instance in given PST state.

#### Example

In the following example, the primary\_power VDD works at 1.2V and related\_bias\_pin VNW works at 1.3V. The primary\_ground VSS works at 0.1V and related\_bias\_pin VPW works at 0.0V.

```
create_supply_set SS_TOP -function {power VDD} -function {ground VSS}
create_power_domain TOP -supply {primary SS_TOP}
connect_supply_net VNW -ports ff_i1/VNW
connect_supply_net VPW -ports ff_i1/VPW
create_psttop_pst -supplies {VDD VSS VNW VPW}
add_pst_state s1 -pst top_pst -state {VDD_12 VSS_00 VNW_13 VPW_00}
add_pst_state s2 -pst top_pst -state {VDD_12 VSS_01 VNW_12 VPW_00}
```

- ❖ PG\_TIEDTO\_SHORT (Error): Bias net [UPFBias] shorted to supply net [DesignSupply] by tied\_to pin [CellPin] on [Instance] ([Cell])

In some LibCell, the supply pins and bias pins are internally connected. For example, power supply and nwell are physically connected inside the cell. So if in design, the power net is connected to one supply net and nwell is connected to a different supply net, the two supply nets are shorted through the cell. Similarly, ground supply and pwell are physically connected inside a cell. In such cases, VC LP reports the PG\_TIEDTO\_SHORT violation.

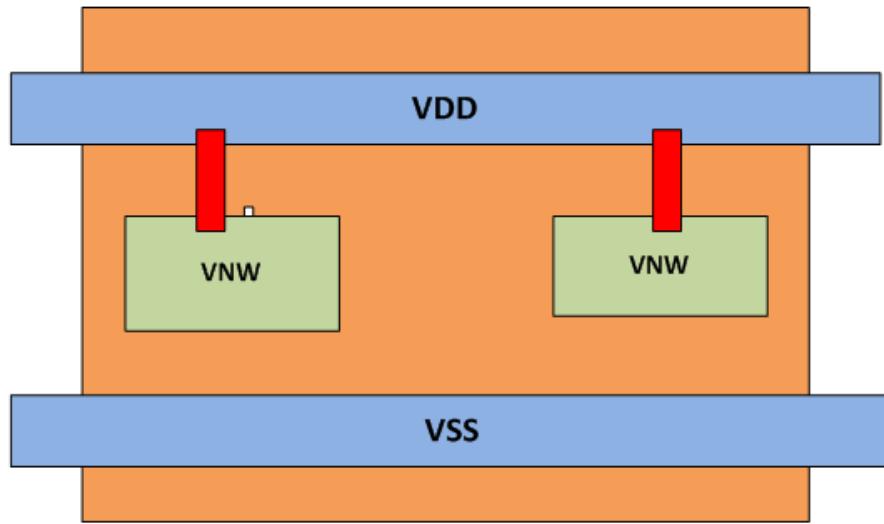
In liberty, this is modeled as:

```
pin (nwell) { tied_to : power }
pin (pwell) { tied_to : ground }
```

The following is an example of the liberty file snippet:

```
cell (AOB_TIED_TO) {
    pg_pin(VDD) {
        voltage_name : VDD;
        pg_type : primary_power
        related_bias_pin : "VNW";
    }
    pg_pin(VSS) {
        voltage_name : VSS;
        pg_type : primary_ground;
        related_bias_pin : "VPW";
    }
    pg_pin(VPW) {
        voltage_name : VSS;
        pg_type : pwell;
        physical_connection : device_layer;
        tied_to : VSS
    }
    pg_pin(VNW) {
        voltage_name : VDD;
        pg_type : nwell;
        physical_connection : device_layer;
        tied_to : VDD
    }
}
```

[Figure 5-6](#) shows an example LibCell Diagram in which the bias pin (VNW ) is internally physically tied to power supply (VDD).

**Figure 5-6 The PG\_TIEDTO\_SHORT Violation**

The following is the corresponding design snippet:

```
AOB_TIED_TO    aob_top1 (.VDD(VDD), .VDDG(VDD), .VNW(VNW), .VPW(VPW), .VSS(VSS));
```

#### **5.1.2.6.3 Violations Flagged at check\_ip -stage upf -family bias Stage**

- ❖ UPF\_BIAS\_SHORT (error): Bias net UPFSupply shorted by connect\_supply\_net on non-insulated pin CellPin on Instance (Cell)

This violation is reported when the bias pins of cells in a domain are connected to domain's primary nwell and pwell and the corresponding bias pin is overwritten by connect\_supply\_net which is different from the primary bias pin's net. This violation is reported only for a standard cell.

#### **Example**

*Standard\_cell.lib*

In following example domain primary bias net is overridden by create\_supply\_net.

```
cell (cellA) {
    pg_pin(VNW) {
        pg_type : nwell;
        ...
    }
    pg_pin(VPW) {
        pg_type : pwell;
        ...
    } ...
}

create_supply_set SS_PD1 -function {power VDD1} -function {ground VSS} -function {nwell
VDD1} -function {pwell VSS1}
create_power_domain PD1 -supply {primary SS_PD1}
connect_supply_net VDDN -ports {CORE1/cellA/VNW}
connect_supply_net VSSP -ports {CORE1/cellA/VPW}
```

- ❖ UPF\_PORT\_PSWOUT:

The UPF\_PORT\_PSWOUT is reported in scenarios where explicitly created UPF supply port with direction in is connected to the output supply of the power switch strategy on the same scope using connect\_supply\_net.

This is a UPF stage tag disabled by default.

```
Tag : UPF_PORT_PSWOUT
Description : Direction of supply port [SupplyPort] connected to power switch [Strategy]
output supply [UPFSupply] in the same UPF scope is not out
Violation : LP:2
SupplyPort
PortName : VDD2
PortType : UPF
Strategy : PSW_PD1
UPFSupply : PSW_PD1/VOUT
```

### 5.1.2.7 PG Checking

The command `check_lp -stage pg` checks all the PG connections including bias connections. If the domain has bias nets defined in UPF, check that the instances in the domain have the correct connections. If the instances have bias pins but there is no bias net defined in the UPF, the check will assume some implicit connections. The nwell connection should go to the primary power net for the domain, and the pwell connection should go to the primary ground net for the domain. The PG\_DOMAIN\_CONN check is applied using the nets mentioned above. The standard checks for PG\_PIN\_UNCONN, PG\_PIN\_TIED, PG\_CSN\_CONN are applied to bias pins on all cell types. The PinType field in the message shows pwell or nwell.

When the design has a large number of incorrect PG connections, then VC LP reports existing violation PG\_DOMAIN\_SETUP and skips other checks. In such cases, if you want to ignore this message, and force VC LP to perform checks, then set the `enable_pg_setup_stop` application variable to true.

When the design has large number of incorrect PG connections, then VC LP reports existing violation PG\_DOMAIN\_SETUP and skips other checks. In such cases, if you want to ignore this message, and force VC LP to perform checks, `-force` should be set with `check_lp` command. If you disable PG\_DOMAIN\_SETUP, then `-force` is not required in `check_lp` command to perform checks.

Similarly, for PG\_BBOX\_SETUP violation, if you disable PG\_BBOX\_SETUP, then `-force` is not required in the `check_lp` command to perform VC LP checks.

### 5.1.2.8 Limitation

VC LP reports violations only for the PG pins where the `related_bias_pins` are defined. VC LP does not report violations for logic pins where `related_bias_pin` is defined.

### 5.1.3 Support for Bias Checks for Flipwell Cells

VC LP supports bias checks for flipwell cell. In conventional cell, nwell is connected to primary power pin, and pwell is connected to primary ground pin. In a flipwell cell, nwell and pwell is connected to primary ground pin.

You can define flipwell domain and flipwell cells using the following application variables:

- ❖ To define flipwell cell, use the `flipwell_cell_list` application variable.

```
set_app_var flipwell_cell_list <lib_cell_name>
```

Note that you must provide the exact `lib_cell_name` in list. VC LP does not report any error/warning for the incorrect `lib_cell_name`.

- ❖ To define a flipwell power domain, use the `flipwell_domain_list` application variable.

```
set_app_var flipwell_domain_list <power_domain_name>
```

Note that you must provide the exact domain name in list. VC LP does not report any error/warning for incorrect domain names.

Both these application variables must be set before the `read_upf` command. Wildcard "\*" is supported in these two application variables so all design power domains and cells are treated as flipwell one.

### Checks Introduced for Flipwell cells/domain

Ideally, flipwell cell must present in flipwell domain. In case you forget to define flipwell domain or cell, VC LP reports the following violations.

- ❖ `DESIGN_BIAS_FLIPWELL`

When a flipwell cell is present in a non-flipwell domain VC LP reports the `DESIGN_BIAS_FLIPWELL` violation.

```
Tag      : DESIGN_BIAS_FLIPWELL
Description : Flip well cell instance [Instance] found in non-flip well domain
[Domain]
Violation   : LP:1
Instance     : CORE1/and_i1
Domain       : PD1
```

- ❖ `DESIGN_BIAS_NONFLIPWELL`

When a non flipwell cell is present in flipwell domain, VC LP reports the `DESIGN_BIAS_NONFLIPWELL` violation.

```
Tag      : DESIGN_BIAS_NONFLIPWELL
Description : Non-flip well cell instance [Instance] found in flip well domain
[Domain]
Violation   : LP:1
Instance     : CORE1/and_i1
Domain       : PD1
```

These violations are disabled by default.

### Impact on Existing PG\_BIAS\_EXCESS/INSUFFICIENT Violations

When a cell is defined as a flipwell cell through the `flipwell_cell_list` application variable in the Tcl file, VC LP compares voltage value of nwell and pwell with primary ground pin. If nwell voltage is lower or higher than primary ground, VC LP reports the `PF_BIAS_INSUFFICIENT` and `PG_BIAS_EXCESS` violations respectively.

### Impact on Existing PST\_BIAS\_EXCESS/INSUFFICIENT Violations

When a domain is defined as a flipwell domain through the `flipwell_domain_list` application variable in the Tcl file, VC LP compares voltage value of pwell with domain primary ground pin. If pwell voltage is lower or higher than primary ground, VC LP reports the `PST_BIAS_INSUFFICIENT` and `PST_BIAS_EXCESS` violations respectively.

### Impact on Existing PG\_BIASON\_STATE/PG\_BIASOFF\_STATE/PG\_BIAS\_PATH Violations

When a cell is defined as a flipwell cell through the `flipwell_cell_list` application variable in the Tcl file, VC LP picks the cell primary ground pin to do these checks, and no checks on the primary power pin.

### Impact on Existing PST\_BIASON\_STATE/PST\_BIASOFF\_STATE/PST\_BIAS\_PATH Violations

When domain is defined as flipwell domain through the `flipwell_domain_list` application variable in the Tcl file, VC LP picks the domain primary ground pin to do these checks, and no checks on the primary power pin.

#### 5.1.4 Support for Self Bias Checks

In a conventional bias technology, the library cells have explicit nwell/pwell PG pins. The existing VC LP `PG_BIAS*` checks looks for nwell/pwell PG pins of the library cell, and performs bias checks.

In a self bias technology, the library cells do not have explicit nwell/pwell PG pins, but the nwell/pwell are internally shorted to power/ground PG pins of the cell. The self bias checks are introduced to support such self bias cells.

The self bias checks support is introduced under the `enable_selfbias_checks` application variable. By default, the `enable_selfbias_checks` application variable is set to false.

When the `enable_selfbias_checks` application variable is set to true, VC LP performs the following `PG_BIAS*` checks:

- ❖ For each self bias library cell instances, VC LP compares UPF nwell supply of the domain with the supply connected to primary\_power/backup\_power PG pins of the cell in DESIGN/NETLIST.
- ❖ For each self bias library cell instances, VC LP compares UPF nwell supply of the domain with the supply connected to primary\_ground/backup\_groud PG pins of the cell in DESIGN/NETLIST.

The following violations are supported for self bias cells:

- ❖ `PG_BIAS_INSUFFICIENT`
- ❖ `PG_BIAS_EXCESS`
- ❖ `PG_BIAS_PATH`
- ❖ `PG_BIAS_NOSTATE`
- ❖ `PG_BIASOFF_STATE`
- ❖ `PG_BIASON_STATE`

By default, all the library cells (Standard and Low Power cells) without explicit nwell/pwell PG pins are considered for self bias checks.



#### Note

The Macro/PAD cells are not considered for self bias checks.

- ❖ If a list of pins is specified using the `upf_selfbiascellpin_list` application variable, then self bias checks are performed for the specified pins only. However, in this list, you can specify backup power/ground pins as well. The self bias checks are then performed on those backup power/ground pins. The cell pins must be specified in `<lib_name> / <cell_name> / <pin_name>` format.

#### Example

```
set_app_var upf_selfbiascellpin_list {bias/INV_SELF/VDD bias/INV_SELF/VSS}
```

Where:

- ◆ `bias` is the name of library
- ◆ `INV_SELF` is the name of cell
- ◆ `VDD/VSS` are the PG pins of the cell

- ❖ If a PG\_BIAS\* violation is covered by a related PST\_BIAS \* violation, the related PG violation is not reported.

### Example

```

Tag          : PST_BIASOFF_STATE
Description  : Bias net [UPFSupply] off, but supply net [ReferenceInfo] on
Violation    : LP:30
UPFSupply   : VDD2
ReferenceInfo
PowerNet
  NetName   : VDD1
  NetType    : Design/UPF
GroundNet
  NetName   : VSS1
  NetType    : Design/UPF
Domain       : da
PinPGType   : NWell
States
State        : my_pst/s5

```

The PG\_BIASOFF\_STATE violation is suppressed:

```

Tag          : PG_BIASOFF_STATE
Description  : Bias net [DesignNet] off, but supply net [UPFSupply] on for instance
[Instance]
Violation    : LP:13
DesignNet
  NetName   : VDD1
  NetType    : Design/UPF
UPFSupply   : VDD2
Instance     : ua/inv_self_inst_u6
CellPin      : VDD
PinPGType   : PrimaryPower
States
State        : my_pst/s5
IsSelfBias   : true

```

To differentiate a violation reported because of self bias checks, VC LP reports the *IsSelfBias : true* debug field in the violation report.

## 5.1.5 Analog Checks

The VC LP analog checks (when -analog is specified or no other switches are specified) ensure that analog pins are all connected only to analog pins. On such an path, all analog pins are driven by the same voltage.

A pin that has the attribute `is_analog=true` in the liberty database for a cell, is considered to be an analog pin. VC LP also performs analog checks if the library cell has `is_analog : true` defined on individual bits of a bus vector port.

### Example

```

bus(VINP) {
  bus_type : bus_19_0;
  direction : input;
  is_analog : true;
  ...
  ...
}

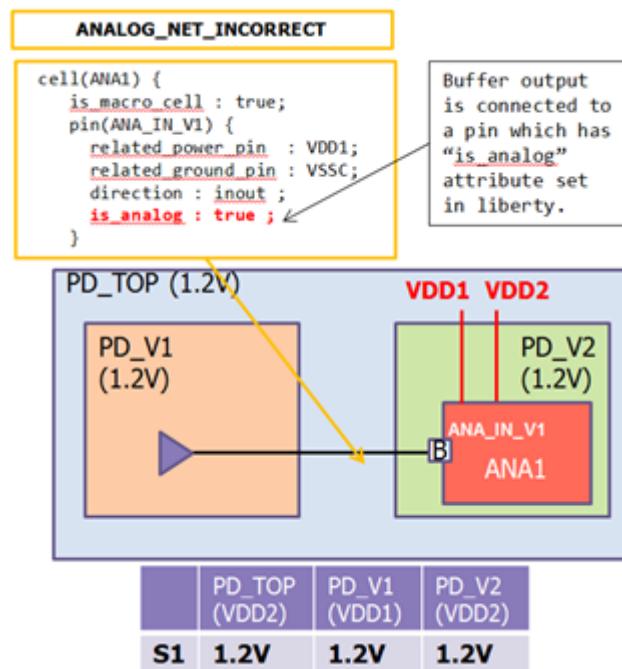
```

To perform analog checks, use the `-analog` switch in the `check_lp -stage` design command.

If this switch is specified, analog checks are performed on the design. Analog checks are run by default if the check\* commands are run without any switches. When the -analog switch is specified in the check\_lp -stage design command, the following features are enabled:

- ❖ Analog net check (reports ANALOG\_NET\_INCORRECT violation)
  - ◆ For all nets in design
  - ◆ Checks if all the DB cell pins are connected to the net.
  - ◆ If any pin is marked as is\_analog, then the net is marked analog net.
  - ◆ If an analog net is mapped with a digital net, VC LP reports the ANALOG\_NET\_INCORRECT violation.

**Figure 5-7 ANALOG\_NET\_INCORRECT**



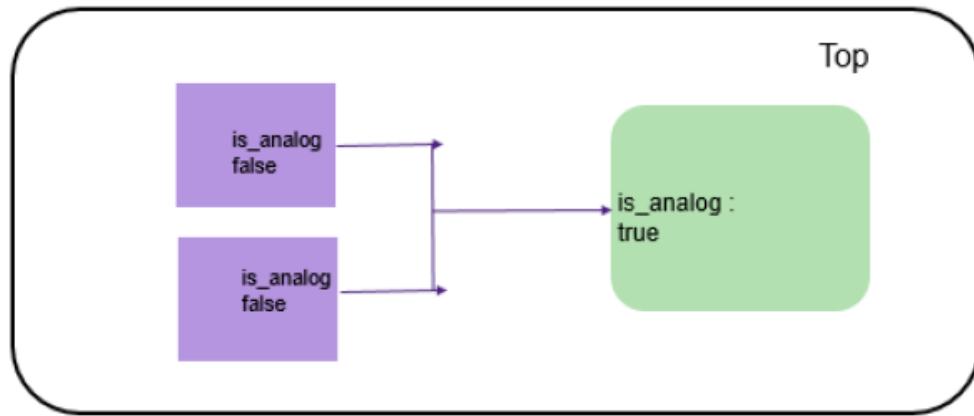
- ◆ For the example scenario as shown in [Figure 5-8](#), multiple digital pins are connected to a single analog pin (is\_analog true). By default, VC LP reports two ANALOG\_NET\_INCORRECT for this case. When the compress\_analog\_incorrect\_violations application variable is set to true, VC LP reports only one ANALOG\_NET\_INCORRECT violation for such cases with SuppressCount 1 debug field.

The compress\_analog\_incorrect\_violations application variable is enhanced to compress ANALOG\_NET\_INCORRECT violation based on digital pins as well. The compress\_analog\_incorrect\_violations application variable can take the following options:

- ❖ analog: Compress based on analog pins. Report one violation per analog pin.
- ❖ digital: Compress based on digital pins. Report one violation per digital pin.
- ❖ true: same as analog, for backward compatibility.
- ❖ false (default): no compression

The compression considers the ReasonCode debug field for compression. One violation is kept uncompressed for each analog pin for each reason code.

**Figure 5-8 The ANALOG\_NET\_INCORRECT Violation**

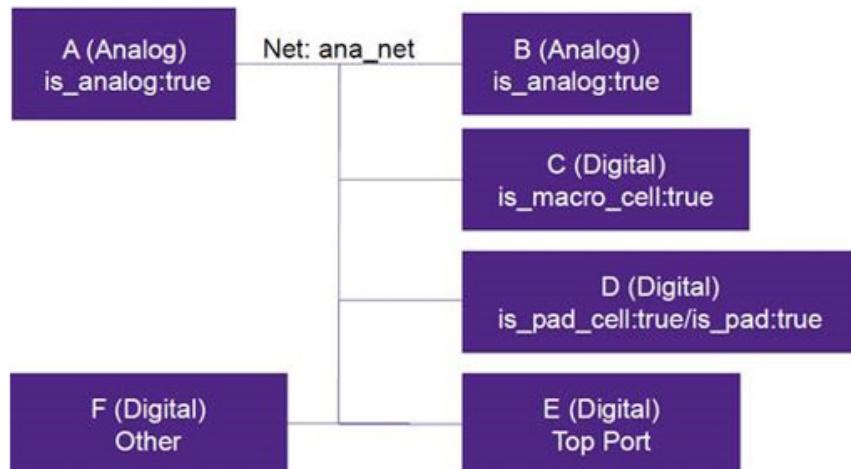


- ◆ The ANALOG\_NET\_INCORRECT violation is reported irrespective of the pin direction. Eight ANALOG\_NET\_INCORRECT violations are reported: (A-C, A-D, A-E, A-F, B-C, B-D, B-E & B-F) for the example in [Figure 5-9](#)

## Example

Consider the design in the following example.

**Figure 5-9 Example for ANALOG\_NET\_INCORRECT**



- ❖ Analog voltage check (reports ANALOG\_STATE\_UNSAFE and ANALOG\_VOLTAGE\_UNSAFE violations)
    - ◆ ANALOG\_STATE\_UNSAFE

The ANALOG\_STATE\_UNSAFE violation is reported if there is a PSR and sink has OFF\_to\_ON scenario in an analog to analog crossover.

## Example

In the following example, the source power net (VDD1) is OFF, while the sink power net (VDD2) is ON, and both the source and sink has the `is_analog:true` attribute.

Consider the following crossover:

`CORE1/and_i1/Z (VDD1, VSS1) --> CORE2/ff_i1/D (VDD2, VSS2)`

### UPF Snippet

```
add_port_state VDD1 -state {VDD_10 1.0} -state {VDD_12 1.2} -state {VDD_OFF off}
add_port_state VDD2 -state {VDD_10 1.0} -state {VDD_12 1.2} -state {VDD_OFF off}
add_port_state VSS -state {VSS_00 0.0} -state {VSS_OFF off}
add_port_state VSS1 -state {VSS_00 0.0} -state {VSS_OFF off}
add_port_state VSS2 -state {VSS_00 0.0} -state {VSS_OFF off}
create_pst top_pst -supplies { VDD1 VDD2 VSS VSS1 VSS2 }
add_pst_state s0 -pst top_pst -state { VDD_12 VDD_12 VSS_00 VSS_00
VSS_00 }
add_pst_state s2 -pst top_pst -state { VDD_OFF VDD_10 VSS_00 VSS_00
VSS_00 }
```

The following is the violation snippet reported:

```
Tag : ANALOG_STATE_UNSAFE
Description : Analog driver pin [AnalogPin] is off when analog
receiver pin [CellPin] is on for states [States]
Violation : LP:1
AnalogPin : CORE1/and_i1/Z
CellPin : CORE2/ff_i1/D
States
State : top_pst/s2
SourceInfo
PowerNet
NetName : VDD1
NetType : UPF
PowerMethod : FROM_UPF_POWER_DOMAIN
GroundNet
NetName : VSS1
NetType : UPF
GroundMethod : FROM_UPF_POWER_DOMAIN
SinkInfo
PowerNet
NetName : VDD2
NetType : UPF
PowerMethod : FROM_UPF_POWER_DOMAIN
GroundNet
NetName : VSS2
NetType : UPF
GroundMethod : FROM_UPF_POWER_DOMAIN
```

### ◆ ANALOG\_VOLTAGE\_UNSAFE

The ANALOG\_VOLTAGE\_UNSAFE violation is reported if there is a PST state where the source and sink are working on different supply voltages in an analog to analog crossover.

#### Example

In the following example, the source power net (VDD1) is working on 1.2V, while sink power net (VDD2) is working on 1.0V, both the source and sink has the `is_analog:true` attribute.

Consider the following crossover:

*CORE1/and\_i1/Z (VDD1, VSS1) --> CORE2/ff\_i1/D (VDD2, VSS2)*

#### UPF Snippet

```
add_port_state VDD1 -state {VDD_10 1.0} -state {VDD_12 1.2} -state {VDD_OFF off}
add_port_state VDD2 -state {VDD_10 1.0} -state {VDD_12 1.2} -state {VDD_OFF off}
add_port_state VSS -state {VSS_00 0.0} -state {VSS_OFF off}
add_port_state VSS1 -state {VSS_00 0.0} -state {VSS_OFF off}
add_port_state VSS2 -state {VSS_00 0.0} -state {VSS_OFF off}

create_pst top_pst -supplies          { VDD1   VDD2   VSS   VSS1
VSS2 }
add_pst_state s0 -pst top_pst -state { VDD_12  VDD_12  VSS_00  VSS_00
VSS_00 }
add_pst_state s2 -pst top_pst -state { VDD_12  VDD_10  VSS_00  VSS_00
VSS_00 }
```

The following is the violation snippet reported:

```
Tag : ANALOG_VOLTAGE_UNSAFE
Description : Analog pin [AnalogPin] connected to analog pin [CellPin]
which has different voltage in states [States]
Violation : LP:1
AnalogPin : CORE2/ff_i1/D
CellPin : CORE1/and_i1/Z
States
State : top_pst/s2
SourceInfo
PowerNet
NetName : VDD1
NetType : UPF
PowerMethod : FROM_UPF_POWER_DOMAIN
GroundNet
NetName : VSS1
NetType : UPF
GroundMethod : FROM_UPF_POWER_DOMAIN
SinkInfo
PowerNet
NetName : VDD2
NetType : UPF
PowerMethod : FROM_UPF_POWER_DOMAIN
GroundNet
NetName : VSS2
NetType : UPF
GroundMethod : FROM_UPF_POWER_DOMAIN
```

#### 5.1.5.1 The ISO\_STRATEGY\_ANALOG and LS\_STRATEGY\_ANALOG Violations

The ISO\_STRATEGY\_ANALOG and LS\_STRATEGY\_ANALOG violations are supported under the handle\_analog\_crossover application variable. The handle\_analog\_crossover can take two values default and pst.

- ❖ When this application variable is set to *default*, VC LP does not perform crossover related checks like ISO\_INST\_MISSING, LS\_INST\_MISSING for ANALOG crossovers, that is, a crossover having either Source or Sink or both with the *is\_analog: true* attribute. By default, this application variable is set to *default*.

- ❖ When this application variable is set to *pst*, VC LP performs crossover related checks for analog crossovers and reports violations as follows:
  - ◆ If an ISO strategy is present on the ANALOG crossover, VC LP reports the ISO\_STRATEGY\_ANALOG violation.  
This violation is reported at the `check_lp -stage` UPF stage with severity *error*. This violation is enabled by default.
  - ◆ If an LS strategy is present on the ANALOG crossover, then VC LP reports the LS\_STRATEGY\_ANALOG violation.  
This violation is reported at the `check_lp -stage` UPF stage with severity *warning*. This violation is enabled by default.

### 5.1.5.2 The ANALOG\_PIN\_INCORRECT violation

The ANALOG\_PIN\_INCORRECT violation is disabled in default. The ANALOG\_PIN\_INCORRECT violation is reported when an analog pin is connected with a digital pin or constant. The ANALOG\_PIN\_INCORRECT violation returns more information about the digital pin. This violation is also reported irrespective of the pin direction. Eight ANALOG\_PIN\_INCORRECT violations flag for example design specified in [Figure 5-9](#).

The following is an example of the ANALOG\_PIN\_INCORRECT violation:

```
Tag : ANALOG_PIN_INCORRECT
Description : Net [DesignNet] connected to analog pin [AnalogPin] is
also connected to digital pin [DigitalPin]
DesignNet
  NetName : ana_net
  NetType : Design
  AnalogPin : and_gate6/Z
  DigitalPin : pad_cell/A
  AnalogCellName : AND2_analog
  DigitalCellName : IOCB2EBTNH4LA01
  DigitalPinType : Pad
```

The *DigitalPinType* debug field returns - Pad, TopPort, Macro, LiteralConstant, SupplyPort, TieCell and Other. It returns which type of digital pin is connected with the analog pin.

## 5.1.6 SCMR Checks

SCMR (std\_cell\_main\_rail) is a liberty attribute that determines how the physical connection to a rail is handled in the floor planning stage. Multi rail cells must have at least one of their PG pins marked as an SCMR. If an SCMR is defined on a cell's pg\_pin either implicitly or explicitly, physically that pin must be connected to the domain primary rail in which the cell instance exists.

### 5.1.6.1 VC LP Checks on SCMR

The DESIGN\_PIN\_SCMR violation is implemented to check if a pg\_pin with explicit std\_cell\_main\_rail attribute in liberty is connected to a supply in the design or UPF that is not same as the domain supply to which the cell belongs.

The check is executed under `check_lp -stage` design command and is disabled by default. To enable it, use the `configure_tag -app LP -enable` command.

#### Violation Example

```
Tag : DESIGN_PIN_SCMR
Description : Standard cell main rail supply pin [CellPin] of instance
```

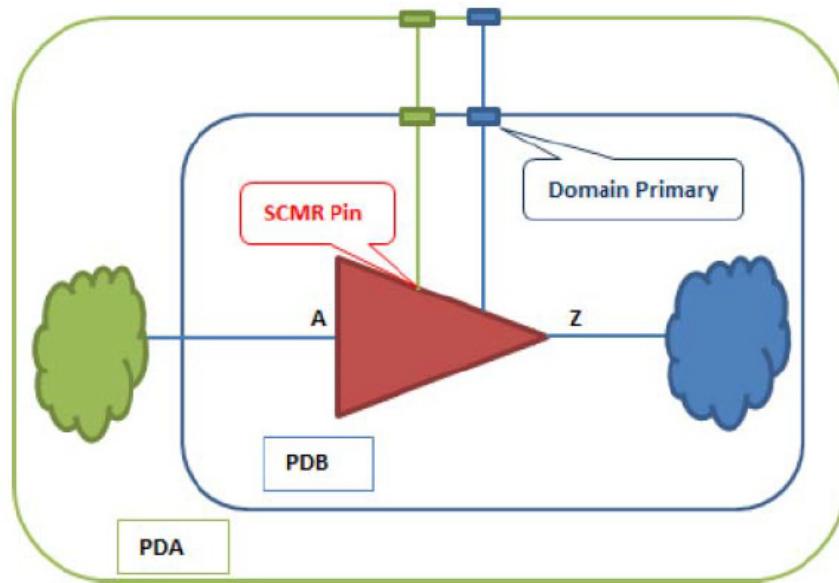
```
[Instance] not connected to
domain primary supply [UPFNet]
UPFSupply: VDD2
DesignNet
NetName : VDD1
NetType : Design StdCellMainRail : True CellPin : VDD
CellType : AlwaysOnBuffer
Instance : u1
Cell : AONBUF_A
PinPGType : PrimaryPower
InstanceStateInfo
PowerNet
NetName : VDD
NetType : Design/UPF
PowerMethod : FROM_PG_NETLIST
GroundNet
NetName : VSS
NetType : Design/UPF
GroundMethod : FROM_PG_NETLIST
```

The tag DESIGN\_PIN\_SCMR is a domain centric check. The following section describes the cases where DESIGN\_PIN\_SCMR is reported.

### Case1: Different root supplies

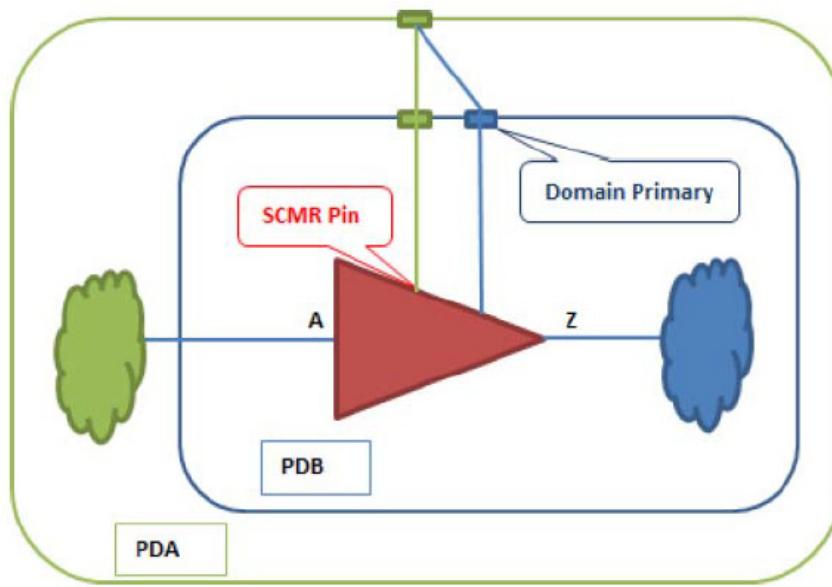
If the domain primary is not connected to the PG pin having an SCMR attribute, then DESIGN\_PIN\_SCMR violation is reported.

**Figure 5-10 Different root supplies**



### Case2: Same root supply

If the domain primary is not connected to the PG pin having an SCMR attribute and the root supply of both the PG pins is the same, DESIGN\_PIN\_SCMR is still reported, as this is a domain centric check. This check does not look for the resolved supply.

**Figure 5-11 Same Root Supply****Case 3: Domain's Primary and CELL's SCMR pin are connected electrically, but not hierarchically**

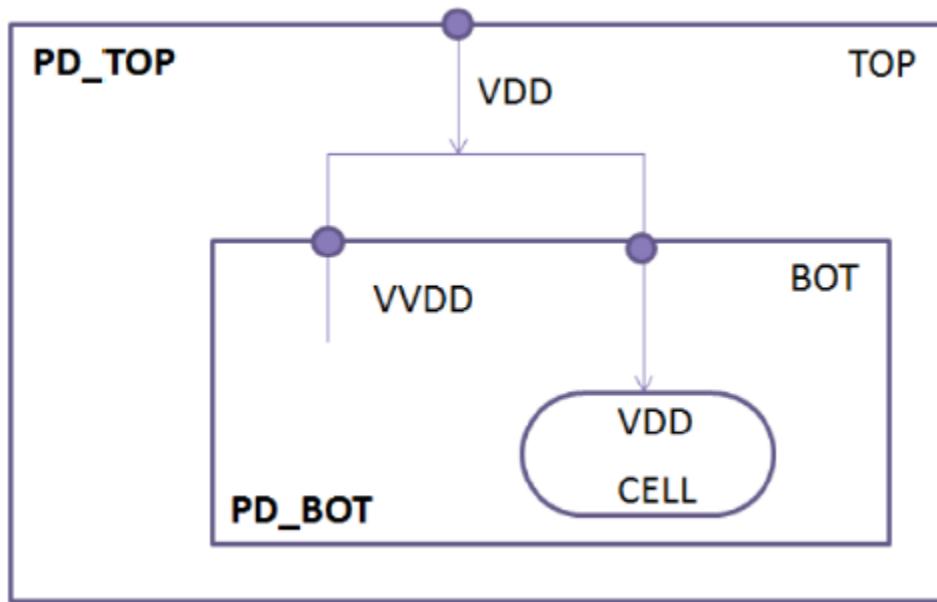
VC LP issues the DESIGN\_PIN\_SCMR violation with debug field ReasonCode: ELECTRIC\_CONN\_HIER\_UNCONN for the following scenario.

As shown in [Figure 5-12](#), the Supply net BOT/VVDD does not drive any cell within BOT directly. It indirectly drives BOT/CELL/VDD pg pin (SCMR). Here the domain's Primary and CELL's SCMR pin are connected electrically, but not hierarchically. The DESIGN\_PIN\_SCMR check can be enabled for this scenario using the enable\_contained\_scmr\_flow application variable.

```
%vc_static_shell> set_app_var enable_contained_scmr_flow true
```

By default, this application variable is set to false.

**Figure 5-12 Domain's Primary and CELL's SCMR pin are connected**

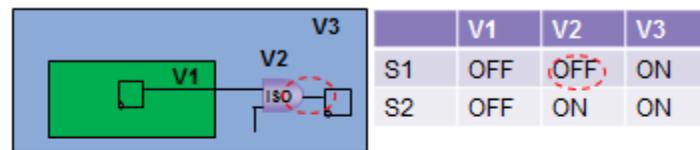


In addition, VC LP performs UPF consistency checks between RPP+CSN and SRSN supplies for a given lib pin

- ❖ If SRSN is defined on the output pin of a lib cell, then related power supply should be more on than SRSN. If this rule is violated then the consistency check issues the error UPF\_LEAFSRSN\_STATE.
- ❖ If SRSN is defined on input pin of a lib cell, then SRSN should be more on than the related power supply. If this rule is violated then consistency check issues the error UPF\_LEAFSRSN\_STATE.
- ❖ If SRSN is specified on an inout pin of lib cell, the pin is considered to be both an input and output pin one at a time and hence both of above consistency checks are performed.
- ❖ If there is a voltage level mismatch between SRSN supply and related power supply then consistency check issues the error UPF\_LEAFSRSN\_VOLTAGE.

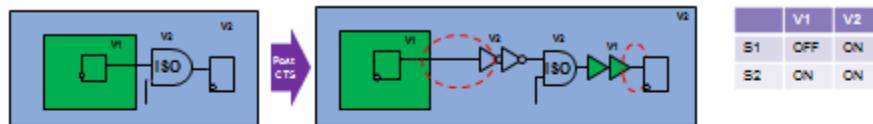
### 5.1.7 Advanced Rail Order Checks

Rail order checks are typically electrical violations that are caused by incorrect supply connectivity for the protection cells [ISO, LS, ELS] cells. Normally, you can insert protection cells to isolate or electrically safe guard a sink that is ON while the source is OFF. But in any PST state, if the protection cells' supply causes electrical violation, then VC LP reports it as a rail order violation on the protection cell. The following example denotes a typical rail order violation.

**Figure 5-13 Rail Order Checks**

**ISO rail order violation**  
**Issue** : Isolation needed but isolation supply is OFF when sink is ON  
**Solution** : Fix the incorrect supply connectivity or PST for isolation supply

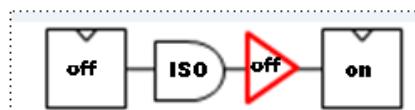
Advanced rail order checks extends the rail order checks to buffers and inverters. Advanced rail order checks are electrical checks caused by incorrect supply connectivity for all the cells that are introduced during low power implementation, that is, for buffers, inverters, LS, ISO, ELS cells on a crossover path. Typically these cells are not present at the RTL stage and are introduced by implementation tools during synthesis. So, for example, if a wrong choice of buffer supply connectivity results in an electrical violation, VC LP reports it as buffer rail order violation (ISO\_\* checks).

**Figure 5-14 Advanced Rail Order Checks**

**Buffer rail order violation**  
**Issue** : Incorrect buffer/inverter supply lead to additional electrical issues  
**Solution** : Fix the incorrect buffer supply connectivity or PST for buffer supplies

There are four types of advanced rail order checks based on their electrical and functional impact. This classification is based on the 'type of cell' whose supply connection has caused electrical and/or functional issues. The classification helps users in debug and root causing design issues.

- ❖ ISO\_\*\_STATE (electrically correct, functionally incorrect (data lost))



- ❖ ISO\_\*\_STATE (electrically incorrect, functionally correct)



- ❖ ISO\_\*\_FUNC (electrically correct, functionally incorrect)



- ❖ ISO\_\*\_WASTE (electrically inefficient, functionally correct)



### 5.1.7.1 Parking and Sink off Clamp Checks

Some designs have domains which go into a reduced voltage state. As long as the inputs to the domain change do not change, the circuit retains these values; but if any input changes, the circuit gets corrupted. Therefore, these domains require isolation on the inputs. The UPF language reference manual uses the term `sink_off_clamp` for this.

All related messages contains a field `SinkOffClamp : true`. These messages can be filtering by `SinkOffClamp == true`.

#### Syntax

```
%vc_static_shell> enable_sink_off_clamp_check -help
Usage
enable_sink_off_clamp_check      # sink off clamp check
[-src_domain <domain>]          (source domain)
[-sink_domain <domain>]          (sink domain)
```

#### Use Model

```
%vc_static_shell> enable_sink_off_clamp_checks -src_domain PD_1 -sink_domain PD_2
```

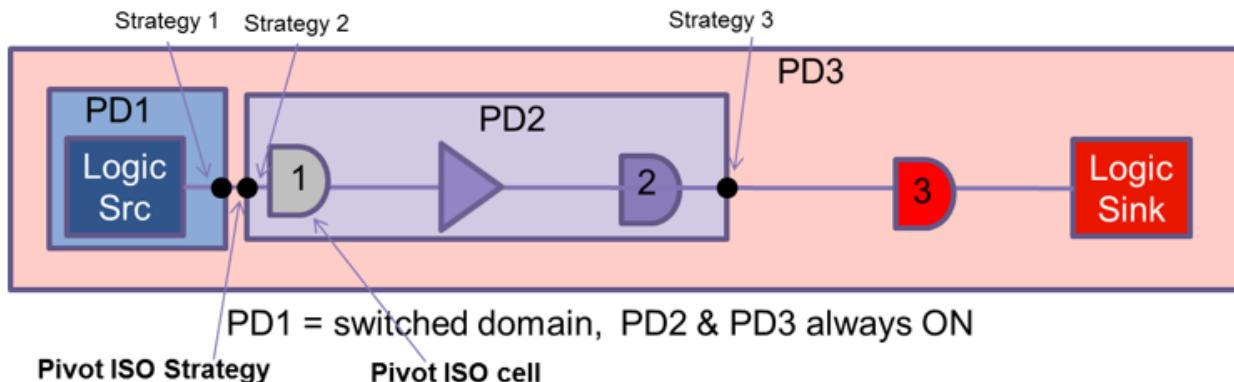
### 5.1.7.2 Enhanced Pivot Flow

Pivot ISO strategy is the ISO strategy which protects the logicSink from any OFF gate in its fanin cone, either electrically or functionally by preventing the cross-over propagation. Pivot ISO cell is the ISO cell is similar to pivot ISO strategy.

By default, VC LP considers the ISO strategy (and ISO cell, if any) which is closest to the logicSink. However, this computation is not always correct for all design scenarios.

In the example in [Figure 5-15](#), VC LP assumes the ISO cell 3 to be the pivot ISO cell. However, in the design, ISO cell 3 does not protect anything, as PD2 and PD3 are always ON. The only ISO cell which needs to protect the sink (hence need to be present) is the ISO cell 1. The ISO cell 2 and 3 are redundant.

The same is true for ISO strategy, strategy 1 is the pivot strategy.

**Figure 5-15 Example of Pivot Computation**

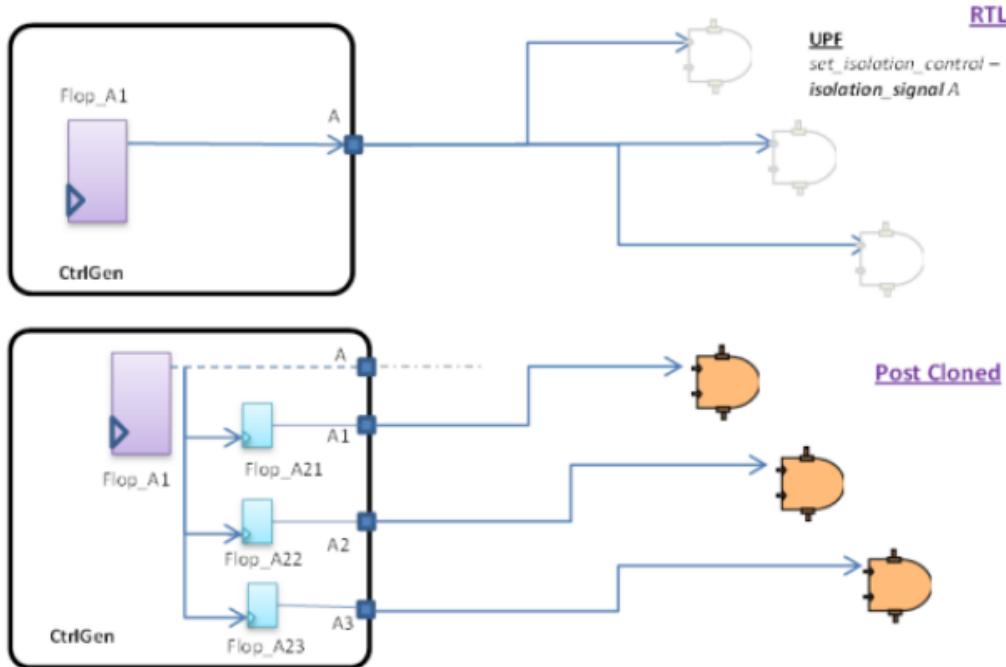
When you set `enable_enhanced_pivot_flow` application variable to true, VC LP computes the pivot ISO cell in the following way.

1. Start from logic sink, VC LP starts traversing backwards on the crossover path towards the logic source.
2. When a pivot ISO gate is not found:
  - a. Find the first gate which can go OFF when sink is ON, and this is reported as the corrupter gate.
    - ◊ In [Figure 5-15](#), it is Logic Src.
    - ◊ If none found, then there is no pivot ISO cell.
  - b. Find the first ISO cell right of the corrupter gate (towards the sink).
    - ◊ If the ISO cell is found, VC LP denotes that as the pivot ISO cell (in [Figure 5-15](#), it is ISO cell 1)
    - Note: An ISO cell can be corrupter gate as well, based on isolation supply.
  - c. If no such ISO cell found, then go to step 2, and traverse further towards Logic Source.

For the pivot ISO strategy, VC LP performs a similar computation, but VC LP finds the ISO strategy instead of the ISO cell. While computing pivot ISO strategy, the ISO supply of other strategies must be considered as corrupter. For example, in [Figure 5-15](#), if the Strategy 1 supply can go OFF when the sink is ON, then Strategy 2 becomes pivot strategy.

### 5.1.8 Isolation Control Port Cloning

VC LP performs a number of checks, like, reach-ability using the isolation enable signals specified in the UPF isolation strategy. If a synthesized netlist containing an isolation enable signal is updated to have cloned ports or signals for the same, without updating the corresponding UPF strategy, the tool would report reach-ability errors for the isolation devices associated with the strategy. [Figure 5-16](#) illustrates such a scenario:

**Figure 5-16 High Fanout Handling for Isolation Control**

To avoid this, use the following command after the design and UPF are read but you must do it before performing the checks.

### Syntax

```
%vc_static_shell> set_clone -help
Usage: set_clone      # specify cloned ports/pins with respect to original control signal
defined in upf
      -upf <str>          (upf signal name)
      -instance <str>        (instance (db or hd1) name)
      [-port <str>]         (port name of cloned ports)
      [-pin <str>]          (pin names of cloned instances)
```

### Examples:

```
set_clone -upf <isoEnSignal> -instance ff -pin Q|-port A1
```

For the example scenario illustrated in [Figure 5-16](#), you must add the following:

For clone ports **CtrlGen/A1**, **CtrlGen/A3** and **CtrlGen/A3** or clone pins **Ctrlgen/Flop\_A21/Q**, **Ctrlgen/Flop\_A22/Q** and **Ctrlgen/Flop\_A23/Q**

Before you clone the ports, you must read the design. Once the design is read, use the **set\_clone** command. After the cloning is done, you can perform UPF and design related checks using the **check\_lp -stage upf** and **check\_lp -stage design** commands.

The following is the syntax for reading the design:

```
read_file -format verilog -top top -netlist "<testcase.v>" -verilog
read_upf <tetscase.upf>
```

```
set_clone -upf CtrlGen/A -instance Ctrlgen/Flop_A21 -pin Q
set_clone -upf CtrlGen/A-instance Ctrlgen/Flop_A22 -pin Q
```

```
set_clone -upf CtrlGen/A -instance CtrlGen/Flop_A23 -pin Q
```

Or

```
set_clone -upf CtrlGen/A -instance CtrlGen -port A1
set_clone -upf CtrlGen/A -instance CtrlGen -port A2
set_clone -upf CtrlGen/A -instance CtrlGen -port A3
```

Or

```
foreach j "[get_object_name [get_cells -hierarchical "Flop_A*"]]" {
set_clone -upf CtrlGen/A -instance $j -pin {Q}
}
```

With the `set_clone` command, isolation controls for different isolation cells (being driven by A1, A2, A3.... ports) will not be reported as isolation unreachable even if the isolation strategy has isolation signal such as CtrlGen/A,. This means that indirectly ISO policy is getting updated with the cloned ports/pins.

#### Note

Signal corruption checks happen between cloned signals (CtrlGen/A1 CtrlGen/A2 CtrlGen/A3) and isolation cells enable pin.

To find out the original root, trace constraints for Flop\_A1 must be considered as the actual driver for enables:

```
create_trace_constraint -trace_through -instance CtrlGen/Flop_A21 \
    -input_port D -output_port Q -signal_type iso_enable
create_trace_constraint -trace_through -instance CtrlGen/Flop_A21 \
    -input_port D -output_port Q -signal_type iso_enable
create_trace_constraint -trace_through -instance CtrlGen/Flop_A23 \
    -input_port D -output_port Q -signal_type iso_enable
```

### 5.1.9 Unconnected Net Handling

VC LP honors the values for `handle_hanging_crossover` application variable while checking the applicability of isolation and level shifter strategies in hanging crossovers.

The `handle_hanging_crossover` can take any one of the following values:

- ❖ load: When set to load, VC LP discards crossover segments with hanging load.
- ❖ driver: When set to driver, VC LP discards crossover segments with hanging driver.
- ❖ both: When set to driver, VC LP discards crossover segments with both hanging driver and load

```
%vc_static_shell> set_app_var handle_handling_crossover driver/load/both
```

By default, this variable is set to both.

#### Note

For backward compatibility, true (equivalent to option both)/false values are also valid options. If any other option is provided the following error message on console will come: OPTION\_NOT\_VALID

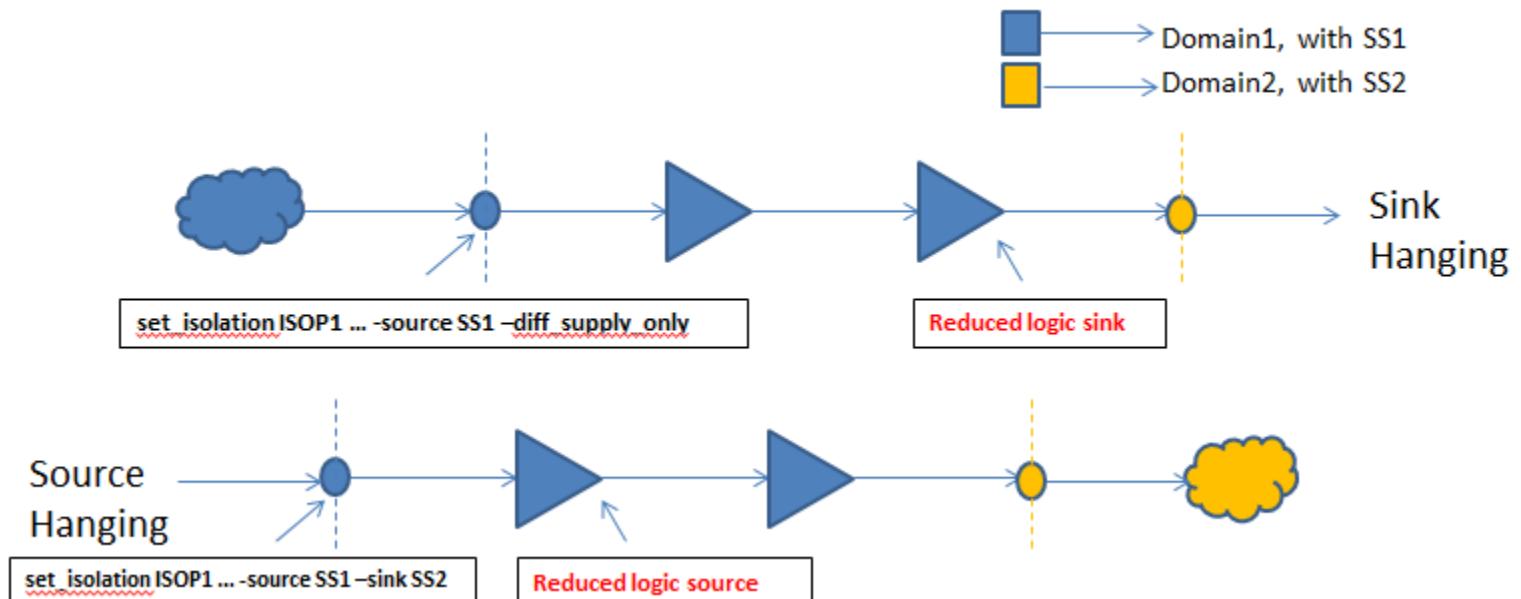
If the `handle_hanging_crossover` variables is set to be both/true:

- ❖ VC LP computes the reduced logic source or reduced logic sink for the hanging nets.
  - ◆ Any path based isolation policy that is beyond these newly computed logic source/sink gets applied according to the following rules:
    - ❖ If the isolation strategy has -sink and is placed beyond the reduced logic sink, it will be dropped.

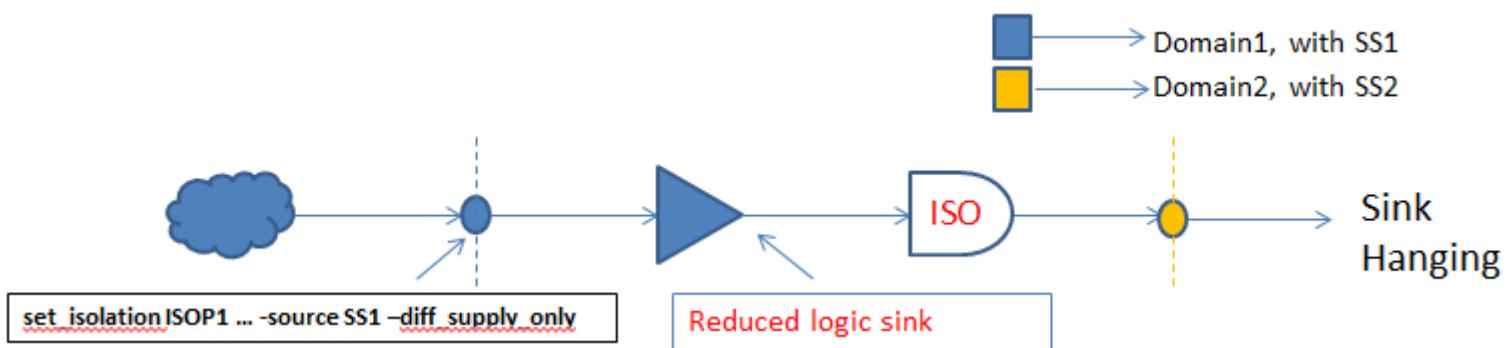
- ❖ If the isolation strategy has -source and is placed beyond the reduced logic source, it will be dropped.
- ❖ If the isolation strategy has -diff\_supply\_only and is placed beyond the reduced logic source/sink to which it has to compare the supply difference, the strategy will be dropped.
- ◆ VC LP checks the applicability of any other isolation strategies with respect to the newly computed reduced logic source/sink.

A reduced logic source/sink of a hanging logic source/sink is the closest logic (buffer/inverter) from the hanging logic source/sink as shown in [Figure 5-17](#) and [Figure 5-18](#).

**Figure 5-17 Handling Hanging Source Crossover - Policy Validity Checks**



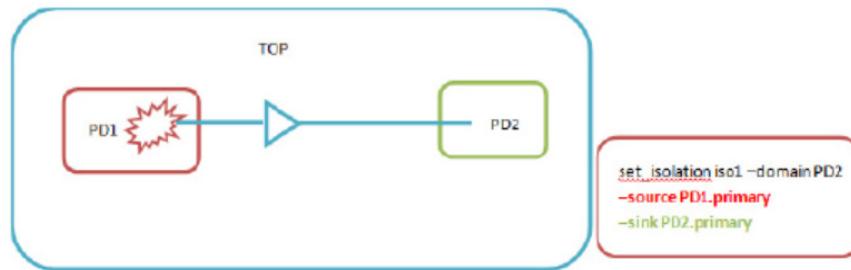
**Figure 5-18 Handling Hanging Sink Crossover - Policy Validity Checks**



### Case1: Isolation strategy present on hanging path

In the following case, the sink is hanging in the PD2 domain. If handle\_hanging\_crossover is set, the reduced sink is the immediate driver of the hanging net which is the buffer present in the TOP domain. The rest of checks happen based on the logic source (output of the glue logic present in the PD1 domain) and the reduced sink (input of the buffer present in the TOP domain).

**Figure 5-19 Isolation Strategy Present on Hanging Path**



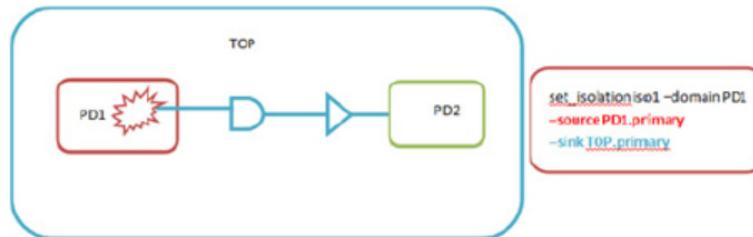
**Table 5-1 Isolation Strategy Present on Hanging Path**

Without handle_hanging_crossover variable	With handle_hanging_crossover variable
ISO_STRATEGY_UNUSED (PD1 -> PD2)	ISO_STRATEGY_MISSING (PD1 -> TOP)
ISO_INST_MISSING (PD1 -> PD2)	ISO_INST_MISSING (PD1 -> TOP)
ISO_STRATEGY_OK (PD1 -> PD2)	

### Case2: Isolation strategy and Isolation cell present on Hanging path

In the following case, the logic source is the output of the glue logic present in the PD1 domain and the reduced sink is the input of the buffer present in the TOP domain. Therefore, the strategy and the cell are associated to each-other.

**Figure 5-20 Isolation strategy and Isolation cell present on hanging path**



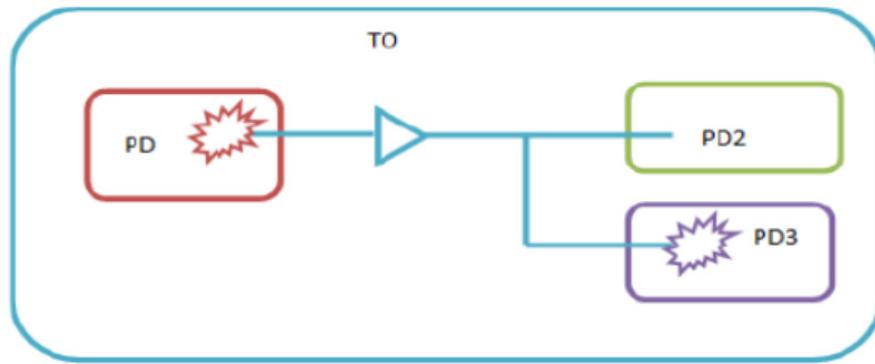
**Table 5-2 Isolation strategy and Isolation cell present on hanging path**

Without handle_hanging_crossover variable	With handle_hanging_crossover variable
ISO_STRATEGY_MISSING (PD1 -> PD2)	ISO_INST_OK (PD1 -> TOP)
ISO_INST_NOSTRAT (PD1 -> PD2)	ISO_STRATEGY_OK (PD1 -> TOP)
ISO_INST_OK (PD1 -> PD2)	

### Case3: Crossover generation for Hanging paths during fan-out

In this case, the sink is hanging in the PD2 domain. In such cases, only one crossover is generated from PD1 to PD3. As intermediate buffer present in TOP can NOT become the reduced sink for the path of PD1 to PD2 and part of the crossover for path of PD1 to PD3. Therefore, only one crossover is generated from the logic source (output of the glue logic present in the PD1 domain) to the logic sink (input of the glue logic present in the PD3 domain).

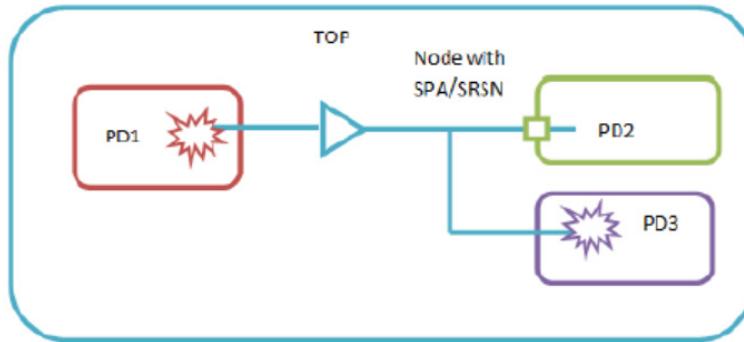
**Figure 5-21 Crossover Generation for Hanging Paths During Fan-out**



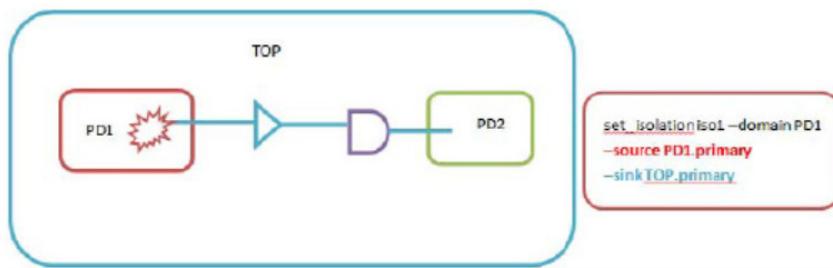
### Case4: Crossover generation with SPA/SRSN of hanging paths

In this case, the sink is hanging in the PD2 domain. SPA/SRSN is defined on the PD2 domain input node. So, VC LP does not reduce the path and it considers the PD2 node as the logical sink. Therefore, in this case, two crossovers are generated from PD1 to PD2 and PD1 to PD3.

**Figure 5-22 Crossover Generation With SPA/SRSN of Hanging Paths**



### Case5: Reduced sink is an isolation cell



In this case, the logic source (output of the glue logic present in the PD1 domain) and reduced sink are (input of an isolation cell present in the TOP domain) supplied by SS\_ISO supply.

- ❖ UPF Policy missing/redundancy related checks are NOT be done based on protection cells. UPF policy missing/redundancy is done based on the buffers/inverters only.
- ❖ Design electrical checks are done based on the protection cells.

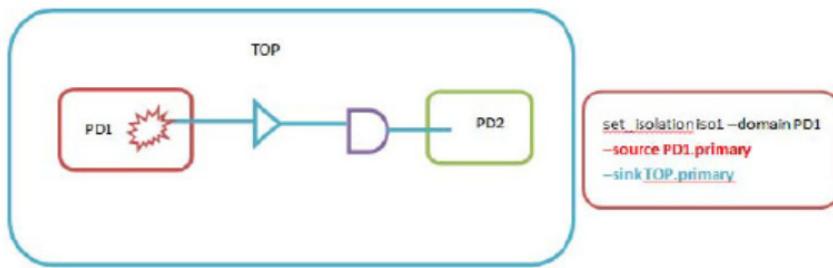
The `check_lp -stage upf` command is executed by considering the logic source is the output of the glue logic present in the PD1 domain and the reduced sink is the input of the buffer present in the TOP domain and are supplied by the SS\_TOP supply.

The `check_lp -stage design` command is executed by considering that the logic source is the output of the glue logic present in the PD1 domain and reduced sink is the input of the isolation cell present in the TOP domain but are supplied by SS\_ISO.

**Table 5-3 Reduced Sink is an Isolation Cell**

Without handle_hanging_crossover variable	With handle_hanging_crossover variable
ISO_STRATEGY_MISSING (PD1 -> PD2)	ISO_STRATEGY_OK (PD1 -> TOP)
ISO_INST_NOSTRAT (PD1 -> PD2)	ISO_INST_OK (PD1 -> SS_ISO)
ISO_INST_OK (PD1 -> PD2)	

**Figure 5-23 Reduced Sink is an Isolation Cell**



**Note**  
Similar to this case, checks happen based on the reduced source when logic source is hanging. Refer to Case6 for more details.

### Case6: Source is hanging and Isolation Strategy/Cell are present on path

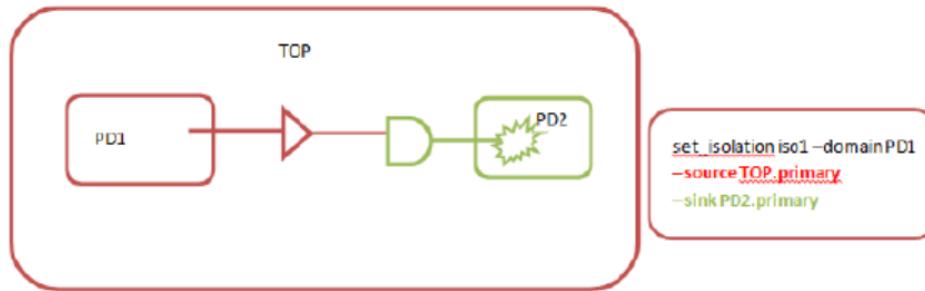
In this case, the logic source is hanging in the PD1 domain. If the `handle_hanging_crossover` application variable is set, the reduced source is the immediate load of the hanging net which is buffer present in the

TOP domain. The reduced source is the output of the buffer present in the TOP domain and the logic sink is the input of the glue logic present in the PD2 domain.

**Table 5-4 Source is Hanging and Isolation Strategy/Cell are Present on Path**

Without handle_hanging_crossover variable	With handle_hanging_crossover variable
ISO_STRATEGY_MISSING (PD1 -> PD2)	ISO_STRATEGY_OK (TOP -> PD2)
ISO_INST_NOSTRAT (PD1 -> PD2)	ISO_INST_OK (TOP -> PD2)
ISO_INST_OK (PD1 -> PD2)	

**Figure 5-24 Source is Hanging and Isolation Strategy/Cell are Present on Path**



### 5.1.9.1 Tags Affected by the handle\_hanging\_crossover Application Variable

The handle\_hanging\_crossover tag shortens the crossover paths. It changes the Logical Source/Sink of a crossover path. Therefore, it can affect all crossover based checks. It does not affect the following non-crossover based checks:

- ❖ Structural/Connectivity ISO/LS rules
- ❖ PSW (power switch) and RET (retention) tags

#### ISO\_\* tags

All rail order checks (gray cloud violations) are affected by the handle\_hanging\_crossover variable. The ISO\_\* tags typically depends on the PST states of the logical source/sink. As handle\_hanging\_crossover variable changes the logical source/sink of a crossover path, all ISO\_\* tags are affected.

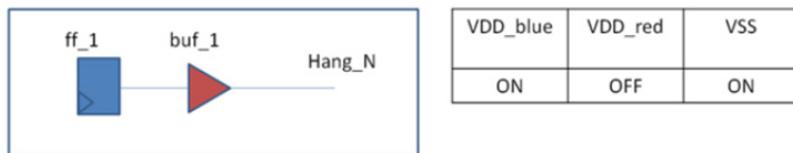
#### Impacted tags:

- ❖ All ISO\_\*\_STATE tags
- ❖ LS\_BUFINV\_VOLTAGE
- ❖ LS\_INPUT\_VOLTAGE
- ❖ All ISO\_\*\_FUNC tags
- ❖ All ISO\_\*\_WASTE tags
- ❖ RAIL\_MULTI\_\* tag

## Example

In the Figure 5-25, consider that *Hang\_N* belongs to a power domain which has power *VDD\_blue*. The ground net is same (*VSS*) for this example. If the tcl variable (*handle\_hanging\_crossover*) is not enabled, then you get *LS\_BUFINV\_VOLTAGE* message. If the Tcl variable is enabled, the reduced logic sink becomes the buffer (*buf\_1*) and no *ISO\** violation is issued if the parking checks are not enabled.

**Figure 5-25 Crossover Affects the ISO\_\* tags**



## The generate\_upf\_ctrl\_signal\_crossover Application Variable

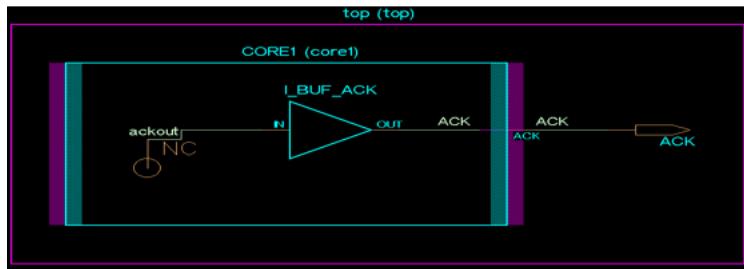
When the *generate\_upf\_ctrl\_signal\_crossover* application variable is set to true, VC LP generates the crossovers for the hanging control signals of the ISO/RET/PSW strategies. The default value of the *generate\_upf\_ctrl\_signal\_crossover* application variable is set to false.



**Note**  
Under the *lp\_enable\_virtual\_protection* application variable, function of *generate\_psw\_ctrl\_signal\_crossover* and *generate\_upf\_ctrl\_signal\_crossover* will also be enabled.

Consider the following path in design where CORE1 belongs to power domain PD1, and the top level module belongs to power domain PD\_TOP.

CORE1/ackout (hanging net, OFF) --> CORE1/ACK (Boundary port) --> ACK (top level port, ON)



### UPF Snippet

```
create_power_switch PSW1 -domain PD1 \
    -output_supply_port {....} \
    ....
    ....
    -ack_port {NSLEEPOUT1 ACK}
```

By default, VC LP does not generate crossovers for the mentioned path, and hence the *ISO\_STRATEGY\_MISSING*/*ISO\_INST\_MISSING* violations is not reported.

When the *generate\_upf\_ctrl\_signal\_crossover* application variable is set to true, VC LP generates the crossover, and reports the *ISO\_STRATEGY\_MISSING*/*ISO\_INST\_MISSING* violations for the same. The hanging net (CORE1/ackout in this case) resolves the supply from the PSW strategy (*input\_supply*). The *handle\_hanging\_crossover* application variable does not shorten this crossover when the *generate\_upf\_ctrl\_signal\_crossover* application variable is set to true.

### 5.1.9.1.1 Isolation Strategy Checks

All crossover based isolation strategy checks are affected by the *handle\_hanging\_crossover* variable. The tcl variable *handle\_hanging\_crossover* shortens a xover path. During this, it can drop the ISO strategies that are on the hanging crossover segment. During shortening, the logical source/sink of a

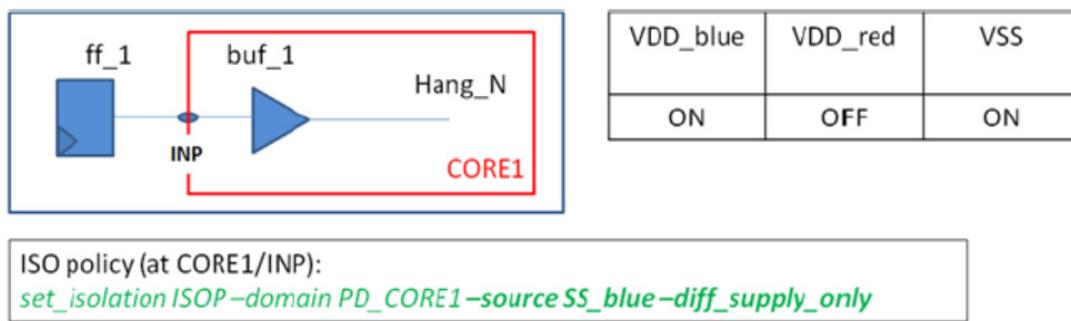
crossover path changes. Any path based the ISO strategy applicability also changes. Therefore, all the ISO strategy related tags are affected.

- ❖ All ISO\_STRAT\*\_\* tags
- ❖ All ISO\_MAP\_\* tags

### Example ISO Strategy Applicability Change

In the Figure 5-26, consider the *Hang\_N* belongs to a power domain which has power *VDD\_red*. The ground net is same (VSS) for this example. If the tcl variable (*handle\_hanging\_crossover*) is not enabled, then the ISO strategy at node *CORE1/INP* is applied. If the tcl variable is enabled, the reduced logic sink becomes the buffer (*CORE1/buf\_1*) and the ISO strategy at the node *CORE1/INP* is dropped.

**Figure 5-26 Crossover Affects Isolation Strategy Checks**



#### 5.1.9.1.2 Level Shifter Strategy Checks

All crossover based level shifter strategy checks are affected by the *handle\_hanging\_crossover* variable. The tcl variable *handle\_hanging\_crossover* shortens a crossover path. During this, it can drop LS strategies that are on the hanging crossover segment. Therefore, all the LS strategy related tags can get affected.

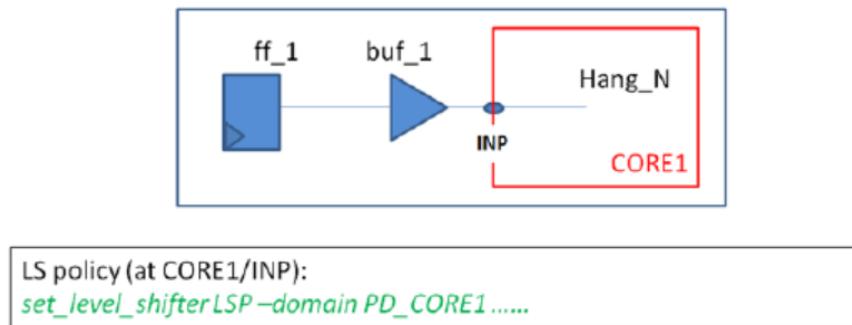
- ❖ All LS\_STRAT\*\_\* tags
- ❖ All LS\_MAP\_\* tags
- ❖ All LS\_\*\_CONN tags
- ❖ LS\_LOCATION\_\* tag

The *lp\_disable\_ls\_heterofanout* application variable is introduced to enable handling of heterogeneous fanout for path based level shifter strategy association.

When a policy associated node has heterogeneous fanout, and the strategy is applicable only on some paths, VC LP associates the strategies on those paths selectively. When the *lp\_disable\_ls\_heterofanout* application variable is enabled, VC LP does not associate source-sink based level shifter strategies on heterogeneous paths.

### Example LS strategy drop scenario

In the Figure 5-27, consider that the LS strategy *LSP* is applied on pin *CORE1/INP*. If the Tcl variable (*handle\_hanging\_crossover*) is not enabled, then the LS strategy at node *CORE1/INP* survives. If the tcl variable is enabled, the reduced logic sink becomes the buffer (*buf\_1*) and the LS strategy at node *CORE1/INP* is dropped.

**Figure 5-27 Crossover Based Level Shifter Strategy Checks**

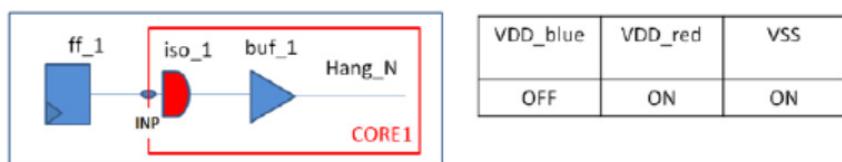
### 5.1.9.1.3 Isolation Instance Checks

All crossover based isolation instance checks are affected by the handle\_hanging\_crossover variable. The tcl variable handle\_hanging\_crossover shortens the crossover path. During this it can drop ISO strategies that are on the hanging crossover segment. During shortening, the logical source/sink of a crossover path gets changed. Any path based ISO strategy applicability is changed, and the ISO tags which uses the PST information of a logical source/sink is impacted. Therefore, all the crossover based ISO instance related tags is affected.

- ❖ All ISO\_INST\*\_\* tags
- ❖ All ISO\_MAP\_\* tags

### Example ISO instance tag scenario

In the Figure 5-28, consider that *Hang\_N* belongs to a power domain which has power *VDD\_red*. The ground net is same (*VSS*) for this example. If the tcl variable (handle\_hanging\_crossover) is not enabled, then the logic sink becomes *Hang\_N* and ISO instance *iso\_1* becomes relevant (non redundant). If the tcl variable is enabled, the reduced logic sink becomes the buffer (*CORE1/buf\_1*) and the ISO instance *iso\_1* becomes redundant.

**Figure 5-28 Crossover Based Isolation Instance Checks**

### 5.1.9.1.4 Level Shifter Instance Checks

The handle\_hanging\_crossover application variable shortens the crossover path. While shortening the crossover path, it can drop the LS strategies that are on the hanging crossover segment. During shortening, the Logical Source/Sink of a crossover path is changed. Therefore, the LS tags which uses PST information of Logical Source/Sink is impacted.

The following crossover based LS instance related tags are affected:

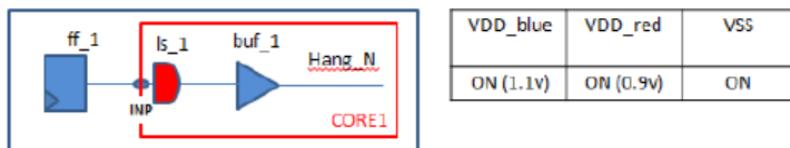
- ❖ All LS\_INST\*\_\* tags
- ❖ All LS\_MAP\_\* tags
- ❖ All LS\_\*\_CONN tags

- ❖ LS\_LOCATION\_\* tag

### Example LS instance tag scenario

In the Figure 5-29, consider that *Hang\_N* belongs to a power domain which has power *VDD\_red*. The ground net is same (*VSS*) for this example. If the tcl variable (*handle\_hanging\_crossover*) is not enabled, then the logic sink becomes *Hang\_N* and LS instance *ls\_1* becomes relevant (non redundant). If the tcl variable is enabled, the reduced logic sink becomes the buffer (*CORE1/buf\_1*) and the LS instance *ls\_1* becomes redundant.

**Figure 5-29 Crossover Based Isolation Instance Checks**



#### 5.1.9.1.5 Support for lp\_ignore\_hanging\_for\_combo\_checks Application Variable

Previously, the ISO\_COMBO\_FUNC/WASTE checks were for hanging paths were controlled by the *handle\_hanging\_crossover* application variable. The ISO\_COMBO\_FUNC/WASTE checks for hanging paths are controlled by the *lp\_ignore\_hanging\_for\_combo\_checks* application variable.

By default, the *lp\_ignore\_hanging\_for\_combo\_checks* application variable is set to false.

- ❖ When the *lp\_ignore\_hanging\_for\_combo\_checks* application variable is false, the ISO\_COMBO\_FUNC/WASTE checks are performed for hanging paths.

When the *lp\_ignore\_hanging\_for\_combo\_checks* is true, ISO\_COMBO\_FUNC/WASTE checks are not performed for hanging paths.

#### 5.1.9.2 Tags that do not get Affected by the handle\_hanging\_crossover Application Variable

The following are the non-crossover tags that are not affected when the *handle\_hanging\_crossover* variable is enabled:

- ❖ All non-crossover based isolation instance checks:
  - ◆ ISO\_INST\_CLAMPED
  - ◆ ISO\_INST\_TRANSPARENT
  - ◆ ISO\_CONTROL\_GLITCH
  - ◆ ISO\_OUTPUT\_UNCONN
  - ◆ ISO\_DATA\_CONSTANT
  - ◆ ISO\_ENABLE\_UNCONN
  - ◆ ISO\_DATA\_UNCONN
  - ◆ ISO\_ENABLE\_EXPR
  - ◆ ISO\_ENABLE\_MISSING
- ❖ All non-crossover based level shifter instance checks:
  - ◆ LS\_INPUT\_TIEHI
  - ◆ LS\_INPUT\_TIELO

- ◆ LS\_OUTPUT\_UNCONN
- ◆ LS\_INPUT\_UNCONN
- ❖ All RET\_\* tags
- ❖ All PSW\_\* tags

### 5.1.9.3 Reporting Xovers/Paths Ignored by VC LP

VC LP lists the Xover paths which are dropped or ignored from analysis at various stages of Xover generation due to different reasons. For example, hanging/floating paths are ignored for analysis, Xover path is not considered when design ends on a Stub module.

To enable this feature, use the `enable_lp_dump_debug_reports enable_report_ignored_xover_paths` command.

```
enable_lp_dump_debug_reports enable_report_ignored_xover_paths
```

The dropped Xovers are reported in the *IgnoredXoverPaths.rpt* file. This file is created in the `vcst_rtdb/reports` directory.

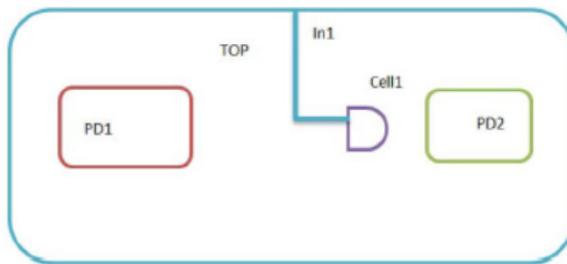
The file contains the reason for the Xover being dropped, along with the details of the source and sink of the dropped path. It also contains the name of the Tcl variable which causes the Xover to be dropped, whenever applicable.

This feature only supports the following reasons for which the Xovers are dropped:

- ❖ Reasons caused because of `handle_hanging_crossover` Application variable
  - ◆ . Hanging Source/Sink or Both
    - ❖ Xover path segments dropped (not entire path) due to a hanging source/sink
    - ❖ Entire Xover path is dropped due to a hanging sink
    - ❖ Entire Xover path is dropped due to a hanging source
- ❖ Reasons caused because of `handle_hanging_crossover` Application variables
  - ◆ Unconnected driver and no AON/Protection cell/op pin found on path
- ❖ Reasons caused because of `handle_hanging_crossover` Application variables
  - ◆ Unconnected load/sink node
- ❖ Reasons not affected by Application variables
  - ◆ Root node is the sink node
  - ◆ No power domain boundary/AON cell/Protection Device/SRSN/SPA Encountered during the path traversal
  - ◆ Invalid same domain Xover
  - ◆ Sink node is a pin of a Diode or Physical-only cell

#### Example 1

In [Figure 5-30](#), the TOP domain input port In1 is connected to Cell1 input pin, which is also in the TOP domain. VC LP ignores this Xover due to the same domain Xover.

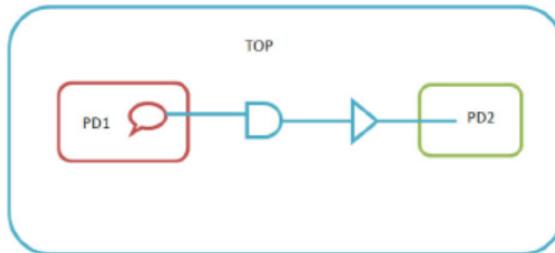
**Figure 5-30 Example for Reporting Invalid Same Domain Xover**

The details reported in the IgnoredXoverPaths.rpt file for this scenario is as follows:

```
-----  
REASON: Invalid same domain Xover  
XOVER PATH: Dropped SourceNode: In1 Dropped SinkNode: Cell1/A
```

### Example 2

In [Figure 5-31](#), the LogicSource is the output of the glue logic present in PD1 domain and the reduced sink is output of the buffer present in the TOP domain. Xover is reported as the buffer output to PD2 domain input is dropped.

**Figure 5-31 Example for Xover Dropped due to Hanging Source/Sink or Both**

The details reported in the IgnoredXoverPaths.rpt file for this scenario is as follows:

```
-----  
REASON: Hanging Source/Sink or Both, Tcl var: handle_hanging_crossover,  
ignore_unconnected_driver, ignore_unconnected_load  
INFO: ignore_unconnected_load is turned ON by tool whenever handle_hanging_crossover is  
ON  
  
LIST OF XOVER PATHS WITH SEGMENTS DROPPED DUE TO HANGING SOURCE/SINK OR BOTH  
XOVER PATH: SourceNode: i_core1/out1 Reduced SinkNode: buf/Z Dropped SinkNode:  
i_core2/in
```

## 5.1.10 Enhanced Hard Macro Handling

### 5.1.10.1 Isolation Checks

Some hard macro cells are represented by liberty models and they have multiple inputs or outputs that are isolated. The liberty syntax contains the `isolation_enable_condition` attribute which can be placed on an input or output. You can use this information to perform many types of checks in VC LP as described in the subsequent sections.

In the following sections, a hard macro means a cell which has a liberty description containing the `is_macro_cell` attribute set to true.

### The ISO\_ENABLE\_EXPR Cell Check

VC LP issues a warning if the `isolation_enable_condition` involves more than one signal. The following is an example of a pin with a complex `isolation_enable_condition`:

```
pin (y) {
    direction : output;
    is_isolated : true;
    isolation_enable_condition : "a & b";
}
```

VC LP generates an error if the enable condition is more complex than a single term with optional inversion. One message is generated for each unique cell (not each instance) in the design with this condition. The message is an error issued during the check design stage as follows:

```
Tag      : ISO_ENABLE_EXPR
Description : Isolation enable condition for cell pin [CellPin]
             of [Cell] is not a single pin
CellPin   : Z[0]
Cell      : MYMAC
```

This message be disabled by default. To enable it, use the `configure_tag -app LP` command.

#### 5.1.10.1.1 Extended Isolation Pin Checks

VC LP issues errors for messages which already exist on pins of standard isolation cells in the following scenarios:

- ❖ If the isolation enable pin of a hard macro, which is internally isolated, is unconnected.
- ❖ If the isolation enable pin of a hard macro, which is internally isolated, is connected to a constant (1'b0 or 1'b1).

The following messages are issued for these pins:

- ❖ Tag: ISO\_DATA\_UNCONN  
Description: Isolation data pin unconnected
- ❖ Tag: ISO\_INST\_TRANSPARENT  
Description: Isolation instance is transparent
- ❖ Tag: ISO\_INST\_CLAMPED  
Description: Isolation instance is clamped
- ❖ Tag: ISO\_ENABLE\_UNCONN  
Description: Isolation enable pin unconnected
- ❖ Tag: ISO\_OUTPUT\_UNCONN  
Description: Isolation output pin unconnected

VC LP issues the following messages for instance pins of hard macro cells for the following conditions.

- ❖ If a pin is mentioned in any `isolation_enable_condition` for the cell or the pin has the attribute `isolation_cell_enable_pin : true`, then:
  - ◆ If the pin is unconnected, VC LP issues ISO\_ENABLE\_UNCONN.

- ◆ If the pin is tied to a constant such as 1'b1 and 1'b0, VC LP issues ISO\_INST\_TRANSPARENT or ISO\_INST\_CLAMPED respectively.
- ❖ If the pin has `is_isolated : true` and `direction : output` and it is not connected, VC LP issues ISO\_OUTPUT\_UNCONN
- ❖ If the pin has `is_isolated : true` and `direction : input` and it is not connected, VC LP issues ISO\_DATA\_UNCONN
- ❖ If the pin has `isolation_data_pin : true` and it is not connected, VC LP issues ISO\_DATA\_UNCONN

### 5.1.10.1.2 The `set_hardmacro_isolation` Command

VC LP provides the `set_hardmacro_isolation` command to check the intended connection and polarity of a net, against the actual connection in the design and the actual polarity of the library cell. This information is used to extend existing ISO\_CONTROL\_CONN, ISO\_CONTROL\_INVERT checks, covering the following:

- ❖ If the signal specified in `isolation_enable_condition` is not connected to the signal specified by `-isolation_signal` in UPF, then an error is issued.
- ❖ If the signal polarity specified in `isolation_enable_condition` does not match with the signal polarity specified by `-isolation_sense` in UPF, then an error is issued.

## Use Model

```
set_hardmacro_enable isolation -elements {list_of_elements} \ -isolation_signal
<signal_name> \ -isolation_sense <high/low>
```

### Options:

- ❖ `-elements {list_of_elements}`: Mandatory option, to specify the boundary ports, to which the signal and polarity information applies. Wild-card is not supported. User has to specify the full hierarchical design name.
- ❖ `-isolation_signal <signal_name>`: Mandatory option, to specify isolation enable signal name.
- ❖ `-isolation_sense <sense_value>`: Mandatory option, takes high or low as values.

When the command is executed, error checking performed. If an element does not exist, or if an invalid `isolation_sense` is given, an error is issued and the command is rejected. If the net named in `isolation_signal` does not exist, no error is generated at this stage. However, all of the elements associated with the incorrect connection and the ISO\_CONTROL\_CONN message are generated during the check design stage.

## Example

Here is an example liberty, design diagram and Tcl script which generates several errors.

```
cell (ISOMAC) {
    pin(da)  { direction : output; isolation_enable_condition: ea; }
    pin(db)  { direction : output; isolation_enable_condition: !eb; }
    pin(dc1) { direction : output; isolation_enable_condition: ec; }
    pin(dc2) { direction : output; isolation_enable_condition: ec; }
}
set_hardmacro_isolation -elements u1/u2/da \
```

```

-isolation_signal u1/na -isolation_sense high
set_hardmacro_isolation -elements u1/u2/db \
-isolation_signal pb -isolation_sense high
set s {u1/u2/dc1 u1/u2/dc2}
set_hardmacro_isolation -elements $s \
-isolation_signal pc -isolation_sense high

```

Comparing these three items, you can see the following two errors:

- ❖ In the second line of the Tcl, macro pin *db* is associated with macro enable pin *eb* and the polarity is given as *high*. However in the liberty, pin *eb* is active low (the function is *!eb*). This results in an incorrect polarity ISO\_CONTROL\_INVERT message.
- ❖ In the last line of the Tcl, macro pin *dc1* and *dc2* are associated with macro enable pin *ec* and the connected net is intended to be *pc*. However, in the design the pin is actually connected to *pd*. This results in an incorrect ISO\_CONTROL\_CONN connection message.
- ❖ The fact that *pc* does not exist in the design is not an error; the third line shows the use of a Tcl variable to give a list.

The following error messages appear in the output of the report\_violations -app LP command:

```

Tag : ISO_CONTROL_INVERT
Description : Strategy [Strategy] isolation control
              signal [UPFNet] is opposite polarity from
              isolation instance [Instance]: [SenseMismatch] UPFNet

UPFNet
  NetName : pb
  NetType : Design
  Instance : u1/u2
  Cell : ISOMAC
  SenseMismatch
    MisSenseDesign : LOW
    MisSenseUpf : HIGH
  StrategyNode : u1/u2/db

Tag : ISO_CONTROL_CONN
Description : Strategy [Strategy] isolation control signal
              [UPFNet] does not match isolation instance
              [Instance] connection pin [CellPin] UPFNet

UPFNet
  NetName : pc
  NetType : UPF
  Instance : u1/u2
  Cell : ISOMAC
  CellPin : ec
  DesignNet
    NetName : pd
    NetType : Design
  StrategyNode : u1/u2/dc1
  Source
    PinName : pd
  SegmentSourceDomain : d_top
  Sink : u1/u2/ec
  SegmentSinkDomain : d_top
  SourceInfo

```

```
PowerNet ...
SinkInfo
PowerNet ...
```

### 5.1.10.1.3 Limitations

Architectural checks and `ISO_CONTROL_GLITCH` are not supported for hard macro isolation enable pin.

### 5.1.10.2 Handling Macros with Built-in LS

For VC LP to use the level shifters in macro inputs:

- ❖ Powering of internal LS must be specified either through primary or internal supply ports on the macro.
- ❖ PST states for these ports must be defined on the UPF power intent.

VC LP supports the following:

- ❖ The level shifters internal to macro cells, connected directly to its input signal pins
- ❖ Both dual-rail Level Shifters and single-rail (over-driven) Level Shifters, for internal level shifting of macro input pins
- ❖ Enables you to specify a voltage tolerance range for the level shifted input pins (only for single-rail internal LS)

### 5.1.10.2.1 Relevant lib-cell and lib-pin Library Attributes

The following relevant library attributes for LS cells are required on the macro input pins with internal LS.

- ❖ Related power: `related_power_pin` and `related_ground_pin`
- ❖ Voltage range: `input_voltage_range` and `output_voltage_range`
- ❖ Signal level: `input_signal_level` and `output_signal_level`

#### The `related_power_pin` and `related_ground_pin` Attributes

The `related_power_pin` and `related_ground_pin` attributes are used to obtain the related supply of each pin, and are important for all MV processing: LS/ISO insertion/checking, AO legalization etcetera.

The macro input pins with internal LS must have these attributes defined on the pin. If the internal LS is of dual-rail type, the related power pins are the ones that power the input side of the LS (this is consistent with the current usage of the attribute: see next paragraph). If the internal LS is of single-rail type, the related power pins are the ones which power the output side of the LS. For this last case, the usage of this attributes are restricted for ISO/AO purposes, but LS insertion/checking uses the other attributes in most cases.

For macro input pins that are not level-shifted (and also for output macro pins), the usage of `related_power_pin` and `related_ground_pin` are the same as for any other pin: it specifies the supply net (indirectly, through the `pg_pin`) which powers the cell (or part of it) that implements the pin. This information is necessary for MV processing, because from that the correct driver/load voltage and the possible on/off states are gathered.

#### The `input_voltage_range` and `output_voltage_range` Attributes

The `input_voltage_range` and `output_voltage_range` attributes specify the range of voltages in which the input and output pins (respectively) of a LS cell is designed to operate safely. They are currently used for LS insertion/checking.

For macro input pins with internal LS, the `input_voltage_range` attribute must be defined on the pin. The usage of this attribute is similar to the current usage on stand-alone LS cells.

Note that, if a not internally level-shifted macro input pin would have `input_voltage_range` attribute defined, then VC LP assumes the presence of an internal LS and that may lead to some messages about internal LS, but the electrical correctness of the tool behavior would still be OK. When LP assumes that the presence of `input_voltage_range` on a macro input pin, it implies that the pin is internally level shifted, either by a single-rail or a dual-rail LS. The absence of attribute `input_signal_level` means that the internal LS cell is dual-rail type. The printed messages indicates the absence of the `input_signal_level` attribute.

The LIB\_PINATTR\_MISSING violation is reported for LS or ELS cells with missing `input_voltage_range` and `output_voltage_range` attributes. The `input_voltage_range` and `output_voltage_range` is both a cell level as well as a pin level attribute. If the LS and ELS cell do not have the `input_voltage_range` and `output_voltage_range` attribute either at the pin level or the cell level, the LIB\_PINATTR\_MISSING violation is reported.

The LIB\_PINATTR\_MISSING violation is not reported if attribute `related_power_pin/related_ground_pin` is not defined on port with attribute `antenna_related_power_pin/antenna_related_ground_pin` in library check.

The LIB\_PINATTR\_MISSING is not reported for the enable pins in ELS.

The following is an example output of the LIB\_PINATTR\_MISSING violation:

```
Tag : LIB_PINATTR_MISSING
Description : Attribute [Attribute] of library cell [Cell] pin [CellPin] is
missing
Attribute : input_voltage_range
Cell : LSHL
CellPin : I
LibCellType : protection
ReasonCode : input_voltage_range is missing
```

By default, along with the `input_voltage_range`, the `input_signal_level` is also required to trigger analysis of the `input_voltage_range`.

However, if `input_voltage_range` defined for a pin, and if the voltage range is the only thing required to determine a level shifter violation, set the `lp_level_range_trigger` application variable to false. By default the `lp_level_range_trigger` application variable is set to true.

## The `input_signal_level` and `output_signal_level` Attributes

The `input_signal_level` and `output_signal_level` attributes are introduced for single-rail LS to account for the functional mismatch of the `related_power_pin` and `related_ground_pin` attributes of the input data pin. Since the cell has only one power and ground pg\_pin, both input and output related attributes are pointing to it. Given that overdriven LS can be driven by a voltage other than the one it's powered, the input side must be de-coupled from that pg-pin, and then we need a way to specify an alternative voltage level for MV linking and opcond checking purposes.

Currently tools do not use `output_signal_level` attribute for level shifter cells, since existing single-rail overdriven LS cells have only one pg\_pin (one power and one ground) and it's functionally related to the output signal pin. Only the `input_signal_level` attribute has significance on LS cells.

Macro input pins with internal LS of single-rail (over-driven) type has this attribute defined. Moreover, the presence of this attribute in a macro input pin (which also has `input_voltage_range` defined) implies

that the pin is internally level shifted by a single-rail LS. Just the attribute presence is used as a marker to know that the type of the internal LS is single-rail.

### 5.1.10.3 Support for Control Path Checks for Partially Instrumented Hard Macro Cells

VC LP performs electrical checks on control paths for isolation, power switch and retention based on the control signals defined in the UPF strategy. However, for partially instrumented macro cells that are instantiated in the early stages of RTL, with the protection cell inside the macro and strategy undefined, VC LP does not perform the electrical checks. To enable this support, set the `lp_check_hardmacro_control_signals` application variable to true.

- ❖ Isolation Control

Isolation control pin of the macro cell is identified by the `isolation_enable_condition` defined on the `is_isolated` pin of the cell. The path connecting the isolation control pin is considered as the isolation enable path and check will be done. `ISO_CONTROL_STATE` and `ISO_CONTROL_VOLTDIFF` violations are enhanced to support these scenarios.

- ❖ Power Switch Control:

Power switch control pin is identified by the "*switch pin*" pin level attribute. The `PSW_CONTROL_STATE` violation is enhanced to support these scenarios.

- ❖ Retention Control:

Retention control pin is identified by the "*retention pin*" pin level attribute. The value of the attribute defines the type of control pin, {SAVE, RESTORE}. The `RET_CONTROL_STATE` and `RET_CONTROL_VOLTDIF` violations are enhanced to support these scenarios.

In addition, the supply of the control signal is inferred from the RPP/RGP of the enable pin identified.

## 5.1.11 Power Switch Checks

### 5.1.11.1 Functional Checks on Ack Nets of Power Switch Policies

VC LP performs functional checks on ack nets of power switch policies. In UPF2.0 power switch specification, a boolean function may be specified along with `ack_port`. The following is a typical example of a power switch specification with the `ack_port` boolean function:

```
create_power_switch PSW
-domain PD
-input_supply_port {VDDT VDDV}
-output_supply_port {VDDC VDDSW}
-control_port {ES PMU/PwrOn }
-on_state {sw_on VDDT {!ES}}
-ack_port {ACK net_ack {!ES}}
```

This example specifies that the logic value at the `net_ack` should be true whenever the control signal `ES` assumes logic value false; false when `ES` assumes the logic value of true.

During the place and route (P&R) phase, design tools might insert some logic between the actual control signal that controls the power switch (PMU/PwrOn in the example) and the net connected to the ack port (`net_ack` in the example). This is required to meet timing requirements in the path and maintain `on_state/ off_state` correctness for switches placed in between. However, if the functionality specified in `ack_port` function is not implemented correctly in the design, whenever the present module is

instantiated as an IP or a soft-block at the chip level might result in functional issues in the design. Using VC LP you can catch such scenarios and flag them as issues/warnings.

As per UPF 2.0, if the `ack_port` boolean function is specified, the result of the boolean expression should be driven on the `ack_port`. If the boolean function is not specified, logic 1 should be driven on the port when `on_state` evaluates to true and logic 0 should be driven on the port when `off_state` evaluates to true.

VC LP verifies and reports a scenario where the `off_state` expression is a complement of the `on_state` expression. Therefore, when the `ack_port` boolean expression is not specified, it is sufficient to check if the resultant expression on the acknowledge net matches the `on_state` expression.

During the PSW functional checks, VC LP prints the expanded `on_state` expression of the policy based on its fan-in connections. In some cases, the number of variables in the expression are higher and the shell hangs while printing. This is not an ideal scenario. The `PSW_FUNC_EXPR_LIMIT` violation is reported in such cases whenever the implemented expression becomes huge and restrict printing the expression to acceptable limit.

In such cases, disable the `lp_full_expression_detail` application variable so that the concise expressions are printed.

The `lp_full_expression_detail` application variable is disabled by default.

#### 5.1.11.1.1 PSW Checks

VC LP reports the following violations in case of a match or a mis-match:

##### **PSW\_ACKFUNCTION\_WRONG**

This tag is reported when the design expression of the specified ack port/net does not match the ack port boolean expression specified in UPF.

- ❖ Severity: Error
- ❖ Message: The `ack_port` Boolean expression of the power switch strategy [Strategy] is implemented incorrectly at the acknowledge net [DesignNet]
- ❖ Fields:
  - ◆ Strategy: <Strategy Name>
  - ◆ DesignNet: <AckPort Name>
  - ◆ StratAckExpr: <Boolean expression in policy>
  - ◆ DesignExpression: <Boolean expression in design>
  - ◆ CounterExample: <Boolean expression for counter example>

##### **PSW\_ACKFUNCTION\_OK**

This tag is reported when the design expression of the specified ack port/net matches with the ack port boolean expression specified in the UPF.

- ❖ Severity: Info (generated when the `check_lp -stage pg -include_info` option is specified)
- ❖ Message: The `ack_port` Boolean expression of the power switch strategy [Strategy] is implemented correctly at the acknowledge net [DesignNet]
- ❖ Fields:
  - ◆ Strategy: <Strategy Name>

- ◆ DesignNet: <AckPort Name>
- ◆ StratAckExpr: <Boolean expression in policy>
- ◆ DesignExpression: <Boolean expression in design>

## PSW\_ACKFUNCTION\_UNDETERMINABLE

This tag is reported when the design expression of the specified ack port/net cannot be verified. You cannot determine if implementation matches or not with the specification in UPF.

- ❖ Severity: Warning
- ❖ Message: The ack\_port Boolean expression of the power switch strategy [Strategy] cannot be verified
- ❖ Fields:
  - ◆ Strategy: <Strategy Name>
  - ◆ DesignNet: <AckPort Name>
  - ◆ StratAckExpr: <Boolean expression in policy>
  - ◆ DesignExpression: <Boolean expression in design>
  - ◆ CounterExample: <Boolean expression for counter example>

## PSW\_ACKFUNCTION\_CONT

This tag is reported in cases where the design expression of the specified ack port/net contains the ack port boolean expression specified in UPF.

- ❖ Severity: Warning
- ❖ Message: The implemented Boolean expression at the acknowledge net [DesignNet] of the power switch strategy [Strategy] contains the Boolean expression specified for the ack\_port
- ❖ Fields:
  - ◆ Strategy: <Strategy Name>
  - ◆ DesignNet: <AckPort Name>
  - ◆ StratAckExpr: <Boolean expression in policy>
  - ◆ DesignExpression: <Boolean expression in design>

### 5.1.11.1.2 Error and Warning Messages

The following error or warning messages are issued in various scenarios:

## PSW\_FUNC\_MULTI\_DRIVEN\_NET

This message is issued if multi-driven nets are found during the ack function check. As a static tool, VC LP cannot resolve the symbol on a multi-driven net, therefore, it creates a new symbol upon issuing this message. All the power switch instances in the fan-out path will have extra symbols in their design expression due to this multi-driven net and hence those switches are likely to have PSW\_ACKFUNCTION\_WRONG violations. This is a soft error.

## PSSTRAT\_UNCONN\_ACKPORT

This message is issued if there is no driving net attached to the design object specified as an ack net or, no immediate driver (could be an hierarchical boundary object) to drive the net attached to the design object specified as the ack net. This is a soft error.

## PSSTRAT\_NOACKPORT\_FUNCTION

This message is issued if there is no boolean function specified to the ack\_port of the policy, and the OnState boolean expression is assumed for ack port functional check. This is only a warning.

## PSSTRAT\_SLICE\_ACKNET

The design object specified as an ack net in the strategy cannot be a multi-bit slice of a vector. In such cases, the ack port functional check cannot be done. This is a soft error.

### 5.1.11.1.3 Examples

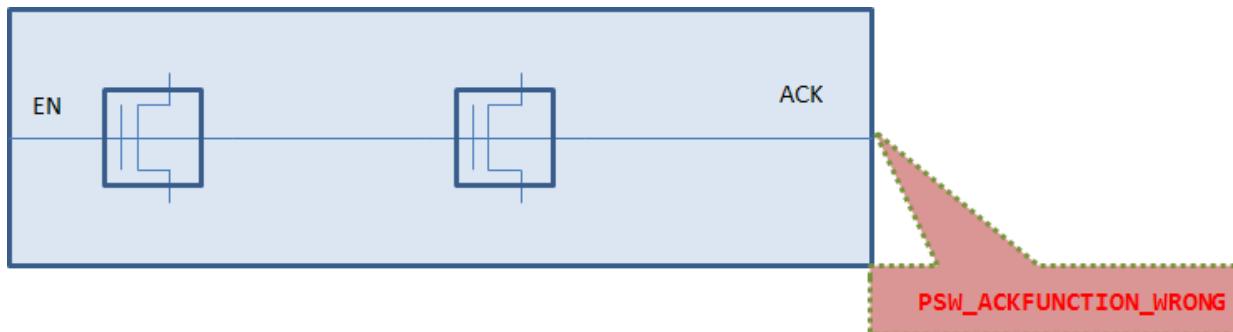
#### Example for PSW\_ACKFUNCTION\_WRONG

##### UPF Specification:

```
create_power_switch psw-domain PD \
-input_supply_port { VDD VDD } \
-output_supply_port { VDDO VDDS } \
-control_port { EN EN } \
-ack_port { ACK ACK { !EN } } \
-on_state { SW_ON VDD { !EN } }
```

##### Design Implementation:

**Figure 5-32 Design Implementation for PSW\_ACKFUNCTION\_WRONG**



According to UPF specification, the logic value at an ack net ACK should be true whenever the control signal EN assumes the logic value of false, and false when EN assumes the logic value of true. So, in the design there should be an inversion in path of the ack and control net. As illustrated in Figure 5-32, the design expression of ACK is EN which does not match with the UPF specified expression !EN. Hence, PSW\_ACKFUNCTION\_WRONG is reported for the ack net ACK.

#### Example for PSW\_ACKFUNCTION\_OK

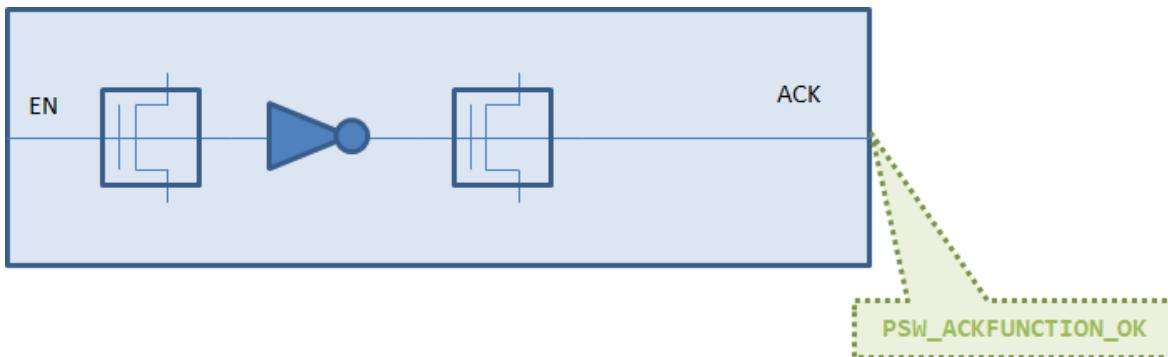
##### UPF Specification:

```
create_power_switch psw-domain PD \
-input_supply_port { VDD VDD } \
-output_supply_port { VDDO VDDS } \
```

```
-control_port { EN EN } \
-ack_port { ACK ACK { !EN } } \
-on_state { SW_ON VDD { !EN } }
```

### Design Implementation:

**Figure 5-33 Design Implementation for PSW\_ACKFUNCTION\_OK**



According to UPF specification, the logic value at the ack net ACK should be true whenever the control signal EN assumes the logic value of false, and false when EN assumes the logic value of true. So, in the design there should be inversion in path of the ack and control net. As illustrated in Figure 5-33, the boolean expression of the ack net works out to be  $\text{!EN}$  which matches with the UPF specified expression  $\text{!EN}$ . Hence PSW\_ACKFUNCTION\_OK is reported for the ack net ACK.

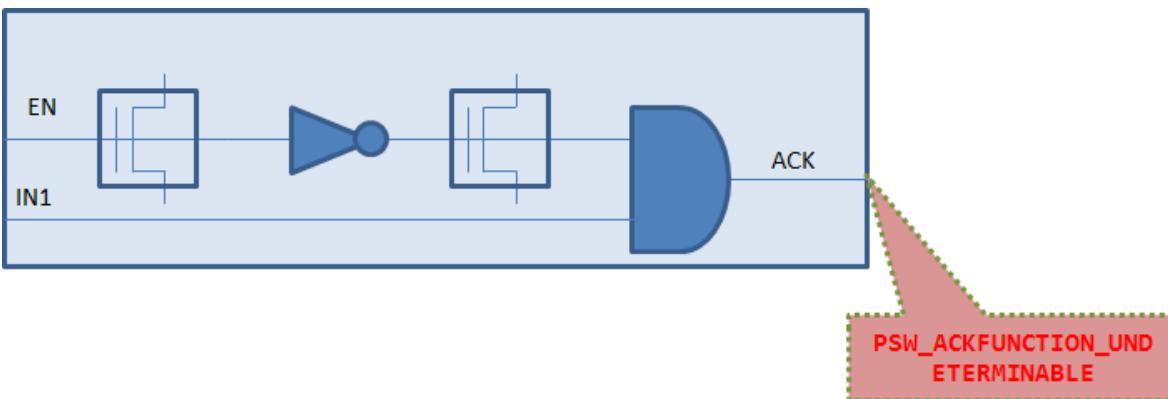
### Example for PSW\_ACKFUNCTION\_UNDETERMINABLE

#### UPF Specification:

```
create_power_switch psw-domain PD \
-input_supply_port { VDD VDD } \
-output_supply_port { VDDO VDDS } \
-control_port { EN EN } \
-ack_port { ACK ACK { !EN } } \
-on_state { SW_ON VDD { !EN } }
```

#### Design Implementation

**Figure 5-34 Design Implementation for PSW\_ACKFUNCTION\_UNDETERMINABLE**



According to the UPF specification, the logic value at an ack net ACK should be true whenever the control signal EN assumes the logic value of false, and false when EN assumes the logic value of true. So, in the design there should be inversion in the path of the ack and control net. As illustrated in [Figure 5-34](#), the boolean expression of the ack net works out to be EN and IN1. In this case, the polarity of ack net cannot be determined. Hence, PSW\_ACKFUNCTION\_UNDETERMINABLE is reported for the ack net ACK.

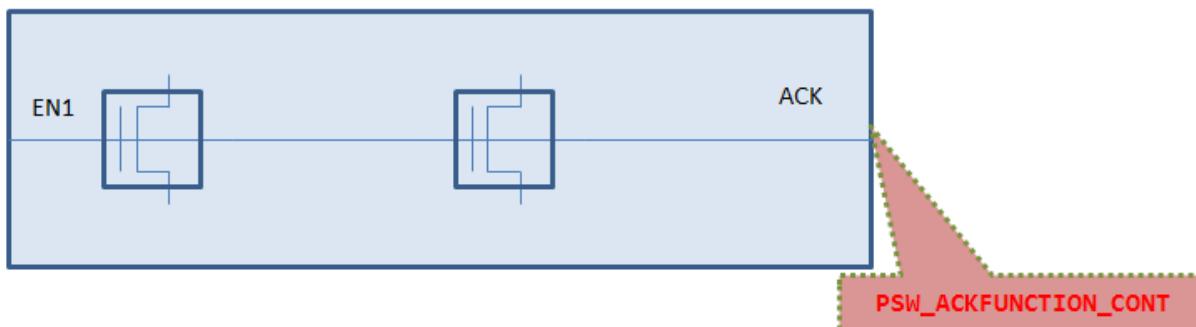
### Example for PSW\_ACKFUNCTION\_CONT

UPF Specification:

```
create_power_switch psw-domain PD \
-input_supply_port { VDD VDD } \
-output_supply_port { VDDO VDDS } \
-control_port { EN1 EN1 } \
-control_port { EN2 EN2 } \
-ack_port { ACK ACK { EN1 & EN2 } } \
-on_state { SW_ON VDD { EN1 & EN2 } }
```

**Design Implementation:**

**Figure 5-35 Design Implementation for PSW\_ACKFUNCTION\_CONT**



According to UPF specification, the logic value at an ack net ACK should be true whenever the control signals EN1 and EN2 assume the logic value of true, and false when either EN1 or EN2 control ports assume the logic value of false. As illustrated in [Figure 5-35](#), the boolean expression of the ack net ACK works out to be EN1. In this case, the design expression of the ack net contains the UPF specified expression EN1 and EN2. Hence, PSW\_ACKFUNCTION\_CONT is reported for the ack net ACK.

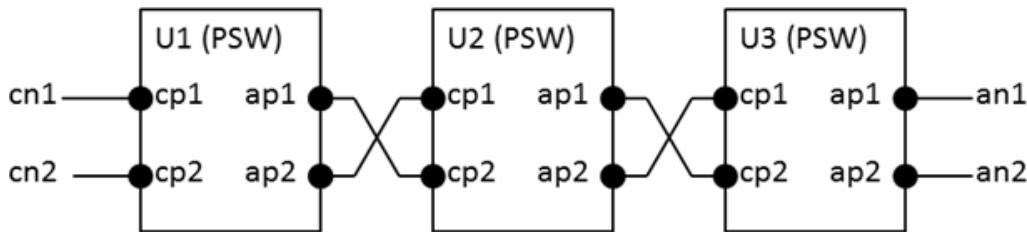


**Note**  
Currently the only simple expressions of ack ports are supported.

#### 5.1.11.2 Support for Multiple Control/Ack Power Switches in Static Verification

VC LP lets you describe a power switch which has multiple control inputs and/or acknowledge outputs in the UPF.

This section shows a design scenario along with the key aspects of the UPF and liberty. Consider [Figure 5-36](#), in which cell PSW is a power switch. The input and output power ports are not shown. There are two control inputs ( $cp1, cp2$ ), and two acknowledge outputs ( $ap1, ap2$ ).

**Figure 5-36 Sample Design Scenario**

The relevant section of the UPF is as follows:

```

create_power_switch psw1 -domain TOP \
    -output_supply_port ... -input_supply_port ... \
    -control_port {cp1 cn1} -control_port {cp2 cn2} \
    -ack_port {ap1 an1} -control_port {ap2 an2} \
    -on_state {st_hi ... {cp1 & cp2}} \
    -on_partial_state {st_lo ... {(!cp1) & !cp2}} \
    -off_state {st_off {(cp1&!cp2) | (cp2&!cp1)}}
  
```

Due to the crossover connections present in the design scenario, instance *U2* is not correctly connected; the net *cn1* should be connected to its port *cp1*, but it is not.

VC LP introduces new swapping checks that checks for swapped connections on the control or acknowledge chain in this design and generates connectivity errors if they are present. These checks are described in detail in the subsequent sections:

### 5.1.11.2.1 Swapping Checks

VC LP performs the following swapping checks:

- ❖ PSW\_CONTROL\_SWAP
- ❖ PSW\_ACK\_SWAP

These checks are disabled by default; you can enable them using the `configure_tag -app LP` command.

In general both these messages are issued, one for each pin that is swapped. The `PSW_CONTROL/ACK_CONN` messages are triggered on each individual input or output logic pin of a PSW instance in the design. VC LP also checks if there are multiple input signal pins, or multiple output signal pins. If so, when the enable tracing database is checked, the pin name is passed into that database.

If none of the control nets of the strategy pass through the pin, then the existing `PSW_CONTROL_CONN` message is issued. However, if that passes, then the pin name is checked; if the pin name does not match, a the `PSW_CONTROL_SWAP` message is issued as a swapping error has occurred. Checking of the pin name is done using simple case sensitive string comparison. Comparable check is done for acknowledge signals.

The following is sample violation report for the `PSW_CONTROL_CONN` check:

```

Tag          : PSW_CONTROL_CONN
Description   : Strategy [Strategy] control signal [UPFNets]           does not
match power switch [Instance]                                     connection pin [CellPin]
Strategy      : s_1
UPFNets
  UPFNet
    NetName   : c
    NetType   : Design/UPF
  Instance     : u4
  CellPin     : CTRL
  
```

```
DesignNet
  NetName    : n3
  NetType    : Design
```

The definition for the PSW\_CONTROL\_SWAP check is identical to PSW\_CONTROL\_CONN, except for the tag name. The PSW\_CONTROL\_CONN message however, does not enforce that the CellPin name must match the PSW control port.

```
Tag          : PSW_CONTROL_SWAP
Description   : Strategy [Strategy] control signal
not match power switch           [UPFNets] does
                                  [Instance] connection pin [CellPin]
  Strategy     : s_1
  UPFNets
    UPFNet
      NetName    : c
      NetType    : Design/UPF
  Instance     : u4
  CellPin     : CTRL
DesignNet
  NetName    : n3
  NetType    : Design
```

### 5.1.11.3 Consistency Checks Between the add\_power\_state and create\_power\_switch

You can write/mention power switch policies to implement power switches in the design. To implement power switches in the design, the logic\_expr in the add\_power\_state command and the on\_state/off\_state of the create\_power\_switch command should match. VC LP has introduced consistency checks to check if the expression in the logic\_expr and the on\_state/off\_state match. If they do not match, then VC reports violations.

The following is an example scenario of inconsistency between add\_power\_state (logic\_expr) and create\_power\_switch (on\_state/off\_state).

```
create_supply_set psw_ss1 -function {power psw_vdd}
...add_power_state psw_ss1 -state pwr_high {-supply_expr {power == `{FULL_ON, 1.0}} -
logic_expr {cntr2}}
add_power_state psw_ss1 -state pwr_off {-supply_expr {power == `{OFF}} -
logic_expr{!cntr2}}
...
create_power_switch psw -domain top_pd -output_supply_port {VDDV psw_vdd} \
                  -input_supply_port {VDD core1_vdd} \
                  -control_port {ENB cntr1} \
                  -ack_port {ACK ack1} \
                  -on_state {PSW1_ON VDD !ENB} \
                  -off_state {PSW1_OFF ENB}
```

In the above example, you can see that the logic\_expr in the add\_power\_state and the on\_state/off\_state in the create\_power\_switch are not the same.

### 5.1.11.3.1 Enabling Consistency Checks Between add\_power\_state and create\_power\_switch

By default, these consistency checks are disabled. To enable these check, use the following command:

```
set enable_aps_vs_cps_check true
```

The default value of the enable\_aps\_vs\_cps\_check application variable is false.

By default, if there are multiple `create_power_switch` commands mentioned with same group (that is, the same `supply_net`), then AND operation is performed for all `on_states` and OR operation is performed for all `off_states`. If you want to change the default behavior, then use the following command:

```
set enable_or_onstates_for_multiplePswDrivingSupplyNet true
```

The default value of the `enable_or_onstates_for_multiplePswDrivingSupplyNet` application variable is false. When you set this application variable to true, an OR operation is performed for all `on_states` and AND is performed for all `off_states`.

### 5.1.11.3.2 Low Power Checks for `add_power_state` and `create_power_switch` Commands

The following error messages are introduced to check for the consistency between the `add_power_state` (with the `logic_expr`) and the `create_power_switch` (`on_state/off_state`) expressions.

- ❖ PSW\_LOGEXP\_ONMISMATCH: This violation is reported when the `add_power_state` ON state logical expression and `create_power_switch` ON state logical expression do not match.  
The severity of this violation is *Error*.
- ❖ PSW\_LOGEXP\_OFFMISMATCH: This violation is reported when the `add_power_state` OFF state logical expression and `create_power_switch` ON state logical expression do not match.  
The severity of this violation is *Error*.
- ❖ PSW\_LOGEXP\_ONPARTMATCH: This violation is reported when the `add_power_state` ON state logical expression has more signals than the `create_power_switch` ON state logical expression.  
The severity of this violation is *Warning*.
- ❖ PSW\_LOGEXP\_ONOK: This violation is reported when the `add_power_state` ON state logical expression and the `create_power_switch` ON state logical expression match.  
The severity of this violation is *Info*.
- ❖ PSW\_LOGEXP\_OFFOK: This violation is reported when the `add_power_state` OFF state logical expression and the `create_power_switch` OFF state logical expression match.  
The severity of this violation is *Info*.
- ❖ APS\_LOGEXP\_ONOFFMISMATCH: This violation is reported when the `add_power_state` ON state and the OFF state logical expressions are not complement to each other.  
The severity of this violation is *Error*.
- ❖ APS\_LOGEXP\_ONOFFOK: This violation is reported when the `add_power_state` ON state and the OFF state logical expressions are complement to each other.  
The severity of this violation is *Info*.
- ❖ APS\_LOGEXP\_LOOP: This violation is reported if `add_power_state` (APS) `logic_expr` signal driver is driven by supply set for which `add_power_state` is defined. This violation is flagging at UPF stage and severity is error. This violation is disabled by default.

### 5.1.11.4 Support for Checks on PSW Instances Without ACK Ports

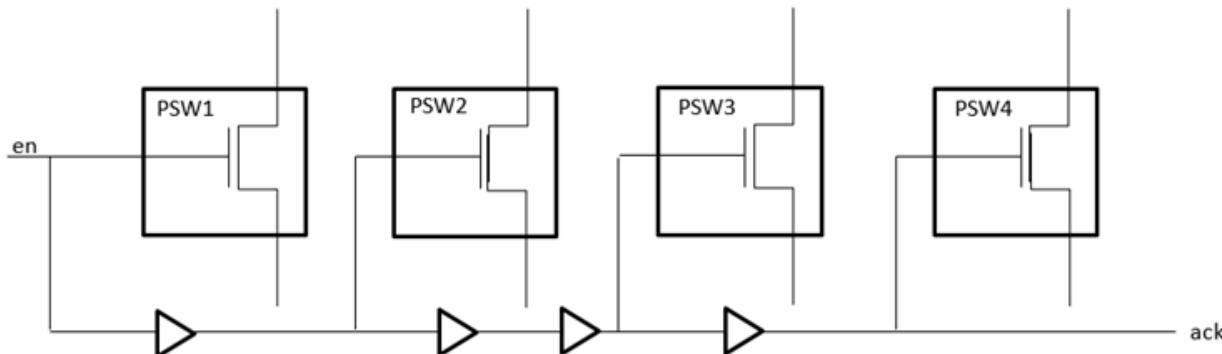
VC LP performs checks for power switch instances that do not have ACK ports. Previously, when there was no ACK port, VC LP was not able to fully match strategy with such cells resulting in few checks not happening on such PSW instances. VC LP now performs checks on PSW instances without ack ports.

To enable VC LP checks on power switches without ack ports, set the following application variable:

```
%vc_static_shell> set_app_var enable_power_switch_without_ack_pin true
```

[Figure 5-37](#) shows an example of power-switch cells that do not have an ack pin.

**Figure 5-37 PSW Instance without ACK port**



By default, the policy association algorithm in VC LP, looks for reachability of enable and ack signals to power switch instances. Only if enable and ack are reachable, the policy is considered as a candidate for full match.

VC LP uses the following policy to cell association algorithm:

- ❖ Forward traversal from enable should reach enable pins of PSW devices.
- ❖ Forward traversal from ack should not reach enable pins of PSW devices.
- ❖ Traversals from InputSupply and OutputSupply to hit the PG pins of PSW.
- ❖ Backward traversal from ack.
  - ◆ If the ack reaches an enable, and both appear in a PSW strategy, then strategy is fully associated to cell as long as above all points hold true.
  - ◆ If the ack does not reach an enable, then it is not a full match.

The following violation tags are introduced for this support:

- ❖ PSW\_CTRLACK\_MATCH
 

This violation is reported when the enable signal is connected to ack signal of same policy.
- ❖ PSW\_CTRLACK\_MISMATCH
 

This violation is reported when the enable signal is connected to ack signal of a different policy.
- ❖ PSW\_CTRLACK\_INVERT
 

This violation is reported when the inversion in the EN net to Ack net Path
- ❖ PSW\_CONTROL\_VOLTAGE
 

This violation is reported if the voltage level of the supply set used in -supply\_set is less than that of input supply voltage according to the PST.

### 5.1.11.5 Support for Reporting Unused PSW Ports

The `display_unused_psw_policy_ports` application variable supports reporting of the unused PSW ports with respect to the strategy port.

When this application variable is set to true, the `UnusedPolicyPorts: PortName` debug fields are reported in the `PSW_CONTROL_CONN/PSW_ACK_CONN` violations.

For example, if a PSW strategy has two control port defined, and one of them is not properly reaching the PSW cell control port when a multi-bit PSW cell is used, VC LP reports the PSW\_CONTROL\_CONN violation with the unused strategy port to all remaining control bit of the PSW cell. The same fundamentals are applicable for the `ack_port`.

Here, `MY_CELL_1` is a PSW cell with a multi-bit (8-bit) ACK/CONTROL port. The PSW strategy contains only three `ack_port` and three `control_port` are defined, among them one is not reaching to PSW cell's `ack/control` port. Therefore, with respect to that unused port, VC LP reports `PSW_CONTROL_CONN/PSW_ACK_CONN` for all remaining 6 bit of the `ack_port` and `control_port` which are not defined in design PSW cell instantiation.

In the following example, for highlighted `ack` and `control` port, VC LP reports `PSW_ACK_CONN` and `PSW_CONTROL_CONN` respectively for all the remaining `ack` and `control` bit which are not mentioned in the cell instantiation.

### PSW Strategy Snippet

```
create_power_switch PWR_SW1 \
-domain PD1 \
-input_supply_port {VDD VDD15} \
-output_supply_port {VDDV VDD1} \
-control_port {ENB5 A/en_1[5]} \
-control_port {ENB6 A/en_1[6]} \
-control_port {ENB7 A/control_new} \
-ack_port {ACK3 A/en_2[3] ENB5} \
-ack_port {ACK4 A/en_2[4] ENB6} \
-ack_port {ACK5 A/ack_new} \
-on_state {SW_ON VDD { ENB5 || ENB6 }} \
-off_state {SW_OFF { !ENB5 || !ENB6 }}
```

### Design Snippet

```
MY_CELL_1 sw_1(.ENB[0] (en_1[5]), .ENB[1] (en_1[6]), .ACK[0] (en_2[3]), .ACK[1]
(en_2[4]), .VDD(VDD15), .VDDV(VDD1));
```

### Violation Snippet

For all the remaining `CellPin`, VC LP reports the `PSW_CONTROL_CONN` violation with unused strategy port as given below.

```
-----
PSW_CONTROL_CONN (6 errors/0 waived)
-----
Tag : PSW_CONTROL_CONN
Description : Strategy [Strategy] control signal [UPFNets] does not match power
switch [Instance] connection pin [CellPin]
Violation : LP:61
Strategy : PWR_SW1
UPFNets
UPFNet
NetName : A/control_new
NetType : Design/UPF
UPFNet
NetName : A/en_1[5]
NetType : Design/UPF
UPFNet
NetName : A/en_1[6]
NetType : Design/UPF
```

```

Instance          : A/sw_1
CellPin          : ENB[5]
Cell              : MY_CELL_1
DesignNets
DesignNet
NetName          : A/N_8[3]
NetType           : Design
Unconnected      : True
UnusedPolicyPorts
PortName         : ENB7

```

Similarly, the PSW\_ACK\_CONN violation is reported for the ACK port.

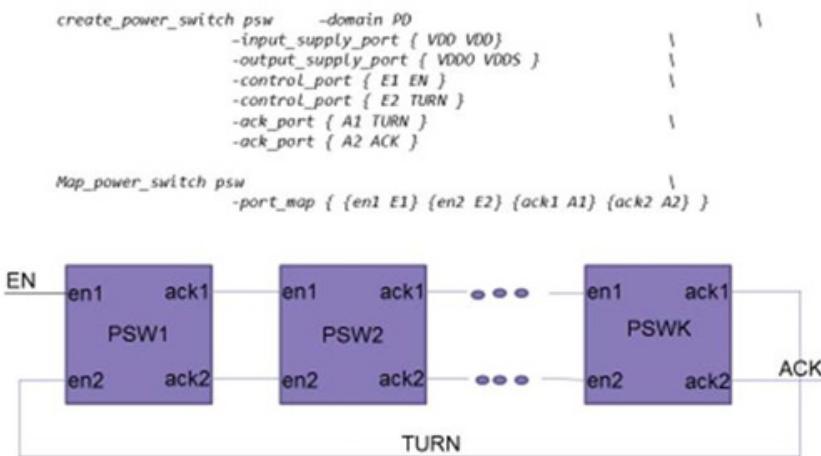
```

-----
PSW_ACK_CONN   (6 errors/0 waived)
-----
Tag            : PSW_ACK_CONN
Description     : Strategy [Strategy] acknowledge signal [UPFNets] does not match
power switch [Instance] connection pin [CellPin]
Violation       : LP:55
Strategy        : PWR_SW1
UPFNets
UPFNet
  NetName       : A/ack_new
  NetType        : Design/UPF
UPFNet
  NetName       : A/en_2[3]
  NetType        : Design/UPF
UPFNet
  NetName       : A/en_2[4]
  NetType        : Design/UPF
Instance        : A/sw_1
CellPin        : ACK[7]
DesignNets
DesignNet
  NetName       : A/N_0[7]
  NetType        : Design
UnusedPolicyPorts
PortName       : ACK5

```

### 5.1.11.6 Support for Mother-Daughter PSW Strategy

VC LP supports the mother-daughter topology of PSW strategies. A typical mother-daughter chain of PSW devices is shown in [Figure 5-38](#).

**Figure 5-38 Mother Daughter PSW Strategy**

To configure this kind of topology, the `set_psw_turn_net` Tcl command is introduced. When this command is specified, VC LP considers the PSW switch as mother-daughter PSW strategy.

### Syntax

```
%vc_static_shell> set_psw_turn_net -help
Usage: set_psw_turn_net      # Sets PSW turn net in case of Mother-Daughter PSW
Configuration
    -strategy <strategy>      (Target power switch strategy)
    -net <net>                (Target power switch turn net)
```

### Example

```
%vc_static_shell> set_psw_turn_net -strategy psw -net TURN
```

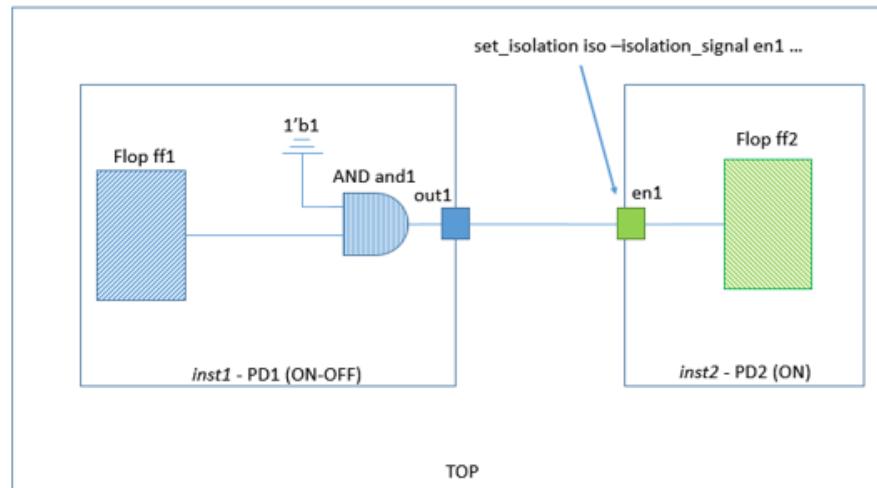
## 5.1.12 Constant Aware Checks

VC LP currently does not consider the impact of design constant values reaching pins of combinational logic by default when performing checks. For example, consider an AND cell having one input as constant value 1, this cell is functionally equivalent to a buffer.

Therefore, VC LP has added support to make some checks as constant aware. When the constant aware checks are enabled, the constant values are propagated in the design before performing checks, subsequently reducing the number of violations reported for some of the checks. The constant aware support may increase the run time of VC LP checks.

### Example

For the example shown in [Figure 5-39](#), a constant input is provided to the AND gate, and VC LP must treat the *and* gate as a buffer, and must not report the ISO\_STRACONTROL\_GLITCH violation for iso isolation strategy.

**Figure 5-39 Constant Aware Checks**

### 5.1.12.1 Enabling Constant Aware Checks

To enable constant aware checks, set the following application variable to true:

```
%vc_static_shell>set_app_var lp_enable_constant_propagation true
```

By default, the *lp\_enable\_constant\_propagation* application variable is set to false.

Use the *configure\_constprop* command with *-mode* option to configure how constant propagation should be performed when the design is accessing constant aware commands such as *get\_attribute<>case\_values* before the *check\_lp*.

#### Syntax

```
vc_static_shell> configure_constprop -help
Usage: configure_constprop      # Configure the constant propagation mode
       -mode                      (const-prop mode)
```

The following options can be used for *-mode*:

- ❖ *user*: Propagate only user/*set\_case\_analysis* constants
- ❖ *design*: Propagate only design constant
- ❖ *all*: Propagate both SCA and design constants
- ❖ *none*: None will be propagated

This command overrides any other configuration/application variables.

#### Examples

- ❖ To propagate user-only or *set\_case\_analysis* constants, use the following command:  
`configure_constprop -mode user  
set_app_var disable_design_constant_analysis false`
- ❖ If you do not want to propagate anything, following configuration should be used.  
`configure_constprop -mode none`
- ❖ If the *configure\_constprop* command is not specified, previous behavior will remain intact.

This will be valid for all constant aware commands such as `all_fanout`, `all_fanin`, `get_trace_paths`, `get_attribute<>case_value`.



### Note

The `lp_enable_constant_propagation` application variable internally sets a few other application variables: `const_aware_traversal` to true (default false), `disable_verilog_supply_constant_analysis` false (default true), and `disable_design_constant_analysis` to false (default true). It is recommended that you do not set these application variable to conflicting values, as it might affect the desired behavior.

When the `lp_enable_constant_propagation` application variable is set to true, while performing some of the checks, VC LP traverses through combination logic cells, multiplexers, and so on, which has a resolved output logic. The impact of the traversal is reflected in the following checks:

- ❖ ISO\_CONTROL\_GLITCH
- ❖ ISO\_CONTROL\_RESET
- ❖ ISO\_DATA\_BLOCKED
- ❖ ISO\_DATA\_CONSTANT
- ❖ ISO\_INST\_CLAMPED
- ❖ ISO\_INST\_TRANSPARENT
- ❖ LS\_INPUT\_TIEHI
- ❖ LS\_INPUT\_TIELO
- ❖ PSW\_CONTROL\_TIEHI
- ❖ PSW\_CONTROL\_TIELO
- ❖ PSW\_INST\_BUF
- ❖ RET\_INPUT\_TIEHI
- ❖ RET\_INPUT\_TIELO
- ❖ ISO\_STRATEGY\_CTRL\_CONST
- ❖ PSW\_STRATEGY\_CTRL\_CONST
- ❖ RET\_CONTROL\_CONST



### Note

VC LP has introduced the `lp_iso_reset_constant` application variable. This application enables checking of constant value propagation through sequential element. By default, this application variable is set to false, and the ISO\_CONTROL\_RESET and ISO\_INST\_TRANSPARENT violations are moved under the `lp_iso_reset_constant` application variable. By default (`lp_iso_reset_constant` set to false), the ISO\_INST\_TRANSPARENT violation will not be reported if the constant value is an output of a flop. Similarly, the ISO\_CONTROL\_RESET is not reported by default.

### Example

For the schematic as shown in [Figure 5-40](#), the ISO control connection is recognized as below.

*Constant 0 -> mux output (mux select is tied to 0) -> AND gate input -> And gate output (Other AND gate input is tied to 1) -> inverter input -> inverter output -> ISO Enable signal*

Hence, ISO\_INST\_TRANSPARENT is reported.

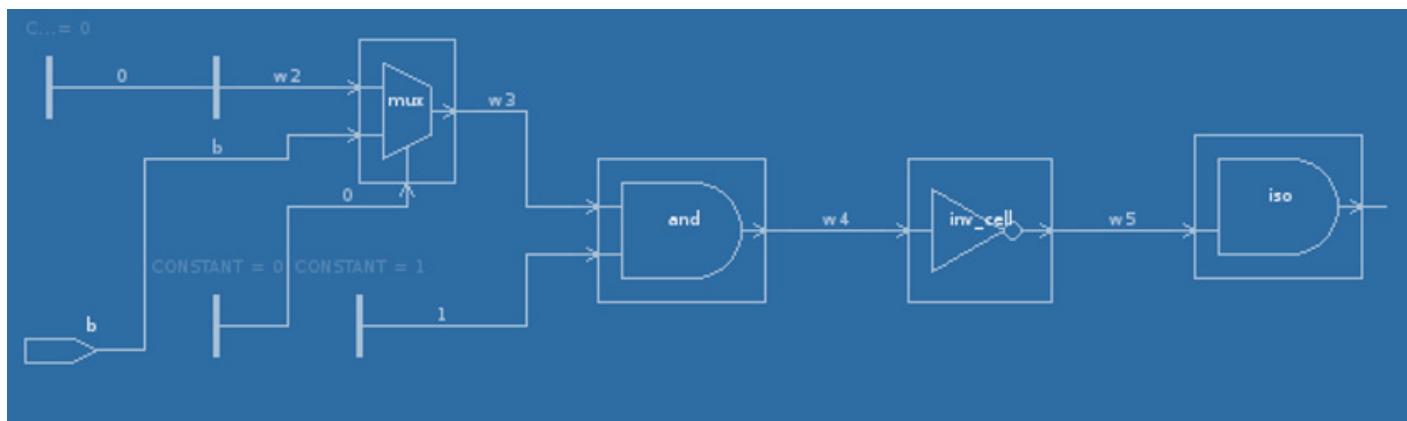
---

ISO\_INST\_TRANSPARENT (1 error/0 waived)

```

-----
Tag : ISO_INST_TRANSPARENT
Description : Isolation instance [Instance] ([Cell]) is transparent as the
enable pin [CellPin] [ReasonCode]
Violation : LP:16
Instance : iso
Cell : ISOANDLO
CellPin : ISO
ReasonCode : TIED_TO_CONSTANT
TieType : TIE_HIGH (because of the inverter)

```

**Figure 5-40 Constant Aware Checks**

### 5.1.13 Predictive Checks

VC LP supports predictive checks. The predictive checks enable you to perform location based checks at the RTL level, when actual LS and ISO cells are not inserted in the design. VC LP can predict the location where the LS and ISO cells will be inserted in the design, and perform various checks based on the predicted location.

#### 5.1.13.1 Enabling Predictive Checks

The predictive checks are disabled by default. To enable predictive checks, set the following application variable to true:

```
%vc_static_shell> set_app_var lp_enable_virtual_protection true
```

VC LP predicts the ISO cell that will be inserted in the design based on the strategy node and its location mentioned in the strategy. The virtual isolation crossover nodes are updated in the crossover database.

VC LP predicts where the LS cell will be inserted after the virtual ISO nodes are inserted in the crossover, because based on the virtual ISO nodes, the various sub segments are made for the level shifter requirement. VC LP performs predictive checks for LS cells by considering the level shifter input supply as source supply, and the output supply as receiver supply.

When the `lp_enable_virtual_protection` application variable is set to true, if the actual node of the virtual ISO node is also a LS implementable policy node, which should be considered as an ELS, the crossover segment source/sink for LS UPF checks does not stop at this virtual ISO node. The crossover passes through the virtual ISO node.

## Note

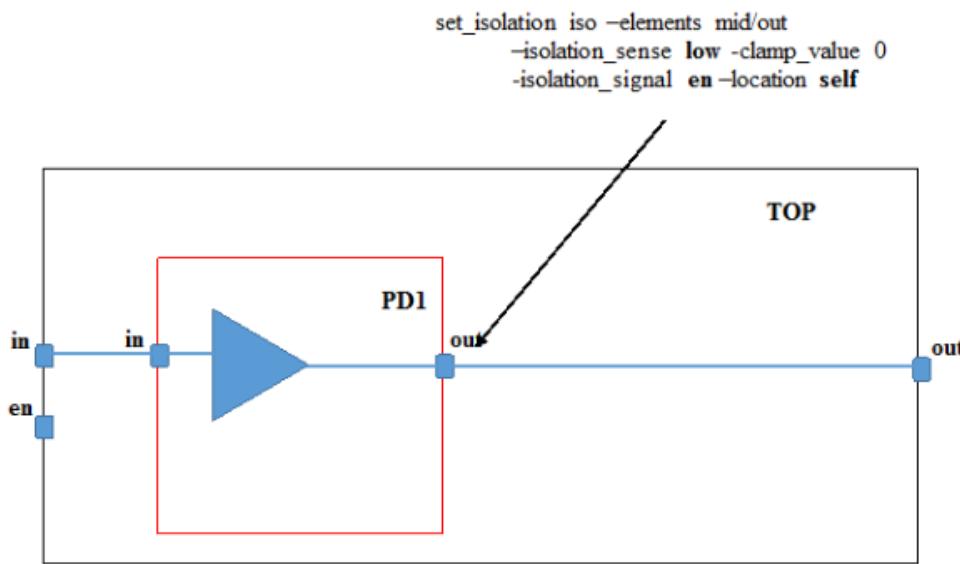
The VC-LP-ULTRA licenses required to enable the lp\_enable\_virtual\_protection application variable.

### Example

The following crossover is generated for the example design shown in [Figure 5-41](#).

```
mid/U1/Z - -> mid/out {Normal Virtual_ISO_Node} --> mid/out {POCILCY_ASSOC_NODE BNDRY NODE} --> out
```

**Figure 5-41 Predictive Checks Example**



#### 5.1.13.2 Violations Introduced for Predictive Checks

The LS\_SCMR\_LOCATION violation is introduced for predictive checks. This violation is reported at the UPF stage, with severity *error* and is enabled by default.

When the predictive checks are enabled, the location of LS cell is predicted at the UPF stage, and the LS\_SCMR\_LOCATION violation checks for the SCMR written on the input pin or the output pin. The LS\_SCMR\_LOCATION violation is reported when the SCMR (std\_cell\_main\_rail) power pin and the power domain primary power does not match with the level shifter cells specified using the map\_level\_shifter\_cell command. The immediate source/sink is considered for this check.

## Note

By default, VCLP reports missing CSN on LS SCMR pins. To disable this support, set the lp\_ls\_check\_csn\_scmr application variable to false.

The following debug fields in the LS\_SCMR\_LOCATION violation are used to indicate if the source/sink are virtual ISO nodes.

- ❖ *IsVirtual: True* - If the logic source is a virtual ISO node.
- ❖ *SegmentSinkVirtual: True* - If the logic sink is a virtual ISO node.

The supply inference method for the following violations honor virtual instrumentation under the lp\_enable\_virtual\_instrumentation application variable:

- ❖ LS\_SUPPLY\_UNAVAIL
- ❖ LS\_SUPPLY\_MISMATCH
- ❖ LS\_STRATEGY\_INCORRECT
- ❖ LS\_ISO\_MISMATCH

When the `lp_instrument_backtoback_ls` application variable is set to true, VC LP instruments back to back LS devices for feedthrough paths. By default, the `lp_instrument_backtoback_ls` application variable is set to false.

#### 5.1.13.3 Changes in Existing Tags When Predictive Checks are Enabled

- ❖ The CORR\_CONTROL\_STATE\_WITHISO/CORR\_CONTROL\_STATE\_NOISO Violation

Under the `lp_enable_virtual_protection` application variable, the CORR\_CONTROL\_STATE\_WITHISO and CORR\_CONTROL\_STATE\_NOISO violation considers ISO strategy and the "*IsolsVirtual*" debug field is added for it.

- ❖ The UPF\_SPADRIVER\_STATE/UPF\_SPARECEIVER\_STATE Tags

By default, the UPF\_SPADRIVER\_STATE/UPF\_SPARECEIVER\_STATE checks the PST states of the SPA driver/receiver, and the PST states of actual driver/receivers.

When the predictive checks are enabled, VC LP considers the virtual ISO nodes and virtual LS nodes as source and sink for checking SPA driver\_supply/receiver\_supply instead of the actual source and sink.

VC LP performs consistency checks with repeater\_supply/set\_repeater under predictive checks. That is, when the `lp_enable_virtual_protection` application variable is set to true, VC LP performs consistency checks for the following:

- ◆ SPA driver and SPA repeater\_supply at input
- ◆ SPA receiver and set\_repeater at input
- ◆ SPA driver and SPA repeater\_supply at output
- ◆ SPA driver and set\_repeater at output

The following violations are reported for these checks:

- ◆ UPF\_SPASUPPLY\_CONN
- ◆ UPF\_SPASUPPLY\_VOLTAGE
- ◆ UPF\_SPADRIVER\_STATE
- ◆ UPF\_SPARECEIVER\_STATE

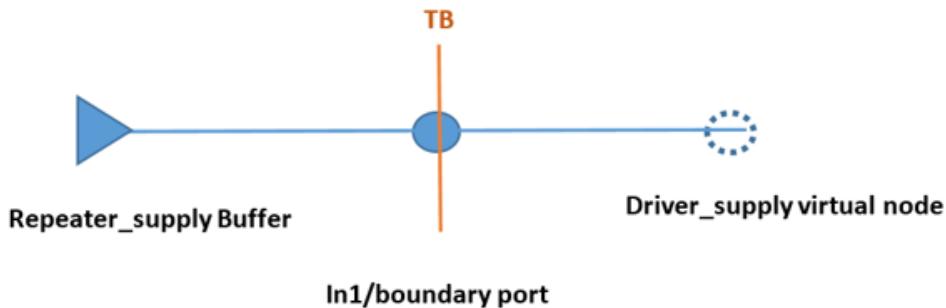
#### Examples

The consistency checks are executed as follows:

#### UPF snippet

```
set_design_attributes -elements { . } -attribute terminal_boundary true
set_port_attributes -ports {in1 out1} -driver_supply SS_SPA -receiver_supply SS_SPA -
repeater_supply SS_RS
```

For input port when both SPA -driver\_supply and repeater\_supply are specified, the repeater buffer is inserted before specified port.



When SPA repeater\_supply (SS\_RS) is specified in the set\_port\_attributes command, VC LP resolves the actual driver supply from the repeater\_supply, and if the repeater\_supply is less on than the SPA driver\_supply (SS\_SPA), VC LP reports the UPF\_SPADRIVER\_STATE violation.

```
-----  
UPF_SPADRIVER_STATE (1 warning/0 waived)  
-----  
Tag      : UPF_SPADRIVER_STATE  
Description : SPA driver supply on, but supply of actual driver is off for SPA constraint on [ConstraintNode]  
ConstraintNode : core1_inst/in1  
DriverNode  : core1_inst/in1  
IsVirtual   : True  
ConstraintInfo  
PowerNet  
NetName    : VDD1  
NetType    : UPF  
PowerMethod : FROM_UPF_DRIVER_SUPPLY  
DriverInfo  
PowerNet  
NetName    : core1_inst/VDD2  
NetType    : UPF  
PowerMethod : FROM_UPF_REPEATERSUPPLY  
TerminalBdry : true  
Internal    : False  
States  
State     : core1_inst/pst_in_BB/f0_c3  
UPFCommand
```

#### ❖ The LS\_SUPPLY\_UNAVAIL/ISO\_SUPPLY\_UNAVAIL Tags

- ◆ The LS\_SUPPLY\_UNAVAIL violation checks if the LS strategy has input/output supply. When the predictive checks are enabled, VC LP checks for the predicted LS cell location accurately using the data in the predictive database.
- ◆ The ISO\_SUPPLY\_UNAVAIL violation checks the isolation\_power\_net and isolation\_ground\_net availability in the ISO cell inferred domain. When the predictive checks are enabled, VC LP checks for the predicted ISO cell location accurately using the data in the predictive database.
- ◆ For back to back level shifter scenarios under the lp\_enable\_virtual\_protection/enable\_predictive\_checks flows, the LS\_SUPPLY\_UNAVAIL and LS\_SUPPLY\_MISMATCH considers the supplies of the immediate level shifter strategy instead of the supplies of the actual source/sink.

The debug fields are updated as follows when the LS\_SUPPLY\_MISMATCH/LS\_SUPPLY\_UNAVAIL/LS\_STRATEGY\_INCORRECT violations are caused by back-to-back level shifter strategies:

- ❖ The segment source debug field shows the power domain boundary pin associated with the front level shifter strategy and *IsVirtual* : *True* if it comes from a front level shifter strategy.
- ❖ The segment sink debug field shows the power domain boundary pin associated with the back level shifter strategy and *SegmentSinkVirtual* debug field shows the *SegmentSinkVirtual* : *True*: if it comes from a back level shifter strategy.
- ❖ The segment source/sink supply debug field shows supplies from the front/back level shifter strategy and output/input supply that causes the violation and *PowerMethod* shows FROM\_UPF\_POLICY.
- ❖ The *LsStrategies* debug field shows front/back level shifter strategies that causes the violation.

### Example

For the crossover segment source->LS1->LS2->sink:

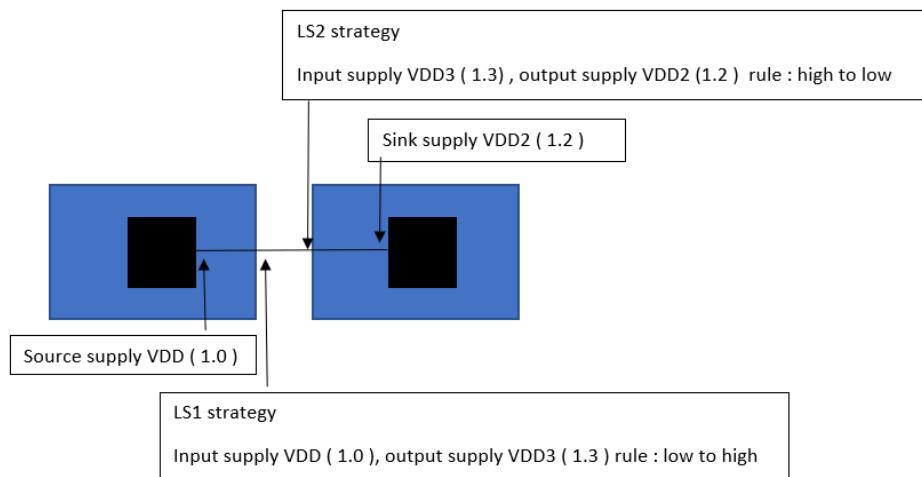
For LS\_SUPPLY\_UNAVAIL for LS2, VC LP checks the supply availability from LS1 instead of source, if LS1 has output supply set, that is considered for the check. If not, the domain supply will be considered. Similarly for LS1, VC LP checks the supply availability from LS2 instead of sink, if LS2 has output supply set, that is considered. If not, the domain supply is considered.

LS\_SUPPLY\_MISMATCH for LS2 input supply set, VC LP checks the LS1 output supply set instead of source supply. If LS1 has no output supply set, there will not be supply mismatch since this is back to back LS case. Similarly, for LS1 output supply set, VC LP checks for LS2 input supply set instead of sink supply.

- ◆ When the `lp_enable_virtual_protection` application variable is set to true, and there are no associated LS cells instantiated, these violations should consider the input and output supplies specified in the back to back level shifter strategies when identifying the supplies to compare.

### Example

Consider the [Figure 5-42](#) as an example

**Figure 5-42 LS\_SUPPLY\_MISMATCH Violations**

In this example, when comparing supplies for LS1 output for LS\_SUPPLY\_MISMATCH, VC LP considers the input supply of LS2 instead of the actual sink VDD2. Similarly, for LS2 input, VC LP considers output supply of LS1 instead of actual source VDD

When the `lp_enable_virtual_protection` application variable is set to false and there are no associated cells, VC LP considers the actual source and sink for comparison for both LS strategies.

When there are associated LS cells instantiated for each strategy, these violations are not reported regardless of the value of the `lp_enable_virtual_protection` application variable.

- ◆ Under the `lp_enable_virtual_protection/enable_predictive_checks` application variable, the LS\_SUPPLY\_UNAVAIL/LS\_SUPPLY\_MISMATCH considers the PSW supply connection.

The LS\_SUPPLY\_MISMATCH will not be flagged if the supplies are connected through PSW strategy.

- ❖ For LS\_SUPPLY\_UNAVAIL if the sink supply is PSW output supply and is not available in the LS power domain, but PSW input supply is available, then VC LP considers it as a valid case and does not report the LS\_SUPPLY\_UNAVAIL violation.
- ❖ The same for LS\_SUPPLY\_MISMATCH, if LS output supply does not match with sink supply which is the PSW output supply, but it matches with PSW input supply, VC LP considers it as a valid case and does not report the LS\_SUPPLY\_MISMATCH violation.

#### ❖ The LS\_SUPPLY\_STATE Tag

Under the `lp_enable_virtual_protection/enable_predictive_checks` application variable, the LS\_SUPPLY\_STATE checks are based on the crossover path. The source/sink information is added in debug field. When the node has both isolation and level shifter strategies defined, the LS\_SUPPLY\_STATE is not reported, since for such scenarios an ELS is expected to be instrumented, as for ELS, the input supply can be OFF while output supply is ON.

#### ❖ The LS\_STRATEGY\_INCORRECT Tag

- ◆ Under the `lp_enable_virtual_protection` application variable for back-to-back level shift scenario, the LS\_STRATEGY\_INCORRECT considers the supplies of the immediate level shifter strategy instead of the supplies of the actual source/sink.

Consider the crossover segment source:

LS1->LS2->sink

For the correct expected LS strategy type with LS1, VC LP checks the LS2 input supply instead of the sink supply, if the LS2 input supply is not present, check the LS2 domain supply. Similarly, for LS2, VC LP checks the LS1 output supply instead of the source supply, if the LS1 output supply is not present, check LS1 domain supply.

- ◆ Under the `lp_enable_virtual_protection` application variable, the `LS_STRATEGY_INCORRECT` checks consider back-to-back level shifters, when the second level shifter does not have `input_supply_set` defined, VC LP infers the output supply for first level shifter from the `output_supply_set` of the second level shifter for LS rule type matching.
- ◆ When the `lp_enable_virtual_protection` application variable is set to true, these violations should consider the input and output supplies specified in the back-to-back level shifter strategies when identifying the supplies to compare.

Example

Consider the example in [Figure 5-42](#)

For LS1 rule type comparison, the voltage shift to be compared should be the one from source (VDD) to LS2 input VDD2. The pst has low to high requirement which matches with LS1 rule type. Therefore, the `LS_STRATEGY_INCORRECT` violation is not reported.

For LS2, rule type comparison, the voltage shift to be compared should be the one from LS2 output (VDD3) to sink VDD2. The pst has high to low requirement which matches with LS2 rule type. Therefore, the `LS_STRATEGY_INCORRECT` violation is not reported.

When the `lp_enable_virtual_protection` application variable is set to false, these violations should consider the actual source and sink for comparison. That is for both LS1 and LS2 the shift to be compared is from source VDD to sink VDD2 which has the pst requirement low to high. Therefore, LS2 would flag this violation.

#### ❖ The `LS_STRATEGY_MISSING` Tag

The `LS_INST_MISSING` violation is reported when the isolation strategy for the crossover with an isolation supply has voltage different from the source/sink supply.

When the predictive checks are enabled, VC LP considers the virtual ISO nodes for source/sink at the RTL level, and then performs the `LS_STRATEGY_MISSING` checks.

#### ❖ The `LS_STRATEGY_MULTIPLE` Tag

Regardless of the application variable `lp_enable_virtual_protection` being set to true or not, if there is just one type of voltage shift in the path (that is either high to low or low to high) and there is more than one LS strategy specified along the path, `LS_STRATEGY_MULTIPLE` should be flagged.

If there is both high to low and low to high shifts along the path and the level shifter strategies with correct rule types are placed, then `LS_STRATEGY_MULTIPLE` is not reported.

If there is both high to low and low to high shifts along the path and level shifters with incorrect rule mapping are placed at node, then if there is both high to low and low to high shifts along the path should be flagged.

#### ❖ The `UPF_HIERSRSN_CONN` Tag

During block level verification if the external boundary condition is modeled as a SRSN at block level input/output port, and such blocks are used and integrated in top level, then the SRSN at the block level becomes a hierarchical node. To validate the boundary condition that the block level

assumption is true with respect to top level actual driver/load connection, VC LP has added the UPF\_HIERSRSN\_CONN consistency check.

When the predictive checks are enabled, VC LP considers the virtual ISO nodes and virtual LS nodes for immediate driver and load of the intermediate SRSN pins.

- ❖ **The ISO\_CONTROL\_STATE and ISO\_CONTROL\_VOLTDIFF Tag**

When the predictive checks are enabled, the checks related to isolation control path at the RTL stage such as ISO\_CONTROL\_STATE and ISO\_CONTROL\_VOLTDIFF considers the root supply from isolation strategy written in the enable path. When the predictive checks are enabled, the shortening of ISO enable signal can be avoided.

- ❖ **The LS\_ISO\_MISMATCH tag**

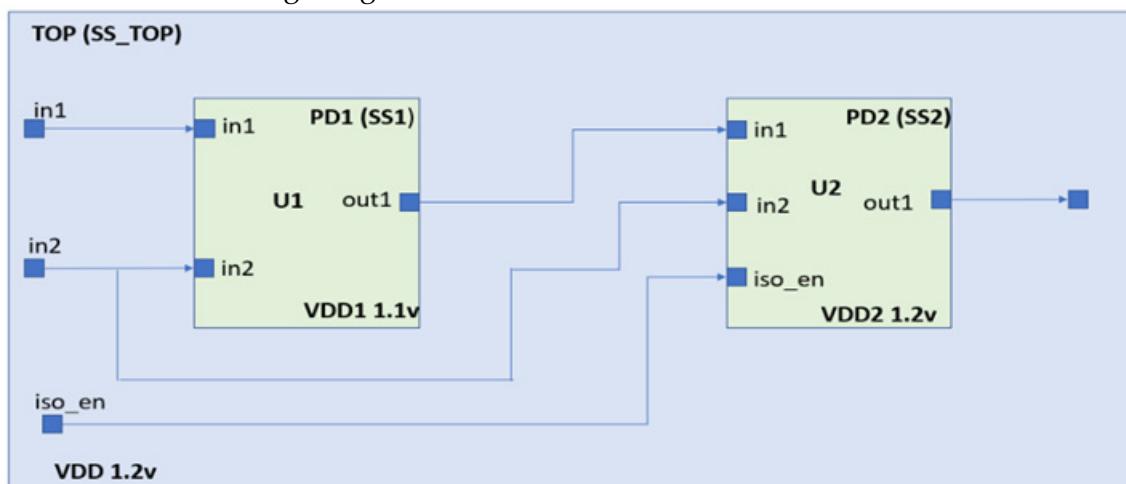
The LS\_ISO\_MISMATCH tag reports the incompatibilities in Isolation and level shifter strategies for ELS nodes in LP virtual instrumentation.

If a node has both ISO and LS strategies, an ELS cell could be instrumented at that node only if the location options in both strategies are matching, and `isolation_supply_set` is matching with `-output_supply_set` option of LS strategy.

If either of these options are not matching, then ELS cells cannot be instrumented on particular node, and it will flag the LS\_ISO\_MISMATCH violation for the respective node.

### Example

Consider the following design



**upf#**

```
set_isolation iso2 -domain PD2 -source SS1 -diff_supply_only true -clamp_value 0 -
isolation_supply_set SS2 -isolation_signal iso_en -isolation_sense low -location parent
set_level_shifter ls2 -domain PD2 -source SS1 -sink SS2 -applies_to inputs -rule
low_to_high -location self -input_supply SS1 -output_supply SS2
```

In this example, the LS\_ISO\_MISMATCH is reported because of the location mismatch between ISO and LS strategy. ELS cannot be instrumented at the input nodes (**in1, in2**) of U2 element.

The following is an example snippet of the LS\_ISO\_MISMATCH violation:

```

LS_ISO_MISMATCH (1 error/0 waived)

Tag : LS_ISO_MISMATCH
Description : ELS cannot be implemented because Level shifter [LsPolicy] and isolation [IsolationPolicy] strategies
Violation : LP:6
LsPolicy
  Strategy : PD2/ls2
  UPFLocation : self
  OutputSupplySet : SS2
IsolationPolicy
  Strategy : PD2/iso2
  UPFLocation : parent
  SupplySet : SS2
ReasonCode : Location
StrategyNode : uSUB2/in1
LogicSource
  PinName : uSUB1/C7/Z
LogicSink : uSUB2/C7/A

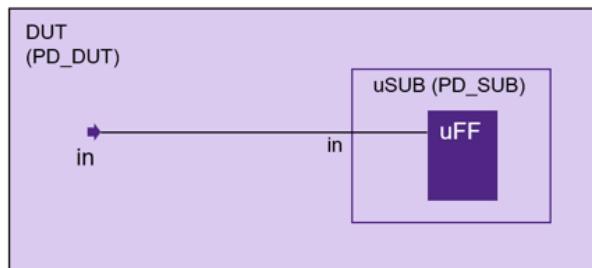
```

### Notes:

When doing crossover analysis, VC LP creates virtual crossover nodes for `set_port_attributes -driver_supply/-receiver_supply` and `set_related_supply_net`. When the `lp_xovercmd_virtual_node` application variable is set to true, the `get_crossover_nodes` query command tries to report the virtual nodes only if both virtual nodes and actual nodes exist.

### Example

Consider the following crossover example:



```

set_isolation ISO1 -domain PD_DUT -applies_to inputs
set_isolation ISO2 -domain PD_SUB -applies_to inputs
set_port_attributes -ports in -driver_supply PD_DUT.primary

```

- "in" & "uSUB/in" are Strategy association node
- "in" have `set_port_attribute -driver_supply`
  - VC LP creates "in" Virtual Node

When the `set_app_var lp_xovercmd_virtual_node true` is set, the following is the output:

```

get_crossover_nodes [ get_crossovers -from_signal in ] -filter { is_src_node == false &&
is_sink_node == false && is_domain_boundary == true}
  {"uSUB/in"}

```

When the `set_app_var lp_xovercmd_virtual_node false` is set, the following is the output:

```

get_crossover_nodes [ get_crossovers -from_signal in ] -filter { is_src_node == false &&
is_sink_node == false && is_domain_boundary == true}
  {"in", "uSUB/in"}

```

#### 5.1.13.3.1 Limitation

VC LP does not consider the location fanout and automatic attributes while predicting LS cells.

#### 5.1.13.4 Changes in Structural Checks for Level Shifters at RTL stage

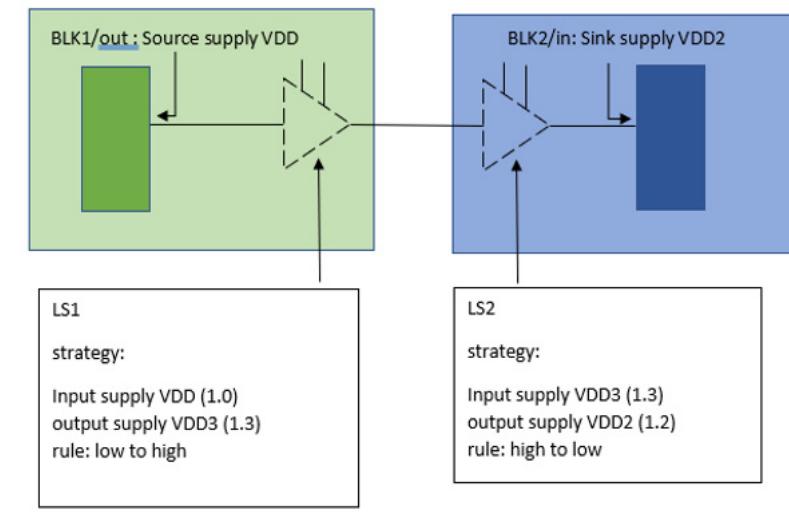
There is noise reduction in the following structural checks related to level shifters at RTL stage when `lp_enable_virtual_protection` is set to true.

- ❖ LS\_SUPPLY\_MISMATCH

Example

Consider the following design illustrated in [Figure 5-43](#).

**Figure 5-43 Level Shifter Structural Checks at RTL Stage**



Consider the example in [Figure 5-43](#), at RTL level without instrumentation, the LS Strategies compare the input/output supplies with actual source/sink supplies.

- ◆ LS1: input supply with BLK1/out (No violation is reported)
- ◆ LS1: output supply with BLK2/in (LS\_SUPPLY\_MISMATCH violation, noise)
- ◆ LS2: input supply with BLK1/out (LS\_SUPPLY\_MISMATCH violation, noise)
- ◆ LS2: output supply with BLK2/in (No violation is reported)

Consider the example in [Figure 5-43](#), at RTL level with instrumentation, the LS Strategies consider the input/output supplies of the adjacent LS strategies

- ◆ LS1: input supply with BLK1/out (No violation is reported)
- ◆ LS1: output supply with LS2/in (No violation is reported)
- ◆ LS2: input supply with LS1/out (No violation is reported)
- ◆ LS2: output supply with BLK2/in (No violation is reported)

- ❖ LS\_STRATEGY\_INCORRECT

Consider the example in [Figure 5-43](#), at RTL level without instrumentation, LS Strategies consider the actual source/sink supplies:

- ◆ LS1: BLK1/out and BLK2/in (No violation is reported)
- ◆ LS2: BLK1/out and BLK2/in (LS\_STRATEGY\_INCORRECT violation, noise)

Consider the example in [Figure 5-43](#), at RTL level with instrumentation, the LS Strategies consider the input/output supplies of the adjacent LS strategies:

- ◆ LS1: BLK1/out and LS2/input supply (No violation is reported)
- ◆ LS2: LS1/output supply and BLK2/in (No violation is reported)
- ❖ LS\_SUPPLY\_UNAVAIL

Consider the example in [Figure 5-43](#), at RTL level without instrumentation, when input/output supplies are not defined in a LS strategy, the supply will be taken from the driver or load of the level shifter cell; which is BLK1/out and BLK2/in

- ◆ LS2: input supply: BLK1/out (VDD) (LS\_SUPPLY\_UNAVAIL, noise)
- ◆ LS2: output supply: BLK2/in (VDD2) (No violation is reported)

Consider the example in [Figure 5-43](#), at RTL level with instrumentation, when input/output supplies are not defined in a LS strategy, the supply will be taken from the driver or load of the level shifter cell; which is BLK1/out and BLK2/in:

- ◆ LS2: input supply: LS2/output supply (VDD3) (No violation is reported)
- ◆ LS2: output supply: BLK2/in (VDD2) (No violation is reported)

- ❖ LS\_STRATEGY\_REDUND

When the `lp_enable_virtual_protection` application variable is set to true, if the actual node of the virtual ISO node is also a LS implementable policy node, which should be considered as an ELS, the crossover segment source/sink for LS UPF checks does not stop at this virtual ISO node. The crossover passes through the virtual ISO node.

### **5.1.14 Support for Predictive Checks Related to map\_isolation\_cell/map\_level\_shifter\_cell Commands**

VC LP has introduced two predictive check related tags ISO\_MAP\_TYPE and LS\_MAP\_TYPE for the map\_isolation\_cell/map\_level\_shifter\_cell commands. This feature is available for user\_interface\_cell command as well.

The ISO\_MAP\_TYPE / LS\_MAP\_TYPE checks are performed during the check\_lp -stage upf stage.

The checks are performed on strategy associated node in crossover path. The strategy association need to be complete before doing the check. If a strategy is dropped because of `src_supply/sink_supply/diff_supply_only` mismatch, the check will not be performed on that strategy.

1. If isolation strategy is associated on the node and no pure isolation cell is present in the map\_isolation\_cell command, VC LP reports ISO\_MAP\_TYPE with reason\_code:PURE\_NONE
2. If level shifter strategy is associated on the node and LS is required by PST and no pure level shifter is present in map\_level\_shifter\_cell command, VC LP reports LS\_MAP\_TYPE with reason\_code:PURE\_NONE
3. If in a crossover path, the node has both isolation and level shifter associated, and level shifter is required
  - a. ISOLATION and LS strategies are in different locations checks are done separately for map\_isolation\_cell and map\_level\_shifter\_cell command as mention in 1) and 2)
  - b. ISOLATION and LS strategies are in same locations (self/parent/fanout) and in this scenario tool will search for ELS cells.
    - i. If isolation map\_cell is not defined, no violations are reported.

- ii. If isolation map\_cell has no ELS cells, VC LP reports ISO\_MAP\_TYPE with reason\_code:ELS\_NONE
- iii. If the ELS cell in level\_shifter map\_cell is not in isolation map\_cell, VC LP reports ISO\_MAP\_TYPE with reason\_code:ELS\_MISSING, and report the expected lib\_cell names
- 4. For a heterogeneous fanout path, if the isolation and level shifter strategies belong to different paths, details are provided in section [Heterogeneous Fanout Checks](#).

#### 5.1.14.1 Examples

❖ ISO\_MAP\_TYPE

- ◆ If pure isolation cell is required for an isolation strategy, but only ELS in the UPF map cells. For this case, the reasonCode debug field is with value PURE\_NONE.

```
Tag : ISO_MAP_TYPE
Description : The lib_cell specified for strategy [Strategy] on node
[StrategyNode] doesn't match the required type
StrategyNode : i_core_1/Q
Strategy : PD_CORE_1/ISO_2
ReasonCode : PURE_NONE
```

- ◆ If ELS cell is required for an isolation strategy but there is only pure isolation cell in the UPF map cells. For this case, the debug field reasonCode is with value ELS\_NONE.

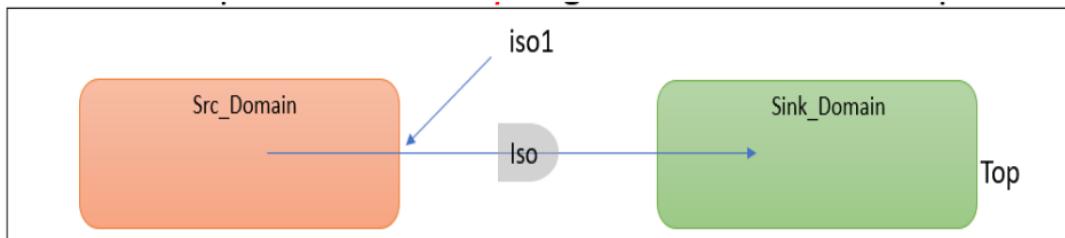
```
Tag : ISO_MAP_TYPE
Description : The lib_cell specified for strategy [Strategy] on node
[StrategyNode] doesn't match the required type
StrategyNode : i_core_4/Q
Strategy : PD_CORE_4/ISO_2
ReasonCode : ELS_NONE
```

- ◆ If ELS cell is required, and level shifter map\_cells has ELS, but it's not in isolation map\_cells. For this case, the reasonCode debug field is with value ELS\_MISSING, and report the ELS cell name.

```
Tag : ISO_MAP_TYPE
Description : The lib_cell specified for strategy [Strategy] on node
[StrategyNode]
doesn't match the required type
StrategyNode : i_core_5/Q
Strategy : PD_CORE_5/ISO_2
ReasonCode : ELS_MISSING
MappedCells
MappedCell : DMLSUAN12V
```

- ◆ When the ISO cell is in the source domain or a third power domain, then it requires a dual rail ISO cell. Therefore, in cases where a dual rail ISO cell is required, but only a single rail ISO cell is defined in the map\_isolation\_cell/use\_interface\_cell command, the ALL\_SINGLE\_RAIL reason code is reported in the ISO\_MAP\_TYPE violation.

Example



```
set_isolation iso1 -domain Src_Domain -location parent -applies_to outputs
map_isolation_cell iso1 -domain Src_Domain -lib_cells {singlerailiso}
```

The dual rail ISO cell is required, but only a single rail ISO cell is defined in the `map_isolation_cell/use_interface_cell` command. Therefore, the `ISO_MAP_TYPE` violation is reported with the `ALL_SINGLE_RAIL` reason code.

---

#### ISO\_MAP\_TYPE

---

```
Tag : ISO_MAP_TYPE
Description : The lib_cell specified for strategy [Strategy] on node
[StrategyNode] doesn't match the required type
Violation : LP:17
Strategy : PD1/iso1
Strategy Node : CORE1/out1
Reason Code : ALL_SINGLE_RAIL
```

- ◆ When the ISO cell is in the sink domain, then it requires a single rail ISO cell. Therefore, in cases where a single rail ISO cell is required, but a dual rail ISO cell is defined in the `map_isolation_cell/use_interface_cell` command, the `ALL_DUAL_RAIL` reason code is reported in the `ISO_MAP_TYPE` violation.
- ❖ LS\_MAP\_TYPE
  - ◆ If pure level shifter cell is required for a level shifter strategy according to the design crossover path info, but only ELS is found in map\_cells. The debug field reasonCode is with value `PURE_NONE` is reported.

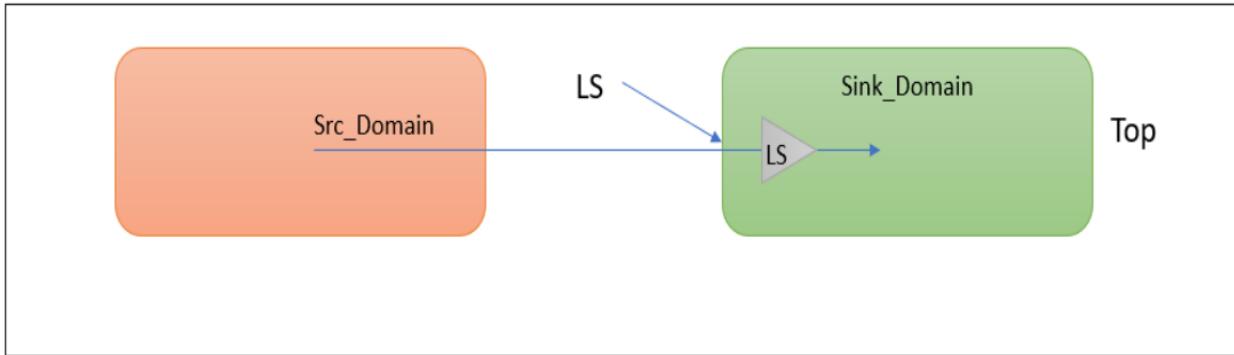
```
Tag : LS_MAP_TYPE
Description : The lib_cell specified for strategy [Strategy] on node
[StrategyNode] doesn't match the required type
StrategyNode : i_core_3/Q
Strategy : PD_CORE_3/LS_2
ReasonCode : PURE_NONE
```

- ◆ When the LS/ELS in the source domain, then `All_Source_Type_LS` (`std_cell_main_rail` is on input pin) is required. Therefore, in cases where `All_Source_Type_LS` is required, and only `All_Sink_Type_LS` is defined in the `map_level_shifter_cell/use_interface_cell` commands, then the `All_Source_Type_LS` reason code is reported in the `LS_MAP_TYPE` violation.

#### Example

Consider the following design and UPF:

```
set_level_shifter LS -domain Sink_Domain -location self -applies_to inputs
map_isolation_cell iso1 -domain Sink_Domain -lib_cells {All_Source_Type_LS}
```



Here, the *All\_Sink\_Type\_LS* is required but only *All\_Source\_Type\_LS* is defined. Therefore the *LS\_MAP\_TYPE* violation is reported with the *All\_SOURCE\_TYPE\_LS* reason code.

---

*LS\_MAP\_TYPE* (1 warning/0 waived)

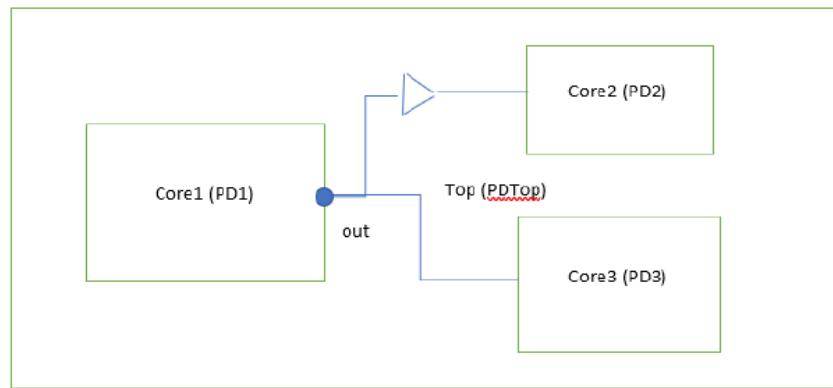
---

Tag : *LS\_MAP\_TYPE*  
 Description : The lib\_cell specified for strategy [Strategy] on node [StrategyNode] doesn't match the required type  
 Violation : LP:24  
 Strategy : PD1/ls1  
 StrategyNode : CORE1/out1  
 ReasonCode : *All\_SOURCE\_TYPE\_LS*

- ◆ **All\_SINK\_TYPE\_LS:** When the LS/ELS is in the sink domain, the *All\_Sink\_Type\_LS* (*std\_cell\_main\_rail* is on output pin) is required. Therefore, in cases where *All\_Sink\_Type\_LS* is required and only *All\_Source\_Type\_LS* is defined in the map command, the *All\_Sink\_Type\_LS* reason code will be flagged with *LS\_MAP\_TYPE* violation.
- ◆ **Third\_PowerDomain:** When the LS/ELS in third power Domain, then *Third\_powerDomain* reason code is reported with the *LS\_MAP\_TYPE* violation.

#### 5.1.14.2 Heterogeneous Fanout Checks

Consider the example in the following figure. For policy association node Core1/out, it has heterogeneous fanout, fanout1 path is to domain PD2, and fanout2 path is to domain PD3. Different strategies are required and defined on different fanout paths: fanout1 has an isolation strategy, while fanout2 requests level shifter cell.



- ❖ For fanout1, PD1 -> PDTOP -> PD2, it has isolation strategy defined.
- ❖ For fanout2, PD1 -> PD3, it requests ls cells, and with level shifter strategy defined.
- ❖ If the locations are fanout/parent/self, or one location is with automatic, the tool considers an ELS cell will be placed. So ELS\_NONE/ELS\_MISSING will be checked for the map\_iso command
- ❖ If the location does not match, different cells (iso and ls) could be inserted into different paths separately, no need of ELS. And pure isolation/ls cell is expected.

### 5.1.15 Support for Architectural Checks in Predictive Flow

The CORR\_CONTROL\_ISO and CORR\_CONTROL\_STATE architectural checks are used to identify any corruption in the control signal paths.

When a control signal is passing through one or more isolation cells, the control signal becomes clamped to a constant value when isolation is enabled, and thus is not independently controllable. In such scenarios, the CORR\_CONTROL\_ISO is reported.

When a control signal is corrupted by relatively off power rails while reaching the sink CORR\_CONTROL\_STATE is flagged.

For both these scenarios, in RTL level where the isolation strategies are available, but the cells are not instrumented, it is possible to predict any corruption that can be caused by the predicted isolation cells that can be instrumented later.

The architecture checker support in the predictive flow can be enabled to anticipate any corruption that might occur with these predicted isolation cells.

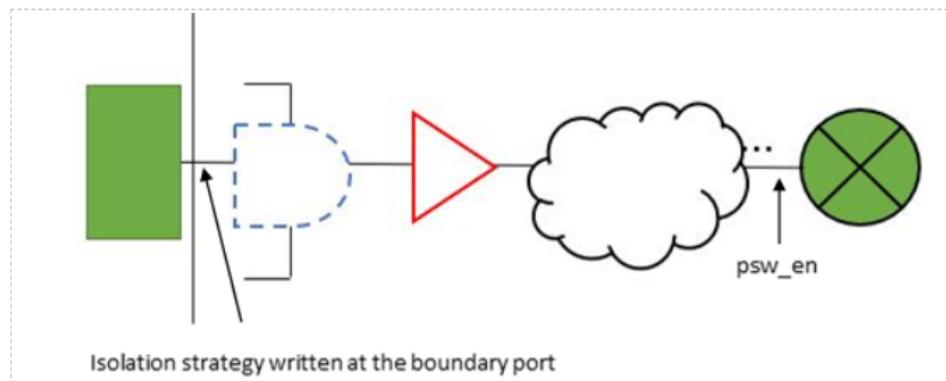
#### Use Model

The architecture checker support in the predictive flow can be enabled using the `lp_enable_virtual_protection` application variable.



#### Note

The VC-LP-ULTRA licenses required to enable the `lp_enable_virtual_protection` application variable

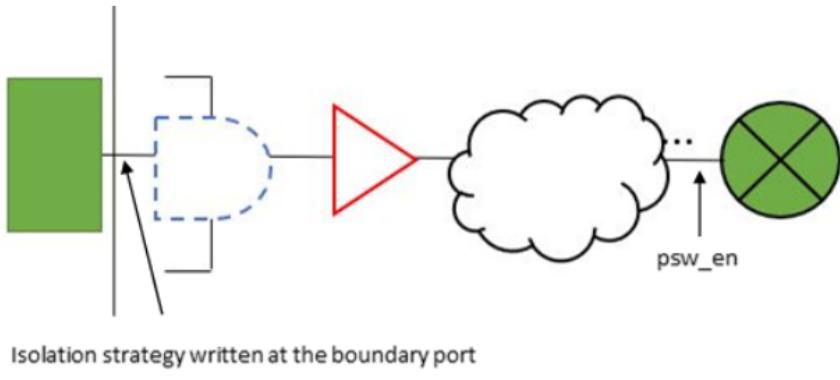
**Figure 5-44 CORR\_CONTROL\_ISO**

The psw\_en control signal is passing through a domain boundary where an isolation strategy is specified. This anticipates an isolation cell to be placed at the boundary, which causes the control signal to be clamped to a constant value when isolation is enabled. With the `lp_enable_virtual_protection` application variable is set to true, the architectural check is done considering the predicted isolation cell.

The CORR\_CONTROL\_ISO violation is reported with the debug field `IsoIsVirtual` : TRUE to indicate that the isolation cell is a predicted one.



**Note**  
The support is available for all control signals.

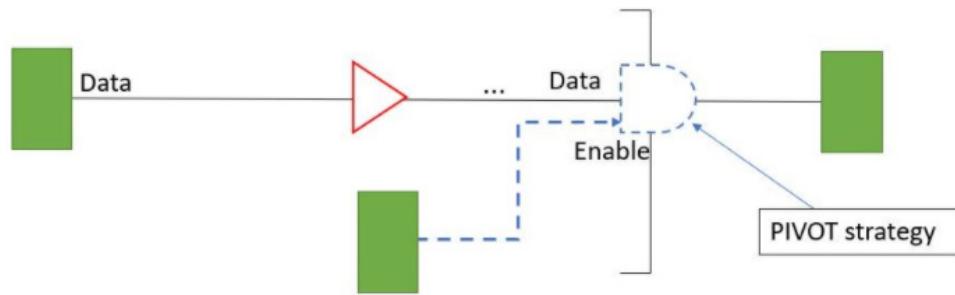
**Figure 5-45 Example for CORR\_CONTROL\_STATE**

The psw\_en control signal is passing through a domain boundary where an isolation strategy is specified, which has an isolation supply with a relatively off state compared to the sink. This anticipates an isolation cell to be placed at the boundary later which causes corruption due to its supply rails going off in relation to the sink.

With the `lp_enable_virtual_protection` is set to true, the architectural check is done considering this predicted isolation cell with relatively off supply.

The CORR\_CONTROL\_STATE violation is reported with the debug field `IsoStrategies`, `StrategySupplyInfo` that points to the predicted corruption source.

The support is available for all control signals. Behavior with hanging upf control signals



For the UPF control signals (iso\_enable, psw control, save, restore), which are not reaching the expected control pins or are hanging, these architectural checks are not done by default.

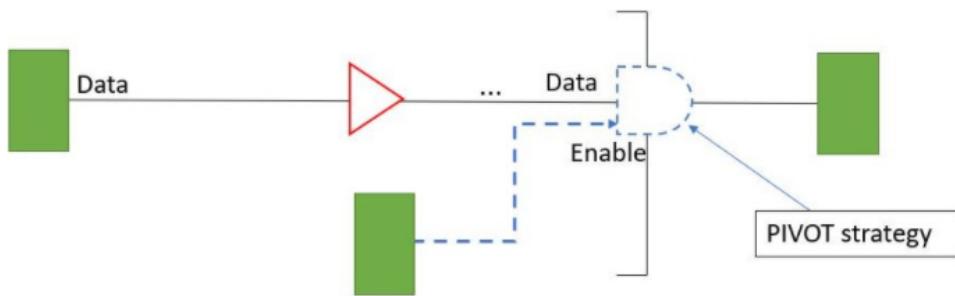
The architectural checks are done if the `lp_enable_arch_check_hanging_signals` application variable is set to true in such cases.

The predictive support is enabled in such scenarios when the `lp_enable_arch_check_hanging_signals` and `lp_enable_virtual_protection` application variables are set to true

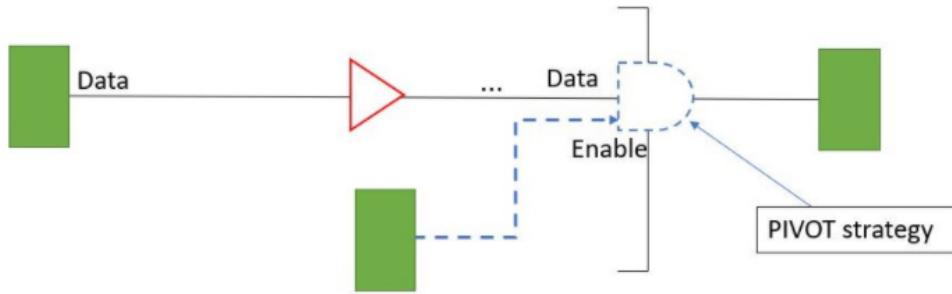
### 5.1.16 Gray Cloud FUNC Checks in Predictive Flow

The ISO\_\*\_FUNC and ISO\_\*\_STATE violations are gray cloud checks that suggest changes in supply rails of cells like buffers, LS, ISO, ELS when their supply states introduce functionally and/or electrically incorrect behavior.

Whether the scenario is a functional violation or an electrical violation is decided by the use of pivot isolation cells. The rightmost ISO lib\_cell in the crossover path is considered as the pivot node. If the violated cells (with OFF states) are in the left side of the pivot node, it is considered to be electrical safe, but with functional issues, and reports ISO\_\*\_FUNC. And if the violated cells are in the right side, or there is no ISO lib\_cells in the path, it is an electrical issue, and reports ISO\_\*\_STATE.



The same checks can be done in RTL mode with isolation strategies defined predicting isolation cells that will be implemented in future.



For this predictive check, the ISO virtual node (the ISO policy associated node) is considered equally as an ISO cell which computes the pivot node, and impacts these checks.

### Use model

The gray cloud FUNC checks in predictive flow are supported under the `lp_enable_virtual_protection` application variable set to true. The following violations are impacted in this flow:

- ❖ ISO\_BUFINV\_STATE/FUNC
- ❖ ISO\_LSINPUT\_STATE/FUNC
- ❖ ISO\_ELSINPUT\_STATE/FUNC
- ❖ ISO\_INPUT\_STATE/FUNC
- ❖ ISO\_OUTPUT\_STATE/FUNC
- ❖ ISO\_LSOUTPUT\_STATE/FUNC
- ❖ ISO\_ELSOUTPUT\_STATE/FUNC
- ❖ ISO\_SINK\_STATE



**Note**  
The VC-LP-ULTRA licenses required for enable the application variable.

### 5.1.17 Marking Design Ports as Supply Ports

Just by parsing netlist, VC LP does not consider VDDI and VSSI as PG pins, therefore, VC LP has introduced application variables using which you can mark design ports as supply ports.

- ❖ To mark pin as a supply, set the following application variable:  
`set_app_var configure_power_pin_rule "VDDI"`
- ❖ To mark pin as a ground, set the following application variable:  
`set_app_var configure_ground_pin_rule "VSSI"`

These application variables support space separated list of values. For example:

```
set_app_var configure_power_pin_rule "VDD* VZZ* VBB*"
set_app_var configure_ground_pin_rule "VSS* VTT* VUU*"
```

If the ports given in the application variable are not available in the design, the following warning message is displayed.

[Warning] CONFIG\_PWRGND\_PIN\_UNAVAIL: The top level ports '<Port list>', specified through 'configure\_power\_pin\_rule' and/or 'configure\_ground\_pin\_rule' app vars, cannot be found in the design  
Please specify correct top level port names

#### 5.1.17.0.1 Tags Introduced for this Support

The DESIGN\_SUPPLY\_NOUPF violation is introduced for this support. This check is performed only for the top module. The DESIGN\_SUPPLY\_NOUPF violation is reported when the UPF does not have create\_supply\_net, create\_supply\_port for a pin, but in the design it has been implemented physically.

#### 5.1.18 Support for Electrostatic Discharge Checks

VC LP supports new checks for Electro-Static Discharge (ESD). If two consecutive nodes are not working on the same supply net, then VC LP reports violations. The following new tags are added for this support.

- ❖ DESIGN\_POWER\_DIFF
- ❖ DESIGN\_GROUND\_DIF

These tags have severity *warning*, family *DesignConsistency*, and are disabled by default. They can be enabled using the `configure_tag -app LP` Tcl command.



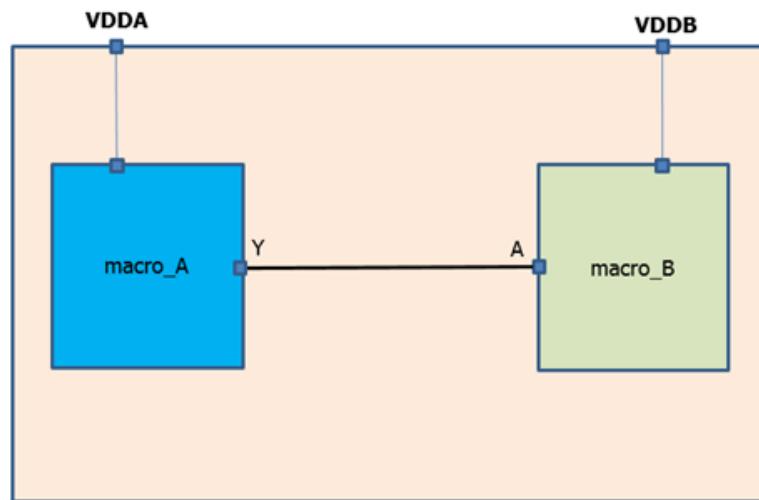
##### Note

When these checks are enabled, there can be potential run time increase because of the large number of DESIGN\_\*\_DIFF checks that are performed.

- ❖ These checks are strictly based on supply net only and not based on PST. These checks are for the immediate driver/load and not for global source/sink.
- ❖ If a driver and a load are working on power switch input and output port respectively or vice-versa, then the DESIGN\_\*\_DIFF violation is not reported.
- ❖ If the load/sink has attribute `is_esd_protected` set to true, then the violation DESIGN\_\*\_DIFF is not reported.
- ❖ If the load/sink is a diode cell, then the DESIGN\_\*\_DIFF violation is not reported.

#### 5.1.18.1 DESIGN\_POWER\_DIFF

This violation is reported if on a given crossover path two consecutive nodes are working on a different power net.

**Figure 5-46 DESIGN\_POWER\_DIFF**

The following is an example output of the DESIGN\_POWER\_DIFF violation.

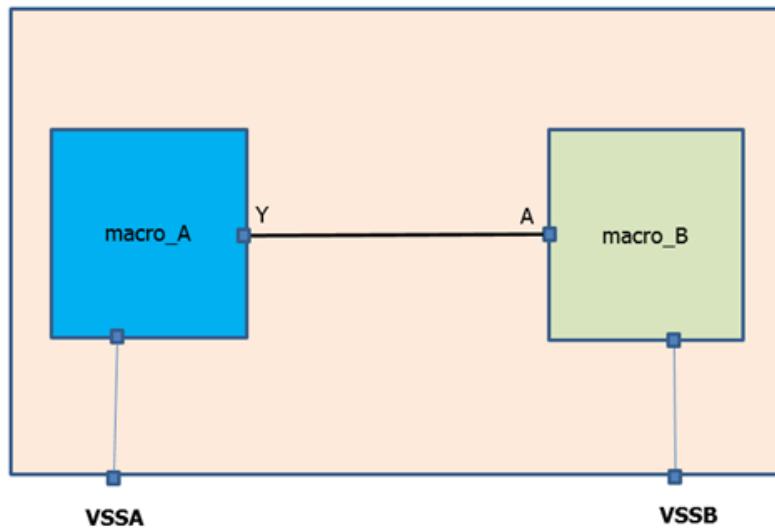
```

Tag : DESIGN_POWER_DIFF
Description : Power supplies for [Source] and [Sink] are physically different
Violation : LP:2
Source
  PinName : macro_A/Y
Sink
  PinName : macro_B/A
SourceInfo
  PowerNet
    NetName : VDDA
    NetType : UPF
    PowerMethod : FROM_UPF_CSN
  GroundNet
    NetName : VSSA
    NetType : UPF
    GroundMethod : FROM_UPF_CSN
SinkInfo
  PowerNet
    NetName : VDDB
    NetType : UPF
    PowerMethod : FROM_UPF_CSN
  GroundNet
    NetName : VSSB
    NetType : UPF
    GroundMethod : FROM_UPF_CSN

```

### 5.1.18.2 DESIGN\_GROUND\_DIFF

The DESIGN\_GROUND\_DIFF is reported if on a given crossover path two consecutive nodes are working on a different ground net.

**Figure 5-47 DESIGN\_GROUND\_DIFF**

The following is an example output of the DESIGN\_GROUND\_DIFF violation.

```

Tag : DESIGN_GROUND_DIFF
Description : Ground supplies for [Source] and [Sink] are physically different
Violation : LP:1
Source
  PinName : macro_A/Y
Sink
  : macro_B/A
SourceInfo
  PowerNet
    NetName : VDDA
    NetType : UPF
  PowerMethod : FROM_UPF_CSN
  GroundNet
    NetName : VSSA
    NetType : UPF
  GroundMethod : FROM_UPF_CSN
SinkInfo
  PowerNet
    NetName : VDDB
    NetType : UPF
  PowerMethod : FROM_UPF_CSN
  GroundNet
    NetName : VSSB
    NetType : UPF
  GroundMethod : FROM_UPF_CSN

```

### 5.1.18.3 ESD Checks for Fanout scenario

For a fanout scenario:

- ❖ If any fanout sink does not have the `is_esd_protected: true` attribute, but both the supply rails (power and ground) are matching with the CDM cell connected to the same fanout path, then DESIGN\_POWER\_DIFF and DESIGN\_GROUND\_DIFF violations are not reported for the non-CDM fanout sink.

- ❖ If only the power net matches with the CDM cell pin power net, then the DESIGN\_GROUND\_DIFF violation is reported.
- ❖ If only the ground net matches with the CDM cell pin ground net, then the DESIGN\_POWER\_DIFF violation is reported.
- ❖ If none of the rail matches with the CDM cell pin nets, then both the DESIGN\_GROUND\_DIFF and DESIGN\_POWER\_DIFF violations are reported.

VC LP considers a cell pin as CDM pin, if it has the `is_esd_protected` pin level attribute set to true.

### Example

Consider the following fanout scenario:

```
ua/out (VDD1, VSS1) --> CDM/in (VDD2, VSS2, is_esd_protected: true)
                           | -> uc/in1 (VDD2, VSS2)
                           | -> ud/in1 (VDD2, VSS3)
                           | -> ue/in1 (VDD3, VSS2)
                           | -> uf/in1 (VDD3, VSS3)
```

VC LP reports following violations for the example scenario:

- ❖ The DESIGN\_GROUND\_DIFF violation is reported for ud/in1.
- ❖ The DESIGN\_POWER\_DIFF violation is reported for ue/in1.
- ❖ The DESIGN\_POWER\_DIFF and DESIGN\_GROUND\_DIFF violations are reported for uf/in1.
- ❖ No DESIGN\_\*\_DIFF violations are reported for CDM/in.
- ❖ No DESIGN\_\*\_DIFF violations are reported for uc/in1.

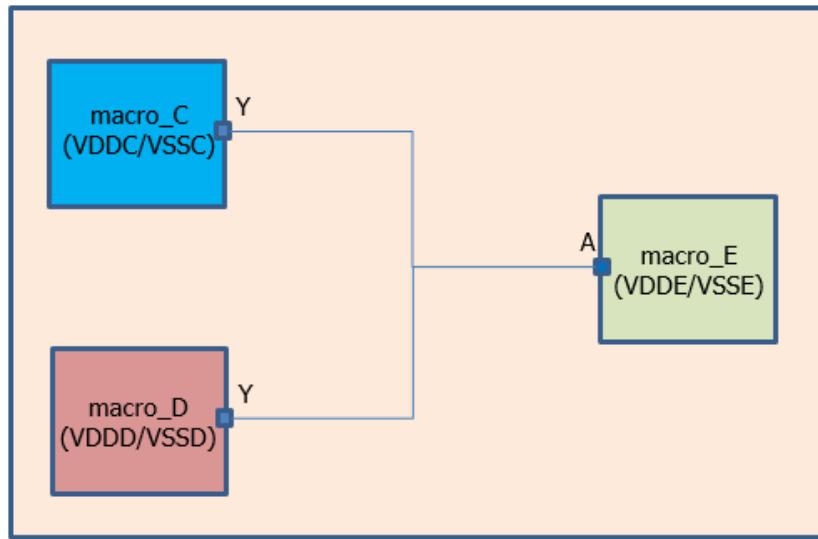
### Limitation

In above example, consider if supplies VDD2 and VDD3 are related through PSW strategy/cell such that one is input supply of PSW and another is output supply of PSW, then ideally VC LP should not report DESIGN\_POWER\_DIFF violations for any of the fanout path, however, VC LP reports the DESIGN\_POWER\_DIFF violation for such scenarios. The same limitation is applicable for the DESIGN\_GROUND\_DIFF violation.

#### 5.1.18.4 Limitation

Consider the following multi-driver scenario:

```
macro_C/Y (VDDC/VSSC) --- |
                           macro_D/Y (VDDD/VSSD) --- | --> macro_E/A (VDDE/VSSE)
```

**Figure 5-48 Multi-driver Scenario Example**

For such scenarios, ideally VC LP must report two violations DESIGN\_\*\_DIFF:

- ❖ From macro\_C/Y to macro\_E/A
- ❖ From macro\_D/Y to macro\_E/A

Currently, VC LP reports a single violation for path the macro\_D/Y to macro\_E/A. The violation for path from macro\_C/Y to macro\_E/A is not reported.

### 5.1.19 The PG\_DIODE\_EXCESS and the PG\_PASSTRAN\_EXCESS Violation

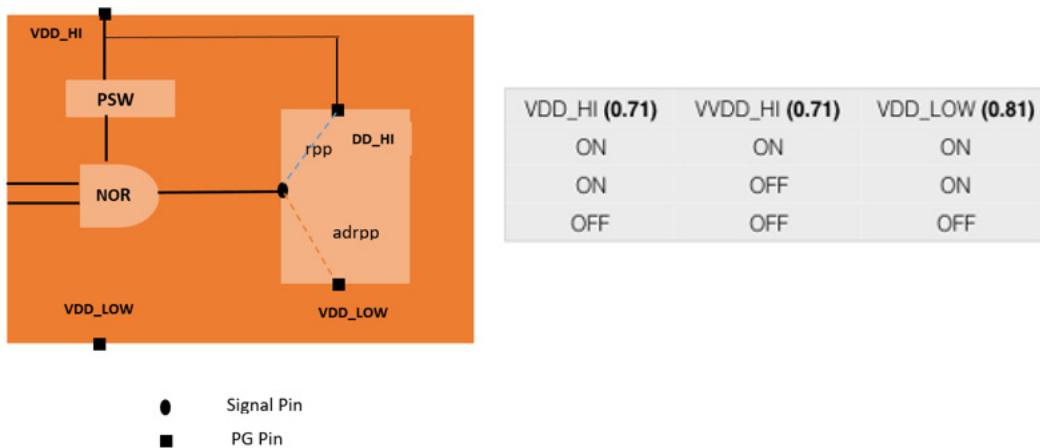
Previously, VC LP did not flag any violation when there is excess voltage on the diode when compared to its driver supply. As illustrated in diagram, the driver supply is less compared to the ADRPP and no leak path exists for the diode. To resolve this issue, the PG\_DIODE\_EXCESS violation is introduced.

The `lp_user_has_pass_gate_list` application variable is introduced to provide a list of lib cell pins which to be considered having `has_pass_gate` attribute. The `has_pass_gate` is a liberty attribute which is usually set on lib pins within liberty model. With this new variable, you can provide a list of lib pins to be considered as they have `has_pass_gate` attribute, even if it is not defined in the liberty model.

#### Example

```
set_app_var lp_user_has_pass_gate_list "cell1/pinA    cell1/pinB cell2/PinA"
```

The PG\_DIODE\_EXCESS violation is introduced to help to connect to the correct supplies.



The PG\_DIODE\_EXCESS violation is reported when the diode instance supply is more than the supply of the driver of the diode pin.

```

PG_DIODE_EXCESS (1 error/0 waived)
-----
Tag      : PG_DIODE_EXCESS
Description : Diode pin [CellPin] of instance [Instance] ([Cell]) is driven by Supply [SourceInfo], which is at a lower voltage than the diode supply [SinkInfo] in state [States], which can cause leakage current.
Violation   : LP:4
CellPin    : INP
Instance   : ub/ADRPP_MAC
Cell       : FOOTPRINT_DIODE
SourceInfo
PowerNet
  NetName  : VDDBLK1
  NetType   : Design/UPF
  PowerMethod : FROM_PG_NETLIST
SinkInfo
PowerNet
  NetName  : VDDBLK2
  NetType   : Design/UPF
  PowerMethod : FROM_PG_NETLIST
States
State     : net1/c1

```

Similarly, if the liberty cell has attribute `has_pass_gate` or `is_pass_gate` such cell is considered as pass transistor. If the supply of transistor cell is more than the supply of the driver of the pass transistor pin, then the `PG_PASSTRA_N_EXCESS` violation is reported.

```
=====
PG_PASSTRA_N_EXCESS (12 errors/0 waived)

Tag      : PG_PASSTRA_N_EXCESS
Description : Pass transistor pin [CellPin] of instance [Instance] ([Cell]) is driven by Supply [SourceInfo] , which is at a lower voltage than the pass transistor supply [SinkInfo] in state [States], which can cause leakage current.
Violation   : LP:25
CellPin     : A
Instance    : CORE3/buf_i1
Cell        : DMBF
SourceInfo
PowerNet
  NetName   : VDD
  NetType    : Design/UPF
  PowerMethod : FROM_UPF_POWER_DOMAIN
SinkInfo
PowerNet
  NetName   : VDD_PG
  NetType    : Design/UPF
  PowerMethod : FROM_PG_NETLIST
States
State      : top_pst/s1
State      : CORE3/macro_pst/s1
DesignNet
NetName    : in3
NetType    : Design
```

## 5.1.20 Device Connectivity Checks

VC LP has introduced the Design Connectivity Checker (also known as DCC). VC LP violations like `ISO_INST_MISSING`, `LS_INST_MISSING`, and so on depend on the crossover generation. If the crossover is not generated due to any reason, these violations for those crossover will never trigger. To remove this dependencies on crossover generation, the following new violations are introduced.

- ❖ `ISO_DEVICE_STATE`
- ❖ `LS_DEVICE_STATE`
- ❖ `UPF_DEVICE_STATE`
- ❖ `DCC_PIN_CONST`

All these violation tags are by-default disabled. Use `configure_tag -app LP -enable` to enable them.



### Note

This support is available under VC-STATIC-BETA license.

### 5.1.20.1 Use Model

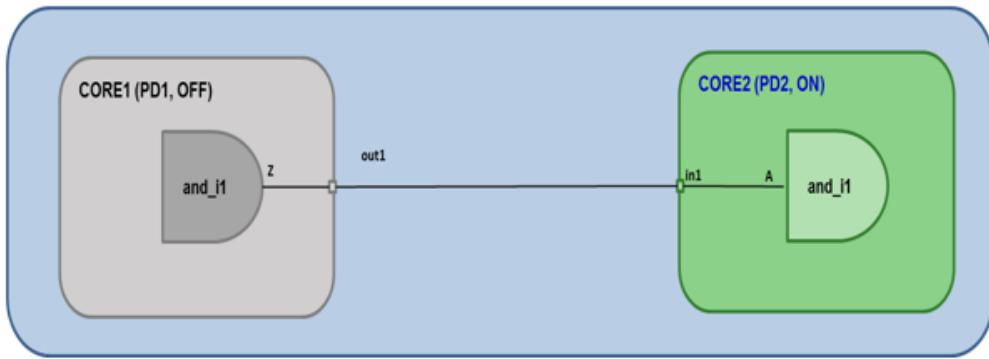
Use the `check_lp_devices` command to check for the new violations depending on their enable/disable status.

#### Example

```
configure_tag -app LP -tag " ISO_DEVICE_STATE LS_DEVICE_STATE UPF_DEVICE_STATE
DCC_PIN_CONST " -enable
read_file -format verilog -top -netlist " Design.v"
read_upf Design.upf
check_lp -stage design
check_lp_devices
```

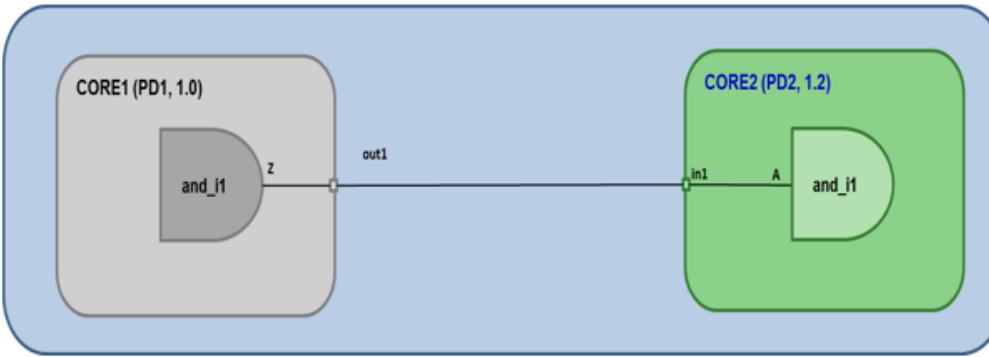
### 5.1.20.2 Violations Introduced

- ❖ `ISO_DEVICE_STATE`



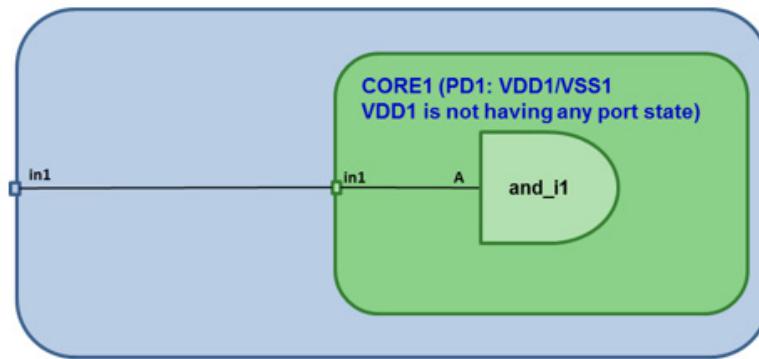
```

Tag : ISO_DEVICE_STATE
Description : Missing or insufficient isolation at crossing from source [PinName]
to pin [CellPin] of cell [Instance] ([Cell])
Violation : LP:2
PinName : CORE1/and_i1/z
CellPin : CORE2/and_i1/A
States
State : top_pst/s2
❖ LS_DEVICE_STATE
  
```



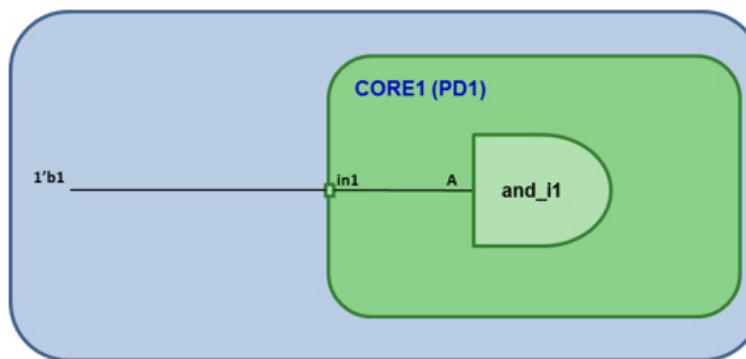
```

Tag : LS_DEVICE_STATE
Description : Unprotected voltage difference at crossing from source [PinName] to
pin [CellPin] of cell [Instance] ([Cell])
Violation : LP:6
PinName : CORE1/and_i1/z
CellPin : CORE2/and_i1/A
States
State : top_pst/s2
❖ UPF_DEVICE_STATE
  
```



Tag : UPF\_DEVICE\_STATE  
 Description : Supply Difference found from source [PinName] to pin [CellPin] of cell [Instance] ([Cell]) but [UPFSupply] driving pin has no states defined.  
 Violation : LP:143  
 PinName : in1  
 CellPin : CORE1/and\_i1/A  
 UPFSupply : VDD1

- ❖ DCC\_PIN\_CONST



Tag : DCC\_PIN\_CONST  
 Description : Pin [CellPin] of cell [Instance] ([Cell]) is driven by constant.  
 Violation : LP:24  
 CellPin : A  
 Instance : and\_i1  
 Cell : DMAN2

### 5.1.20.3 Support for Comparison Script

Along with the four new violations, the DCC also supports comparison script that helps to compare these new four violations with the existing electrical violations.

To perform this comparison, perform the following steps:

- ❖ Run the test case and dump the report in a xml format using the following command:  
`report_violations -app LP -verbose -file report_lp.xml -xml -limit 0`
- ❖ The following command performs the comparison and dumps seven reports in the specified path.  
`vc_static_shell> lp_compare_dcc_result -report report_lp.xml -dump_path .`

The following seven reports are generated:

- ❖ match\_summary.csv: Contains the summary of comparison
- ❖ dcc\_vclp\_full\_match.csv: Contains the violations along with their ids that are matching
- ❖ only\_dcc.csv: Contains violation ids that are reported by DCC only, and there is no matching violation reported by normal checker.
- ❖ only\_vclp.csv: Contains violation ids that are reported by VC LP only, and there is no matching violation reported by DCC.
- ❖ src\_sink\_supply\_match.csv: Contains violations along with their ids that have matching source and sink supply.
- ❖ src\_only.csv: Contains violation ids that match based on source only.
- ❖ sink\_only.csv: Contains violations ids that match based on sink only.

#### Notes

- ❖ The Constant propagation will be performed in DCC under check\_lp\_devices-enable\_constant\_propagation.
- ❖ Violation DCC\_PIN\_CONST will be reported when source is a literal constants or TIE\_CELL (not for PG\_ports).
- ❖ DCC is not honoring terminal boundary, so checks will happen as if terminal boundary ports are pass-through.
- ❖ DCC is not honoring supply resolution from SPA -iso\_source/sink
- ❖ DCC is not honoring the checks which normal checker performs under the generate\_upf\_ctrl\_signal\_crossover application variable.
- ❖ There are open issues with comparator scripts, hence currently 100% match cannot be happen between DCC and Default checker violations.

### 5.1.21 Support for Controlling Signal Connectivity Checks

You can control VC LP behavior on control signal connectivity checks using an application variable.

Previously, the retention save/restore, PSW enable signals, and ISO enable-signals find their effective roots using fan-in and fan-out traversals from UPF specified signals while skipping the buffers and inverters.

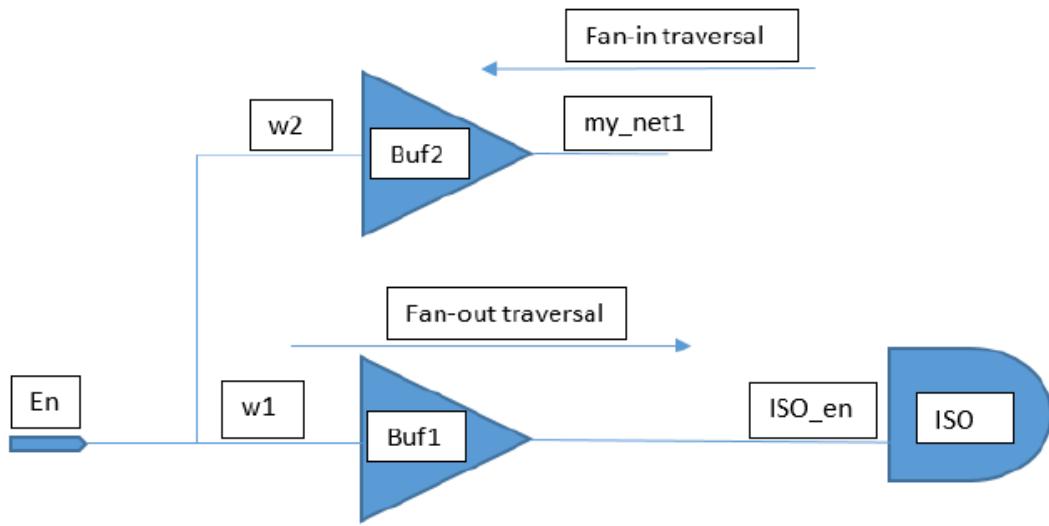
#### Example

```
set_isolation ISO_1 -domain TOP -isolation_supply_set SS_A -clamp_value 1 -applies_to
outputs -isolation_signal W1
```

The lp\_traverse\_control\_fanin application variable is introduced to disable and enable fan-in traversals. When this application variable is set FALSE, the fan-in traversals does not happen for the retention save/restore, PSW enable signals, and ISO enable-signals. By default, the lp\_traverse\_control\_fanin application variable is set to TRUE.

#### Example

Consider the example shown in [Figure 5-49](#).

**Figure 5-49 Example for lp\_traverse\_control\_fanin Application Variable**

Consider the following UPF:

```
set_isolation ISO_1 -domain TOP -isolation_supply_set SS_A -clamp_value 1 -applies_to
outputs -isolation_signal my_net1
```

- ❖ When the `lp_traverse_control_fanin` application variable is set to true, the fan-in traverse is enabled, the isolation enable signal of the strategy and isolation enable signal of the cell has the same root, and no violation is reported.
- ❖ When the `lp_traverse_control_fanin` application variable is set to false, the fan-in traversal is disabled, and the isolation enable signal is not able to find the root when connected to the isolation enable cell pin. Therefore, ISO\_CONTROL\_CONN violation is reported for this scenario.

## 5.1.22 Support for Multiple Power Domains in a Single Voltage Area

VC LP allows you to specify that multiple power domains in a single voltage area. This way, all the domains inherit the same primary power and power switch strategies (if any). This is safe when the domains all have switched supplies derived from the same input supply, using the same control signals.

### 5.1.22.1 Use Model

VC LP has introduced the following `set_design_attribute` attribute for sharing domains:

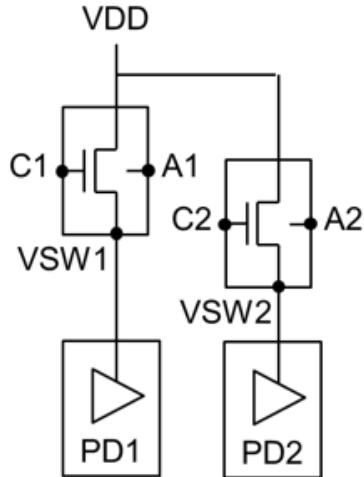
```
set_design_attributes -elements . -attribute {shared_voltage_area{{PD1 PD2}}}
```

**Where:**

- ❖ PD1 refers to the master domain
- ❖ PD2 refers to the slave domain
- ❖ The `-elements` only allows `.` (dot) as argument, and the `attribute` argument should be list of lists.

### 5.1.22.2 Example

In the UPF+RTL, consider two power domains are defined, each with their own power switch strategy as shown in the following figure:



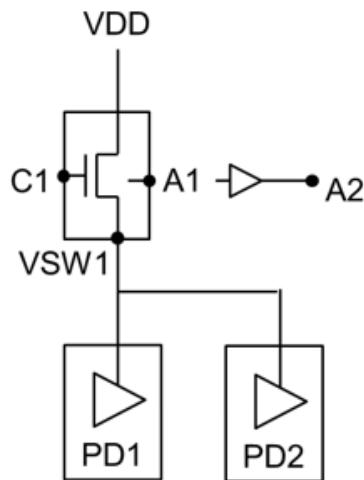
If you want PD1 and PD2 to be implemented in the same voltage area, sharing the same domain primary and the same power switches. You can provide the following new UPF attribute:

```
set_design_attributes -elements . -attribute {shared_voltage_area{{PD1 PD2}}}
```

Where:

- ❖ The first list element PD1 is a master domain
- ❖ PD2 and any other domains in the list are slave domains
- ❖ Slaves are implemented using the master domain primary
- ❖ Slave power switch strategies are ignored, except any PSW ack is driven by the master ack.

The resulting PG netlist after applying the attributes is as follows:



### 5.1.22.3 Parser Messages Introduced

The following parser messages are introduced for the `set_design_attribute -attribute {shared_voltage_area{{PD1 PD2}}}` attribute:

- ❖ If the first domain is invalid, the attribute cannot be applied, the following error message is reported, and the entire list is discarded:

*FIRST\_DOMAIN\_INVALID*

[Error] *First domain in shared\_voltage\_area invalid*

#### Example

```
set_design_attributes -attribute {shared_voltage_area {{da db} {dc dd}}}
```

In this example, suppose da is an invalid domain, but db, dc, dd are valid domains. Then no sharing happens for db, but sharing still may apply for dc, dd.

- ❖ If the second or the latter domain is invalid, the attribute is still applied to all the domains that are valid, and the following error message is reported:

*SHARED\_DOMAIN\_INVALID*

[Warning] *Shared domain in shared\_voltage\_area invalid*.

#### Example

```
set_design_attributes -attribute {shared_voltage_area {{da db dc}}}
```

Suppose db is invalid, but da, dc is valid, then sharing still may apply for da (as the master) and dc (as the only slave)

The SHARED\_DOMAIN\_INVALID and FIRST\_DOMAIN\_INVALID parser message are reported in the following scenarios:

- ◆ Domain is undefined
- ◆ Domain with zero PSW strategies
- ◆ Domain with more than one PSW strategies
- ◆ Domain has PSW strategy which contains more than one input supplies
- ◆ Domain has PSW strategy which contains more than one control ports
- ❖ If a list of lists is not specified, the following warning message is reported:

*SHARED\_AREA\_LIST\_FORMAT*

[Warning] *Attribute shared\_voltage\_area expects list of lists*

#### Example

```
set_design_attributes -attribute {shared_voltage_area {da db}}
```

A list of lists is required, not a simple list.

- ❖ If the domain specified in the `shared_voltage_area` is already shared, the following warning message is reported:

*SHARED\_DOMAIN\_SHARED*

[Warning] *Shared domain in shared\_voltage\_area is already shared*

#### Example

```
set_design_attributes -attribute {shared_voltage_area {{da db} {dc db dd}}}
```

A slave is assigned to one master, and later assigned to a second master. The assignment to the second master is ignored. It is equivalent to `{da db} {dc dd}`.

- ❖ If a master domain is specified twice with all the slave domain combined, the following warning message is reported:

***COMPAT\_SHARED\_DOMAIN***

[Warning] Not all tools allow reuse of a shared domain

**Example**

```
set_design_attributes . -attribute {shared_voltage_area {{da db} {da dc}}}
```

One master is given twice, with different slave lists. All the slaves are combined and is equivalent to a single list {da db dc}.

- ❖ If the supply in the shared voltage area is not completely shared, the following warning message is reported:

***SHARED\_DOMAIN\_UNSHARED***

[Warning] Supply in shared\_voltage\_area is not completely shared

**Example**

```
set_domain_supply_net db1 -primary_power db
set_domain_supply_net db2 -primary_power db
set_design_attributes -attribute {shared_voltage_area {{da db1 dc}}}
```

Reimplementing the primary of db1 using the primary of da involves splitting the supply net db to preserve the primary of db2. The incompletely shared domain is ignored. It is equivalent to the list {da dc}.

#### 5.1.22.4 New Violations Introduced

The following violations are introduced for this support:

- ❖ The PSW\_SUPPLY\_SHARED Violation
- ❖ The PSW\_CONTROL\_SHARED Violation
- ❖ The PSW\_OUTPUT\_SHARED Violation
- ❖ The PSW\_INST\_SHARED Violation

##### 5.1.22.4.1 The PSW\_SUPPLY\_SHARED Violation

The PSW\_SUPPLY\_SHARED violation is reported when a slave PSW strategy has an input supply which is not equivalent to the PSW strategy of its master domain. This violation is reported at the UPF stage with severity *warning*. If input supplies have equivalent *pst* states, "PstEquiv" debug field returns as True. If not it returns as False.

**Example**

```
set_design_attributes -elements . -attribute {shared_voltage_area {{PD1 PD2}}}
set_domain_supply_net PD1 -primary_power VSW1
set_domain_supply_net PD2 -primary_power VSW2
create_power_switch s1 -domain PD1 \
-input_supply_port {VDD VDDX} \
-output_supply_port {VDDSW VSW1} ...
create_power_switch s2 -domain PD2 \


```

The following is an example snippet of the PSW\_SUPPLY\_SHARED violation:

```
Tag : PSW_SUPPLY_SHARED
Description : Power switch strategies [PswStrategies] are shared with an
incompatible input supply [UPFSupply] of strategy [Strategy]
PswStrategies
```

```

Strategy      : s2
UPFSupply    : VDDX
Strategy      : s1
Supplies
  UPFSupply   : VDDY
PstEquiv     : True

```

#### 5.1.22.4.2 The PSW\_CONTROL\_SHARED Violation

The PSW\_CONTROL\_SHARED violation is reported if the control port of a slave PSW strategy does not match the PSW strategy of its master domain. This violation is reported at the UPF stage with severity *warning*.

##### Example

```

set_design_attributes -elements .-attribute {shared_voltage_area {{PD1 PD2}}}
set_domain_supply_net PD1 -primary_power VSW1
set_domain_supply_net PD2 -primary_power VSW2
create_power_switch s1 -domain PD1 \
-control_port {ENA ena1} ...
create_power_switch s2 -domain PD2 \
-control_port {ENA ena2} ...

```

The following is an example snippet of the PSW\_SUPPLY\_SHARED violation:

```

Tag          : PSW_CONTROL_SHARED
Description   : Power switch strategies [PswStrategies] are shared with a different
control input [UPFNet] of strategy [Strategy]
PswStrategies
  Strategy    : s2
  UPFNet
    NetName    : ena1
    NetType     : Design/UPF
  Strategy    : s1
  UPFNets
    UPFNet
      NetName    : ena2
      NetType     : Design/UPF

```

#### 5.1.22.4.3 The PSW\_OUTPUT\_SHARED Violation

The PSW\_OUTPUT\_SHARED violation is reported when a slave PSW strategy has an output supply which is not equivalent to the PSW strategy of its master domain. This violation is reported at the UPF stage with severity *warning*. If output supplies have equivalent pst states, "PstEquiv" debug field returns as True. If not, it returns as False.

##### Example

```

set_design_attributes -elements .-attribute {shared_voltage_area {{PD1 PD2}}}
set_domain_supply_net PD1 -primary_power VSW1
set_domain_supply_net PD2 -primary_power VSW2
create_power_switch s1 -domain PD1 \
-input_supply_port {VDD VDDX} \
-output_supply_port {VDDSW VSW1} ...
create_power_switch s2 -domain PD2 \


```

The following is an example snippet of the PSW\_SUPPLY\_SHARED violation:

```

Tag : PSW_OUTPUT_SHARED
Description : Power switch strategies [PswStrategies] are shared with an
incompatible output supply [UPFSupply] of strategy [Strategy]
PswStrategies
  Strategy : s2
  UPFSupply : VSW1
  Strategy : s1
  Supplies
    UPFSupply : VSW2
  PstEquiv : False

```

#### 5.1.22.4.4 The PSW\_INST\_SHARED Violation

When a domain is shared as a slave, the downstream tools should not implement any power switch instances for the slave PSW strategies. Only PSW strategies of master should be implemented. However, sometimes, some PSW instances may match with the slave PSW strategies. If some PSW instances may match with slave PSW strategies, VC LP allows associating such PSW strategies to the slave strategies if that is the best match. However, VC LP reports PSW\_INST\_SHARED violation in such scenarios. This violation is reported at the PG stage with severity warning.

##### Example

```

set_design_attributes -elements .-attribute {shared_voltage_area {{PD1 PD2}}}
set_domain_supply_net PD1 -primary_power VSW1
set_domain_supply_net PD2 -primary_power VSW2
create_power_switch s1 -domain PD1 \
  -input_supply_port {VDD VDDX} \
  -output_supply_port {VDDSW VSW1} ...
create_power_switch s2 -domain PD2 \
  -input_supply_port {VDD VDDY} \
  -output_supply_port {VDDSW VSW2} ...

```

Suppose there is a PSW instance associated with s2 slave psw strategy. The following is an example snippet of the PSW\_INST\_SHARED violation:

```

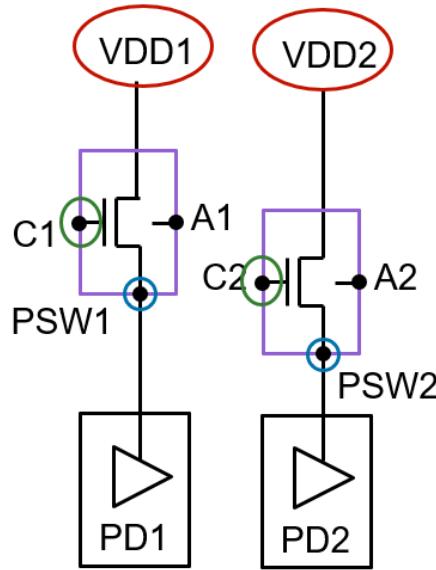
Tag : PSW_INST_SHARED
Description : Power switch instance [Instance] associated with shared strategy
[Strategy]
  Instance : psw2
  Strategy : s2

```

#### 5.1.22.4.5 Examples

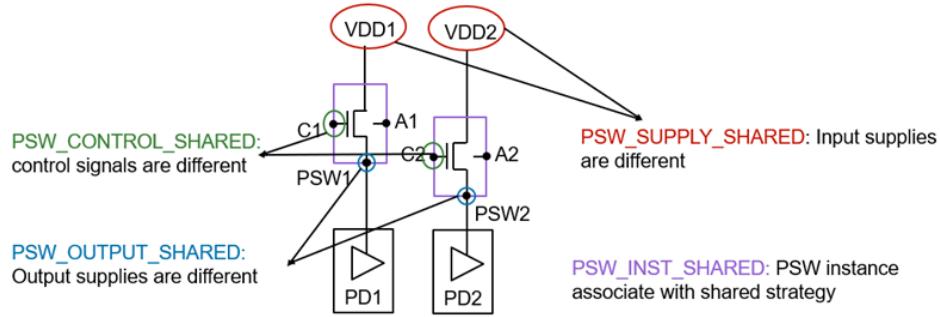
##### Example 1

Consider the following design:



```
set_design_attributes -elements . -attribute {shared_voltage_area{{PD1 PD2}}}
```

The following VC LP violations are reported:



## Example 2

Consider the following UPF snippet

```
set_domain_supply_net PD1 -primary_power VDD1_I
set_domain_supply_net PD2 -primary_power VDD2_I

create_power_switch PSW1 -domain PD1 -input_supply_port { VDD VDD1 } -
output_supply_port { VDDSW VDD1_I } -control_port { NSLEEPIN1 ctrl1 } -on_state { on1 VDD
{NSLEEPIN1} }
create_power_switch PSW2 -domain PD2 -input_supply_port { VDD VDD2 } -
output_supply_port { VDDSW VDD2_I } -control_port { NSLEEPIN1 ctrl2 } -on_state { on1 VDD
{NSLEEPIN1} }

set_design_attributes -elements . -attribute {shared_voltage_area{{PD1 PD2}}}
```

The following violations are reported:

- ❖ PSW\_SUPPLY\_SHARED

- ❖ PSW\_OUTPUT\_SHARED
- ❖ PSW\_CONTROL\_SHARED

#### 5.1.22.5 Tcl Commands for Querying Shared Domains and Power Switch Strategies

You can use the `get_attribute [get_power_switch_strategies <strategy_name>] shared_domain` command to find the parent strategy. If the specified strategy is a parent, then the command return NULL. If strategy is a child, the command reports its parent domain. The same command works for power domains.

##### Example

Consider the following UPF, where PSW1 output supply is PD1 primary, and PSW2 output supply is PD2 primary.

```
set_design_attributes -elements . -attribute shared_voltage_area {{ PD1 PD2 }}
```

The following is the output of the Tcl Command:

- ❖ `get_attribute [get_power_switch_strategies PSW1] shared_domain`  
Result: NULL
- ❖ `get_attribute [get_power_switch_strategies PSW2] shared_domain`  
Result: PSW1

#### 5.1.22.6 Limitations on Support for Multiple Power Domains in a Single Voltage Area

The support for multiple power domains in a single voltage area feature has the following limitations:

- ❖ The `set_design_attribute shared_voltage_area` UPF command is not dumped in the correct format in the `expanded_tokens.upf` file. Hence, it is unable to read back the dumped UPF.
- ❖ Multi-input supply PSW strategies are not supported.
- ❖ Multi-control port PSW strategies are not supported.
- ❖ Domains with more than two power switch strategies is not supported.
- ❖ The `set_design_attribute shared_voltage_area` attribute is unable to share two power switches when their outputs are electrically equivalent.

##### Example UPF

```
set_design_attributes -elements . -attribute {shared_voltage_area {{PD1 PD2}}}

create_power_switch PSW1 -domain PD1 -input_supply_port { VDD VDD1 } -output_supply_port
{ VDDSW VDD1_I } -control_port { NSLEEPIN1 ctrl } -on_state { on1 VDD {NSLEEPIN1} }

create_power_switch PSW2 -domain PD2 -input_supply_port { VDD VDD2 } -output_supply_port
{ VDDSW VDD2_I } -control_port { NSLEEPIN1 ctrl } -on_state { on1 VDD {NSLEEPIN1} }

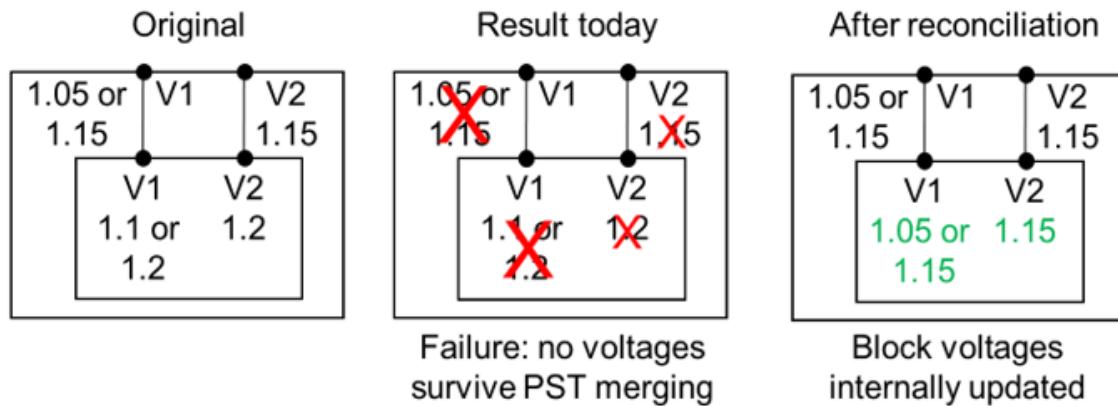
set_equivalent -nets {PSW1/VDD PSW2/VDD_I} -function_only
False FIRST_DOMAIN_INVALID parser warning message is reported.
```

#### 5.1.23 Support for Voltage reconciliation

When IP's are designed, voltage values for each supply are coded into the UPF. When the IP's are integrated into an SOC, the SOC voltages may be different.

Today, the IP designers must edit their UPF to change the voltages, or implementation cannot proceed. This support enhances VC to automatically reconcile the IP voltages to match what the

SOC and IP designers would agree upon. After reconciliation, SOC voltage values propagate to block level block level power state table will be updated internally.



### 5.1.23.1 Setting Up for Voltage Reconciliation

1. To enable voltage reconciliation set the following application variable.

```
set_app_var lp_volt_recon_method voltage
```

The `lp_volt_recon_method` application variable can be set to one of the five different modes for voltage reconciliation.

- ◆ `Off`: Indicates no reconciliation will be done.
- ◆ `All`: Reconciliation works on both voltage values and OFF states.
- ◆ `voltage`: It performs reconciliation on voltage values only.
- ◆ `voltage_automatic`: See section [Voltage Reconciliation Across all Scopes](#).
- ◆ `all_automatic`: See section [Voltage Reconciliation Across all Scopes](#).

2. Set the voltage threshold using the `set_variation` UPF command.

```
set_variation [-supply {[*]{*}supply_net_list{[*]{*}}}] -tolerance { vboth | vlow
** vhigh }
```

3. Set the following attribute on the appropriate modules where reconciliation is needed.

```
set_design_attributes -models M -attribute upf_reconcile_boundary
"reconcile_voltages"
```

### 5.1.23.2 Voltage Reconciliation Across all Scopes

When you prefer to reconcile PST voltage states across all the scopes use the following setting:

```
set_app_var lp_volt_recon_method voltage
set_variation -tolerance 999
set_design_attributes -models <model> -attribute upf_reconcile_boundary
"reconcile_voltages"
```

For better usability, VC LP allows you enable you to reconcile PST voltage states across all the scopes either by using the following application variable or the UPF attribute.

```
set_app_var lp_volt_recon_method voltage_automatic
```

Or

```
set_design_attributes -elements { . } -attribute upf_reconciliation_automatic true
```

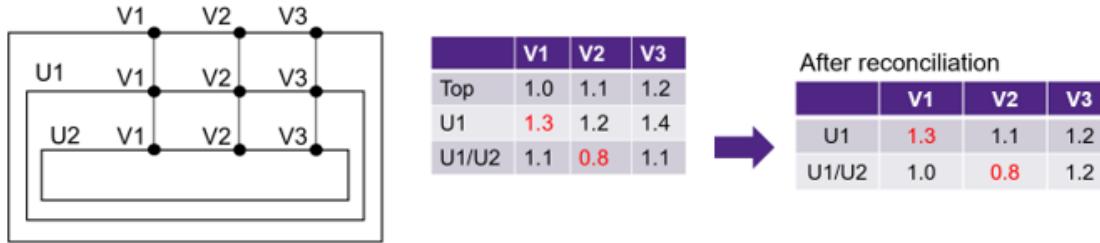
In addition, VC LP allows voltage + off state reconciliation across all scopes by setting the `lp_volt_recon_method` application to `all_automatic`.

```
set_app_var lp_volt_recon_method all_automatic
```

### 5.1.23.3 Examples

#### 5.1.23.3.1 Example 1

```
set_app_var lp_volt_recon_method voltage
set_variation -tolerance 0.2
set_design_attributes -models {u1 u2} -attribute upf_reconcile_boundary
"reconcile_voltages"
```



In this case, U1/V1 is outside the threshold (-0.3V difference when moving from block to top) and U1/U2/V2 is outside the threshold (+0.3V when moving from block to top).

In the final result, U1/U2/V1 will be reconciled to 1.0 but U1/V1 will be kept at 1.3, resulting in all port states of V1 canceling out.

Similarly, U1/V2 will be reconciled to 1.1 but U1/U2/V2 will be kept at 0.8.

For V3, both U1/V3 and U1/U2/V3 will be reconciled to 1.2 with no conflict.

#### 5.1.23.4 Example 2

##### Setup

```
set_app_var lp_volt_recon_method voltage
set_variation -supply v1 -tolerance 0.2
set_variation -supply v2 -tolerance 0.2
set_design_attributes -models {U1} -attribute upf_reconcile_boundary
"reconcile_voltages"
```



Let's take block/V1 (1.3v) voltage. Both top level V1 values (1.1 and 1.2) are inside the threshold. Therefore block/V1 is substituted by both 1.1v and 1.2v

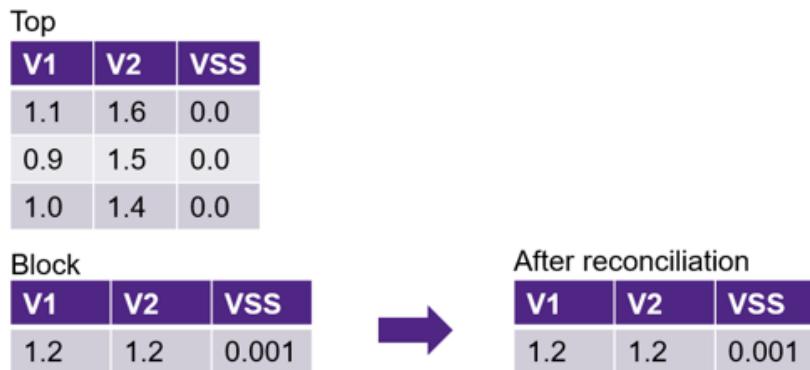
Same as that all the voltages are substituted (shown in the diagram)

Expanded block PST will have four different states.

### 5.1.23.5 Example 3

#### Setup

```
set_app_var lp_volt_recon_method voltage
set_variation -supply v1 -tolerance 0.2
set_variation -supply v2 -tolerance 0.2
set_design_attributes -models {U1} -attribute upf_reconcile_boundary
reconcile_voltages"
```

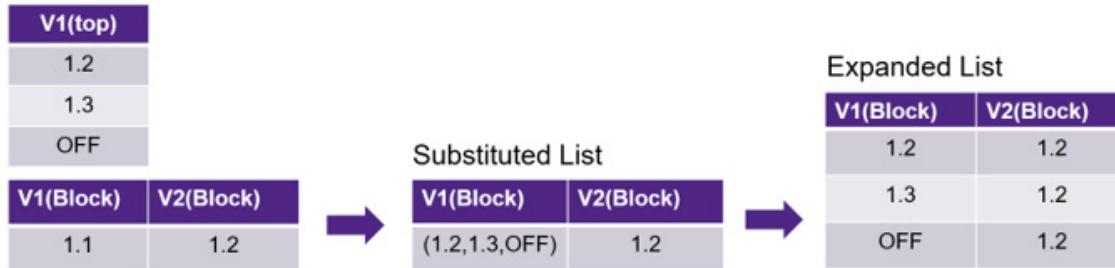


Let's take block/V1 (1.2v) voltage. Both top level V1 values (1.1 and 1.0) are inside the threshold. But top level 0.9v is outside the threshold. Therefore top V1 is not propagated to block. Similarly, V2 is not propagated to block too. Since threshold value is set for the vss supply (default is taken as 0), the top vss does not propagate to the block.

### 5.1.23.6 Handling isolation

- ❖ VCLP provides an option which inserts the off state during reconciliation. When the option is given, off is treated the same as any other voltage value

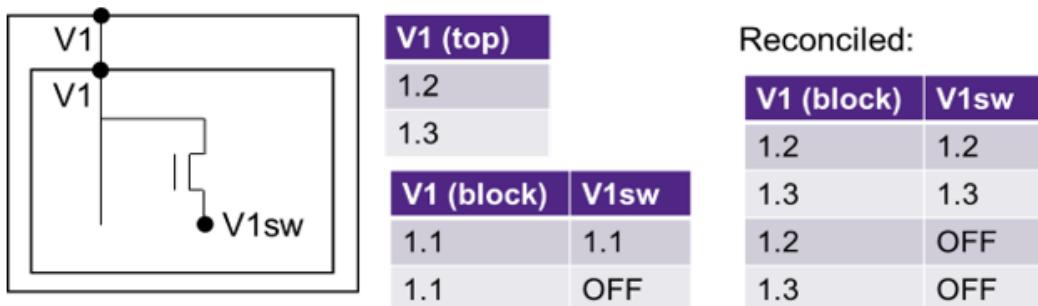
- ❖ To enable voltage reconciliation with both voltages and isolation, set  
set\_app\_var lp\_volt\_recon\_method all
- ❖ When this value is used, the PST\_VOLTAGE\_CHANGED message will include "off" among the voltages.



### 5.1.23.7 Handling Power Switches

- ❖ All the supplies in one passively derived" family are reconciled to the same voltages.
- ❖ A supply V1sw is considered passively derived from another supply V1 if
  - a. There is a power switch strategy with V1 as the input supply and V1sw as the output supply
  - b. Every unique port state voltage value except off which appears in V1sw, also appears in V1.

```
add_port_state V1 -state {IH 1.2}
add_port_state V1sw -state {OH 1.2} -state {OFF off}
create_power_switch ... -input_supply_port {VDD V1} \
    -output_supply_port {VSW V1sw}
```



### 5.1.23.8 New tags Introduced

Two new UPF stage warnings are introduced for this support:

- ❖ PST\_VOLTAGE\_CHANGED: Each supply where reconciliation has taken place will have one message of PST\_VOLTAGE\_CHANGED.

Example:

```
Tag : PST_VOLTAGE_CHANGED
      Description : Local voltage values of supply [Supplies] in block changed by
      voltage reconciliation
      Violation   : LP:6
      Supplies
      UPFSupply  : A2
      NewVoltageList
```

```

    Voltage          : A2_L (1.1)
    Voltage          : A2_H (1.2)
OldVoltageList
    Voltage          : C2_L (1.2)
    Voltage          : C2_H (1.3)
PresentInPSTs
    PowerState
        PowerStateTable : ua/tb
        LocalSupply     : B2
    PowerState
        PowerStateTable : ua/ub/tb
        LocalSupply     : C2

```

- ❖ **PST\_VOLTAGE\_UNCHANGED:** PST\_VOLTAGE\_UNCHANGED is reported for the supplies where reconciliation was not able to change the voltage, because the new voltages would be outside the legal range for the supply.

#### Example

```

Tag          : PST_VOLTAGE_UNCHANGED
Description   : Local voltage value of supply [Supplies] in block not changed
by voltage reconciliation due to threshold
    Violation      : LP:1
    Supplies
        UPFSupply   : A1
    NewVoltage      : A1_L (1.1)
    OldVoltage      : B1_L (1.5)
    VoltageThreshold : 0.3
PresentInPSTs
    PowerState
        PowerStateTable : ua/tb
        LocalSupply     : B1
    PowerState
        PowerStateTable : ua/ub/tb
        LocalSupply     : C1

```

#### 5.1.23.9 Parser Message Introduced

The following parser messages is introduced under the `set_variation` command support.

- ❖ *[Error] UPF\_VOLTAGE\_VALUE\_OUTOFRANGE is reported when user defines threshold value which is not between -999 and 999*

#### Example

```

set_variation -supply {v1} -tolerance {1000}
[Error] UPF_VOLTAGE_VALUE_OUTOFRANGE: Voltage value out of range
Voltage value '1000' is out of the range, tool support voltage value in the range of '-999' to '999'.
Please specify a value which is greater than -999 and less than 999.

```

- ❖ *[Warning] UPF\_THRESHOLD\_CONFLICT is reported when user defines conflicts threshold values for same supply.*

#### Example

```

set_variation -supply {v1} -tolerance {0.5}
set_variation -supply {v1} -tolerance {0.4 0.5}

```

[Warning] UPF\_THRESHOLD\_CONFLICT: Definition of supply threshold conflicts  
 The supply threshold of supply net 'v1' is set to '{0.5 0.5}' before, but set to '{0.4 0.5}' now.  
 Please do not set different supply threshold for a supply net

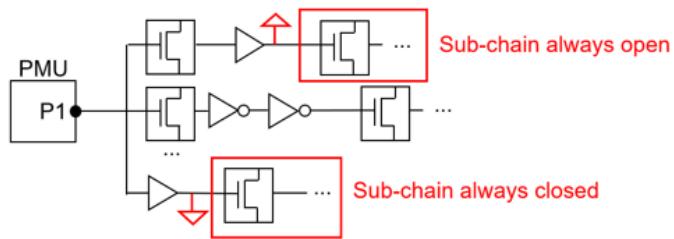
### 5.1.23.10 Limitations

- ❖ Unique port state names required.
- ❖ Voltage reconciliation support is not available for `add_power_state -supply/-group`

### 5.1.24 Support for Adding Test Circuits in PSW Chains

PSW sub-chains can be stuck at 0 or 1 faults. This may lead to partial on or partial off. As a solution, you can insert test circuit to ensure that all sub-chains are not stuck as 0 or 1 by inserting "wide AND" and "wide OR" gates to the ends of the chains.

**Figure 5-50 Sub Chains in PSW Circuits**

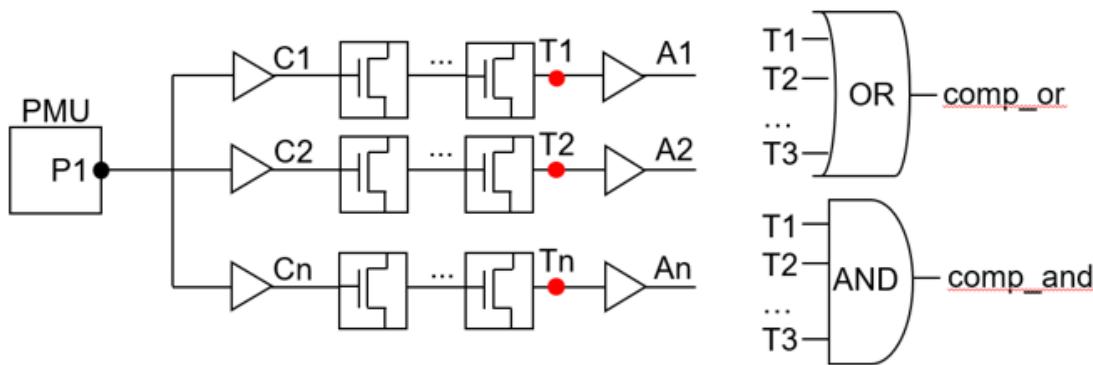


VC LP has introduced checks to identify issues in test circuits (AND and OR gates placement and power states are correct). These checks do not use PSW strategy information. You should provide the PSW chain information using the following command.

#### Syntax

```
lp_psw_andor      # check and/or logic on PSW outputs
    -input <net>          (Trace from this input control net)
    [-and <net>]           (Expected AND output)
    [-or <net>]            (Expected OR output)
    [-fringe <depth>]       (Allowed fringe depth)
    [-prequal <ports>]     (Prequalified primary input list)
    [-verbose]              (Print derived test nodes)
```

Consider the following example:

**Figure 5-51 PSW Chain Example**

The syntax description for the example in [Figure 5-51](#) is as follows:

- ❖ -input: PMU/P1 (Chain Starting Node)
- ❖ -and: comp\_and (AND logic output wire)
- ❖ -or: comp\_or (OR logic output wire)
- ❖ -verbose: Print additional information after command execution.
- ❖ -fringe: Described in section "[Fringe PSW instances](#)".
- ❖ -prequal: Described in section "["Prequalified Primary Inputs"](#)"

You cannot use both -and and -or options together in a single command. You should write separate commands for each logic outputs.

### Example

```
lp_psw_andor -and -input PMU/P1
lp_psw_andor -or -input PMU/P1
```

#### 5.1.24.1 report\_pst\_andor

This command reports all PSW instances which were not traveled by any lp\_psw\_andor command.

#### 5.1.24.2 Tags Introduced for this Support

The following checks introduced to identify issues in the test circuit.

If all endpoints (DesignNets) (T1, T2, T3, ...) of each PSW chain are not connected to the output wire mentioned in the lp\_psw\_andor command, or if the logic between endpoints and output logic wire is not matched with given logic in the lp\_psw\_andor command, then this violation is reported for each individual lp\_psw\_andor command which fails.

```
Tag : PSW_ANDOR_EXPR
Description : Power switch and/or chain from [DriverNode] to [ReceiverNode]
              does not have required logic function
DriverNode : i
ReceiverNode : oo
DesignNets
  DesignNet
    NetName : u7/i9/i9/i9/w8
    NetType : Design
  DesignNet ...
```

- ❖ PSW\_ANDOR\_STATE

If there exists a power state where the driver is on, and any of the set of gates is off, then this violation is reported once per instance in the tree with insufficient power.

```

Tag          : PSW_ANDOR_STATE
Description   : Power switch and/or chain instance [Instance] is off,
                when driver [DriverNode] is on
Instance      : ua/n1
DriverNode    : i
InstanceInfo
  PowerNet
    NetName  : vdda
DriverInfo
  PowerNet
    NetName  : vdd
ReceiverNode: ua/o
States
  State     : t/s2

```

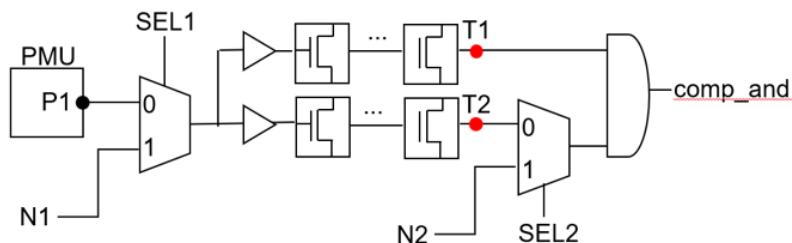
### 5.1.24.3 Special Scenarios

#### 5.1.24.3.1 Muxes inside test circuit (DFT Muxes)

Muxes are allowed inside test circuit if set\_case\_analysis command is given for the muxes.

##### Example

```
set_case_analysis 0 {SEL1 SEL2}
```



#### 5.1.24.3.2 Prequalified Primary Inputs

- ❖ There can be some design nets which connected to the test circuit inputs that are not the endpoints of the PSW chain starting from given staring point by user.
- ❖ But these nets can be connected intentionally by the user because there are pre-qualified in another chain.
- ❖ Provide these pre-qualified inputs as below

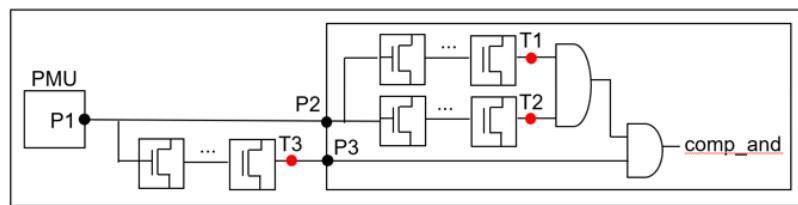
```

lp_psw_andor
-and
-input PMU/P1
-prequal {list_of_prequalified_design_nets (separate by space) }

```

##### Example

- ❖ Consider the following design.



- ❖ If the check is staring from P2 (Inner hierarchy) then P3 will not be the part of the PSW chain but it is connected to the test circuit.
- ❖ But when check stars from P1, then check will be a pass.
- ❖ So when the check starts from P2, P3 should be prequalified net.
- ❖ Therefore, if you provide P3 in the prequel list, then the test will be pass and no violations reported.

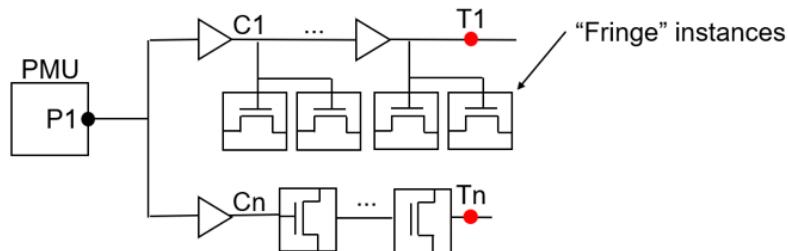
#### 5.1.24.3.3 Fringe PSW instances

There can be fringe PSW instance in the PSW chain (example, fishbone chain, a chain with PSWs which does not have acknowledge pin). To support this kind of chain, you should provide the following option with the `lp_psw_andor` command.

```
lp_psw_andor -fringe 1
```

The default value of `-fringe` is 0.

Currently, VC LP supports one level of fringe instance as shown in the following figure:



#### 5.1.24.4 Limitation

This check is not supported for fine-grain power switches.

### 5.1.25 Support for RAM Enabled Connectivity Checks

RAM controls like `sleep_en`, `ret_en`, and so on should be properly connected in the design to make designs electrically and functionally correct. UPF does not provide a way to define RAM controls. As per today, users rely on the simulation to verify these connections. With this new support, RAM controls can be specified using Tcl commands. VC LP introduces checks to control connectivity and reachability.

#### Use Model

RAM control signal connectivity can be specified for RAM cells where `is_memory_cell: true` attribute is applied, using the `define_ram_controls` Tcl command. This command should be added after `read_upf` stage and before executing any `check_lp` command.



#### Note

This feature is under VT-VC-BETA license. VT-VC-BETA checks out at `define_ram_controls`.

## Syntax

```
vc_static_shell> define_ram_controls -domain domain name [-elements cell list] -ctrl
{{pin {signal_list}* } [-ctrl_supplies {{pin supply_name}* }}] [-quiet] RAM control
strategy name
```

Where:

- ❖ -domain [domain\_of\_RAM\_cells]: Specify the domain name for which the ram control applies. The elements' power domain check are performed only if domain is specified. At least one of -domain and -elements should be given, otherwise the command will be ignored with message LP\_CMD\_IGNORED.
- ❖ -ctrl {{RAM\_CELL\_PIN connected\_signal\_name }}: The pins of RAM cells for which the connectivity should be checked with the respect to specified valid signals. Inverting signal for CRL pins allowed (for example, -ctrl { {CEN ~cen} } ).
- ❖ [-elements] {RAM\_CELL\* }: RAM cell instance names that are to be checked. Wild card support is available.
- ❖ [-ctrl\_supplies] {{RAM\_CELL\_PIN supply\_vdd}}: UPF supply to override the supply of RAM control pins.\*

\*Options within [] are not mandatory

## Example

```
define_ram_controls RAM_PD1 -domain PD1 \
-elements {RAM1 RAM2 RAM3*} \
-ctrl {{SLEEP_EN ram_sleep_en} {RET_EN {ram_ret_en_0 ram_ret_en_1}} } \
-ctrl_supplies {{SLEEP_EN VDD_1} {RET_EN VDD_2}}
```

### 5.1.25.1 New tags Introduced

The following violations are introduced for RAM enabled connectivity checks.

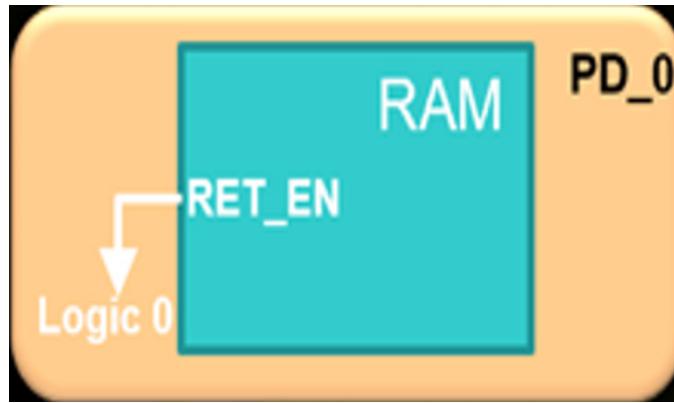
- ❖ RAM\_CTRL\_CONN

This violation is reported if the RAM control signal does not match RAM instance pin connection. This tag is enabled by default.

## Example

RAM\_CTRL\_CONN violation is reported for following incorrect signal match of pin

```
define_ram_controls RAM_PD1 -domain PD_0 \
-elements {RAM*} \
-ctrl {RET_EN 1'b1 }
```




---

```
RAM_CTRL_CONN (1 error/0 waived)
-----
Tag          : RAM_CTRL_CONN
Description   : RAM control signal does not match RAM instance [Instance]
connection pin [CellPin]
Violation    : LP:1
Instance     : u_SRAM_0
CellPin      : CEN
SignalDesignList
  SignalName  : cen
```

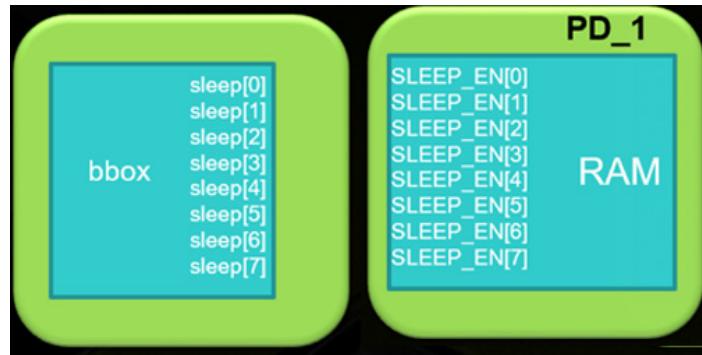
#### ❖ RAM\_CTRL\_UNCONN

This violation is reported if the RAM instance control pin is unconnected. This tag is enabled by default.

#### Example

For the following unconnected pin, this violation will be flagged.

```
define_ram_controls RAM_PD1 -domain PD_1\
    -ctrl {SLEEP_EN bbox/sleep }
```



#### ❖ RAM\_CTRL\_COMBO:

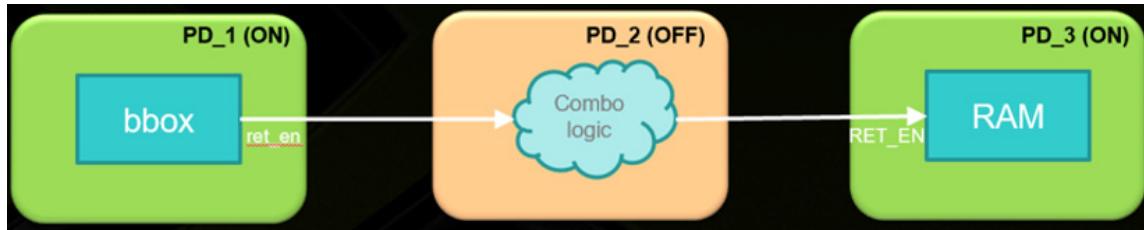
This violation is reported if the RAM control signal passes through non-buffer logic.

This tag is enabled by default.

#### Example

For following connection with combo logic, this violation will be flagged.

```
define_ram_controls RAM_PD1 -domain PD_3 \
    -elements {RAM*} \
    -ctrl {RET_EN bbox/ret_en }
```



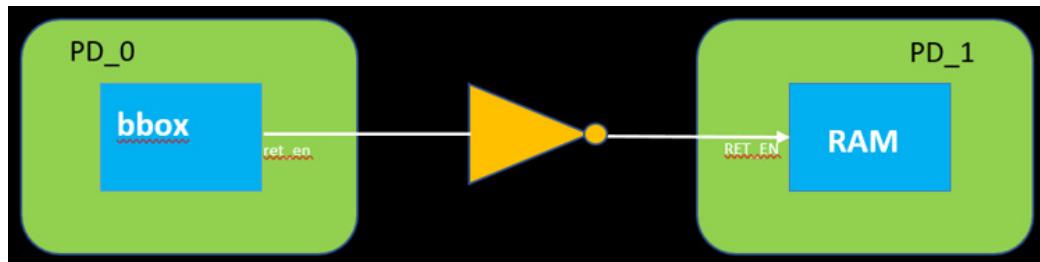
- ❖ **RAM\_CTRL\_INVERT:**

This violation is reported if the RAM control signal has opposite polarity from RAM instance pin, odd number of inverters exist in the path. This tag is enabled by default.

#### **Example**

In the following example, the inverted output is connected to Ram cell pin

```
define_ram_controls RAM_PD1 -domain PD_1 \
    -elements {RAM*} \
    -ctrl {RET_EN bbox/ret_en }
```

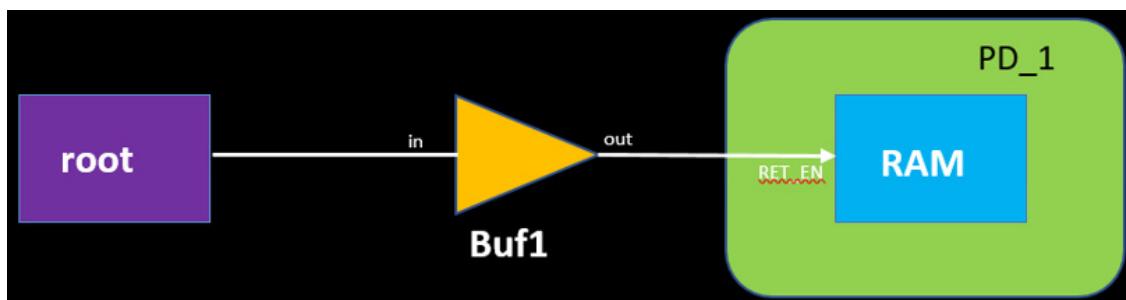


- ❖ **RAM\_CTRL\_RESET**

This violation is reported if the driver of RAM Instance pin <CellPin> has no reset. This tag is disabled by default.

#### **Example**

In following figure, if the root has no reset pin, this violation is reported.



- ❖ **RAM\_CTRL\_GLITCH**

This violation is reported if the RAM instance pin <CellPin> is not driven by a state signal, output of a flip-flop, latch or a top-level port. This tag is disabled by default.

### Example

In the above scenario if the root driver is not a state signal, output of a flip-flop, latch or a top-level port this violation flags.

#### ❖ RAM\_CTRL\_STATE (error)

This violation is reported for an instance where a RAM instance pin (CellPin) supply ON, but its driver signal (DesignSignal) supply is OFF. That is, when the -ctrl\_supplies option is specified in the `define_ram_controls` command with the RAM control pin supply ON and ctrl signal driver's supply OFF.

This tag is disabled by default. To enable it, use `configure_tag -app LP RAM_CTRL_STATE -enable` command.

### Example

```
define_ram_controls ctrl1 -element {u_SRAM_1} -domain PD_top -ctrl {{CEN cen}} -ctrl_supplies {{CEN Vdd_1} {CEN VSS}}
```

The following is an example snippet of the violation report:

---

```
-----  
RAM_CTRL_STATE (1 error/0 waived)  
-----  
Tag : RAM_CTRL_STATE  
Description : RAM instance [Instance] pin [CellPin] supply ON but its driver signal [DesignSignal] supply is OFF  
Violation : LP:1  
Instance : u_SRAM_1  
CellPin : CEN  
DesignSignal  
SignalName : cen  
ControlInfo  
PowerNet  
NetName : Vdd_1  
NetType : UPF  
PowerMethod : FROM_RAM_CTRL  
GroundNet  
NetName : VSS  
NetType : Design/UPF  
GroundMethod : FROM_PG_NETLIST  
DriverInfo  
PowerNet  
NetName : Vdd_top  
NetType : Design/UPF  
PowerMethod : FROM_UPF_POWER_DOMAIN  
GroundNet  
NetName : VSS  
NetType : Design/UPF  
GroundMethod : FROM_UPF_POWER_DOMAIN  
States  
State : top_pst/ALL_OFF
```

---

#### ❖ RAM\_CTRL\_VOLT (error)

This violation is reported when a RAM instance pin (CellPin) supply voltage is different from the signal (DesignSignal) in the control signal path. This violation is disabled by default. To enable it, use `configure_tag -app LP RAM_CTRL_VOLT -enable` command.

### Example

```
define_ram_controls ctrl1 -element {u_SRAM_1} -domain PD_top -ctrl {{CEN cen}} -
ctrl_supplies {{CEN Vdd_1} {CEN VSS}}
```

The following is an example snippet of the violation report:

```
-----
RAM_CTRL_VOLT (1 error/0 waived)
-----
Tag : RAM_CTRL_VOLT
Description : RAM instance [Instance] pin [CellPin] supply voltage is different
from
    signal[DesignSignal] in control signal path
Violation : LP:2
Instance : u_SRAM_1
CellPin : CEN
DesignSignal
    SignalName : cen
ControlInfo
    PowerNet
        NetName : Vdd_1
        NetType : UPF
    PowerMethod: FROM_RAM_CTRL
    GroundNet
        NetName : VSS
        NetType : Design/UPF
    GroundMethod : FROM_PG_NETLIST
DriverInfo
    PowerNet
        NetName : Vdd_top
        NetType : Design/UPF
    PowerMethod : FROM_UPF_POWER_DOMAIN
    GroundNet
        NetName : VSS
        NetType : Design/UPF
    GroundMethod : FROM_UPF_POWER_DOMAIN
States
    State : top_pst/TOP_ON
```

#### 5.1.25.2 Parser Messages Introduced

The following parser warnings and errors are introduced for the `define_ram_controls` TCL command support.

- ❖ **RAM\_ELEMENT\_IGNORED** [Warning]

If the cells specified in `-elements` does not exist in the specified domain, and the cells specified in `-elements` exist in the specified domain, but does not have `is_memory_cell: true` attribute, the `RAM_ELEMENT_IGNORED` warning message is reported.

- ❖ **RAM\_PIN\_IGNORED** [Warning]

For ram cells, invalid pin such as output pin, internal pin is not found in the `-ctrl` option, `RAM_PIN_IGNORED` warning message is reported.

- ❖ **RAM\_PIN\_OVERRIDE** [Warning]

RAM\_PIN\_OVERRIDE is reported if it is used in different `define_ram_controls` command for same ctrl pin of same RAM, and it will override the previous one if the latter one is valid.

- ❖ **RAM\_CTRL\_IGNORED [Warning]**

`RAM_CTRL_IGNORED` is reported if this signal specified in `signal_name_list` in the `-ctrl` option does not exist.

- ❖ **UPF\_SUPPLY\_SAME [Warning]**

`UPF_SUPPLY_SAME` is reported in case supply mentioned in `-ctrl_supplies` is found to be same as Pin supply.

- ❖ **RAM\_SUPPLY\_OVERRIDE [Warning]**

`RAM_SUPPLY_OVERRIDE` is reported if the supply of the pin is defined in different `define_ram_controls -ctrl_supplies`, and it will override the previous one if the latter one is valid.

- ❖ **RAM\_CTRL\_NAME\_DEFINED [Warning]**

`RAM_CTRL_NAME_DEFINED` is reported if the same `ram_control_name` is used in different `define_ram_controls`.

- ❖ **SIGNALS\_NOT\_SAME\_WIDTH [Error]**

`SIGNALS_NOT_SAME_WIDTH` is reported for bus width mismatch between pins and respected signals in the `-ctrl` option.

- ❖ **POWER\_DOMAIN\_NF [Error]**

`POWER_DOMAIN_NF` is reported in case UPF supply mentioned in `power_domain` is not found in the design in the `-domain` option.

- ❖ **SUPPLY\_NET\_NP [Warning]**

`SUPPLY_NET_NP` is reported in case UPF supply mentioned in the `-ctrl_supplies` is not found in the design.

### 5.1.25.3 Control supply option

From the `-ctrl_supplies` option, the supply of RAM control pins is overridden, and consider when doing the checks. A new supply method `FROM_RAM_CTRL` is introduced.

## 5.1.26 Support for Synopsys Safety Format (SSF) Checks in VC LP

Automotive chip designing is getting more advanced and complex. Therefore, the chances for failures are high. To avoid such failures, the key safety mechanisms listed are as follows:

- ❖ Double Modular Redundancy (DMR)
- ❖ Triple Modular Redundancy (TMR)
- ❖ Safe Finite State Machine (FSM)

All the Synopsys tools have safety specification conceptualized as Synopsys Safety Format (SSF).

In VC LP flow, the UPF is read on the design with safety mechanisms already applied through SSF. However, the UPF file is written without the knowledge of Safety Intent. Therefore, VC LP reads the UPF for a design with Safety mechanisms (DMR/TMR/FSM) applied.

In the VC LP setup, SSF must be read before `upf_read` with the `read_ssf` command.

### Syntax

```
read_ssfsf <ssf file name>
```

### Example

```
read_ssfsf example.ssf
```



### Note

If SSF is read after UPF is read, VC LP reports an error message and does not load SSF.

### Safety Practices

- ❖ Safety registers should not be used as retention register because registers with no clock can flip value when in retention state.
- ❖ Similarly, for safety related registers, saving power by clock-gating exposes those registers to longer time with no clock, increases chance of flipping value because of SEU (Single Event Upset).

#### 5.1.26.1 News Checks Introduced

In figure 1, assume FF1 register need safety application. It can be achieved with the addition of redundant registers along with FF1 as shown in Figure 2 (TMR mechanism).

In VC LP flow, as the tool is not aware of the redundant registers, these can be used in commands like `set_retention`, `set_retention_elements`, `set_design_attributes`, `set_isolation`, `create_power_switch`, and so on, which is incorrect and should be avoided.

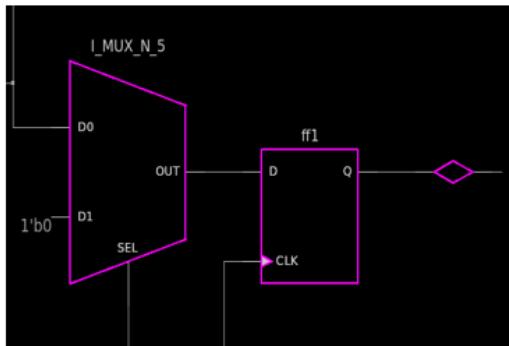


Figure 1

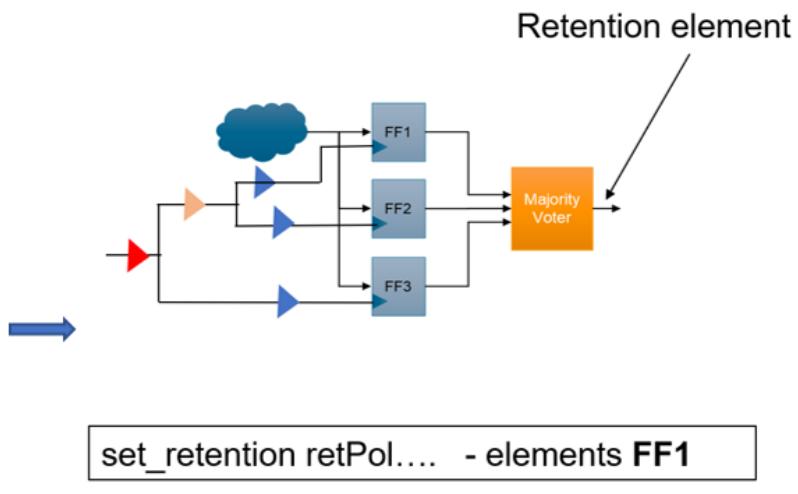


Figure 2

Therefore, the following new checks were introduced in VC LP to report such incorrect usages of redundant registers in retention command.

- ❖ RET\_ELEMENT\_SAFETY: This violation is reported at the UPF stage when any register which is referred in UPF retention strategy and specified as safety register.
- ❖ RET\_ELEMENT\_FAILSAFE: This violation is reported at the UPF stage when any register referred in UPF retention strategy, and as part of fail-safe FSM.

These tags are disabled by default. To enable it, use the `configure_lp_tag -enable` command.

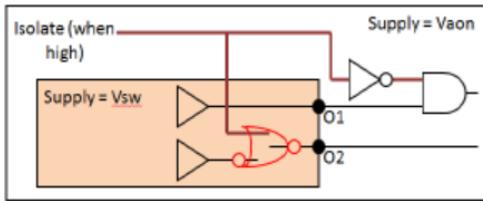
## 5.2 Advanced Low Power Cells

### 5.2.1 NOR-style Isolation Cells

The NOR-style isolation strategy can clamp to zero even if it is placed in the switched off domain, where the cell's power is off and ground is on. The isolation gate has a solid connection to ground when the power is off, such that it can clamp to zero correctly, even if power is off and ground is on. [Figure 5-52](#) shows an example of a NOR-style isolation gate.

In [Figure 5-52](#), the two outputs at the right are correctly isolated. The gate in red is the NOR-style isolation gate. Even though it is placed in the switched domain, it clamps to zero. The internal transistor layout ensures a solid connection to ground when the isolate signal is high, regardless of any unknown value on the power connection. This means that the outputs can be isolated by single rail gates in the switched domain, which removes the need to route the always-on supply for Isolation cell.

**Figure 5-52 Example for NOR-style Isolation Gate**



#### 5.2.1.1 Prerequisites for a NOR-style Isolation Gate

A liberty cell is a NOR-style isolation gate, if:

- ❖ The cell has liberty attribute `is_isolation : true`
- ❖ The output matches function `AND_NOT` " ( $y=a \& !b$ )
- ❖ The PG ports has liberty attribute `permit_power_down`
- ❖ The output pin has liberty attribute `alive_during_partial_power_down`

#### 5.2.1.2 Disabling NOR-style Isolation Checks

By default, the NOR-style isolation checks are performed.

To disable the NOR-style isolation checks, set the `enable_iso_nor` application variable to `false`.

```
%vc_static_shell>set_app_var enable_iso_nor false
```

By default, this application variable is set to `true`.

VC LP performs NOR style isolation checks under the `enable_iso_nor` application variable. The `load_for_nor_iso_without_crossover` application variable decides whether or not to consider a sink as a valid load. The fix considers,

- ❖ Hierarchical boundaries with SPA Receiver/ SRSN/ REPEATER supplies as a valid load.
- ❖ PG pins, NOR style enable pins, internally isolated pins, analog pins, enable pins of single rail ISO/ELS cells are ignored as load pins.
- ❖ Hanging nets are ignored as load pins.
- ❖ Connectivity operators are ignored as load pins.

When the `load_for_nor_iso_without_crossover` application variable is set, VC LP gives a parser warning at `check_lp -stage design` stage for NOR style isolation cells without a valid load.

## Example

[Warning] NOR\_ISO\_UNLOADED: Nor iso cell is unconnected at the output  
Nor Iso cell u\_RD/u\_PRPL/NORISO2/ISO is unconnected at the output, no sink forwarding will be done  
Users are recommended to set the application variable "load\_for\_nor\_iso\_without\_crossover" to true if NOR style isolation cells are implemented in the design.

### 5.2.1.3 Defining NOR-style Isolation Strategies

You can define NOR-style isolation strategies using the following commands:

- ❖ map\_isolation\_cell
- ❖ set\_isolation\_supply\_set {}
- ❖ set\_design\_attributes

#### 5.2.1.3.1 Defining NOR-style Isolation Strategies using map\_isolation\_cell

You can use the map\_isolation\_cell command to define NOR-style isolation strategies:

```
set_isolation s1 -domain d1 ...
map_isolation_cells1 -domain d1 -cells {A B C D}
```

If one or more of the cells A B C D happen to be NOR-style isolation cells, then VC LP assumes that the strategy is a NOR-style isolation strategy.

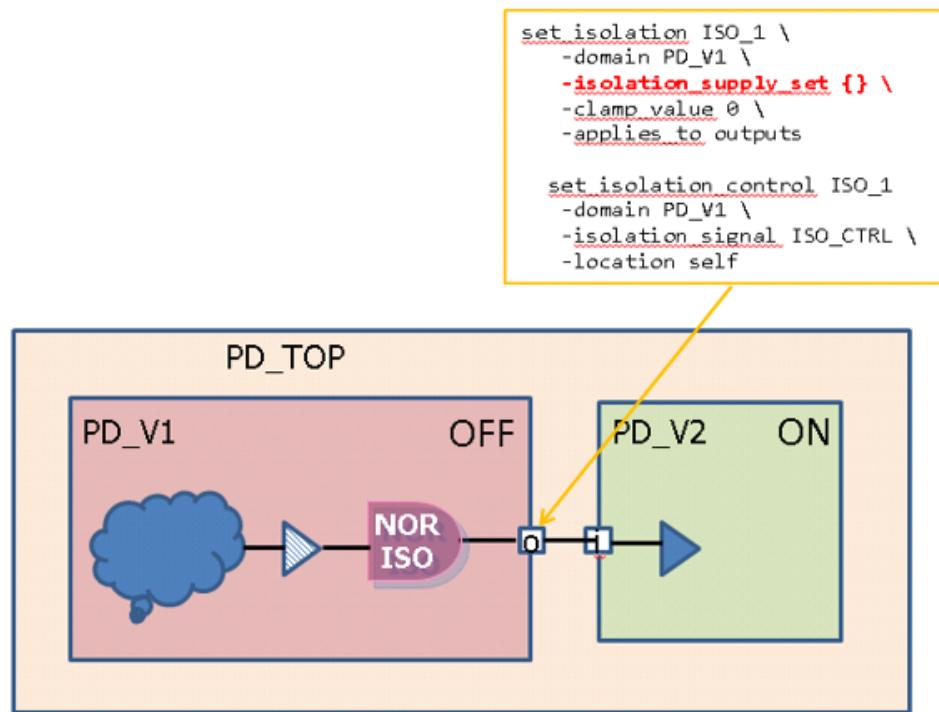
#### 5.2.1.3.2 Defining NOR-style Isolation Strategies using set\_isolation\_supply\_set {}

You can use the set\_isolation\_supply\_set {} command to define NOR-style isolation strategies:

```
set_isolation -domain <domain_name> - clamp_value 0 -applies_to outputs -
isolation_supply_set {}
```

When you define isolation strategies with -isolation\_supply\_set {}, by default, VC LP takes domain supply where the isolation cell is present. You need not map the isolation strategy with the NOR-style isolation cell.

[Figure 5-53](#) shows an example of defining isolation strategies with -isolation\_supply\_set {}.

**Figure 5-53 Example for Defining Isolation Strategies with -isolation\_supply\_set {}**

### 5.2.1.3.3 Defining NOR-style Isolation Strategies using set\_design\_attributes

You can define the NOR-style isolation strategy using the `set_design_attributes -attribute iso_nor` command.

#### Use Model

```
set_design_attributes -elements <scope_of_the_domain> -attribute iso_nor
{domain1.startegy1 domain2.startegy2}
```

#### Example

```
set_isolation NOR_ISO -domain PD_TOP -isolation_supply_set SS_VVDD_SOC -elements
{dout[0]} -clamp_value 0
set_isolation_control NOR_ISO -domain PD_TOP -isolation_signal isolate -
  isolation_sense high -location self
set_design_attributes -elements . -attribute iso_nor { PD_TOP.NOR_ISO }
```

When you use this feature, ensure that you adhere to the following rules:

- ❖ The `-elements` option is mandatory and you can specify only one scope in it.

Example:

```
set_design_attributes -elements { . m0 } -attribute iso_nor { PD_TOP.ISO_NOR }
set_design_attributes -attribute iso_nor { PD_TOP.ISO_NOR }
```

- ❖ You can specify more than one attribute value separated by space.

Example:

```
set_design_attributes -elements . -attribute conservative_diff_supply_only_isolation
true -attribute iso_nor { PD_TOP.ISO_NOR }
```

- ❖ You can specify more than one iso\_nor at the same scope.

Example:

```
set_design_attributes -elements . -attribute iso_nor { PD_TOP.ISO_NOR1 }
set_design_attributes -elements . -attribute iso_nor { PD_TOP.ISO_NOR2 }
```

- ❖ You cannot use wild card characters in the value and the -element option.

Example:

```
set_design_attributes -elements { . } -attribute iso_nor { PD_TOP.* }
```

### 5.2.1.4 VC LP Checks on NOR-style Isolation Strategies

For a NOR-style isolation strategy, if the isolation ground is set to Off and the isolation power is set to On, then the ISO\_STRATSUPPLY\_INCORRECT messages are reported. If the isolation power is set to Off and the ground is set to On, the ISO\_STRATSUPPLY\_INCORRECT messages are not reported.

VC LP also checks the NOR-style strategy power supply with the source supply, and if strategy power supply is on and the source supply is off, the ISO\_STRATEGY\_SOURCE violation is reported.

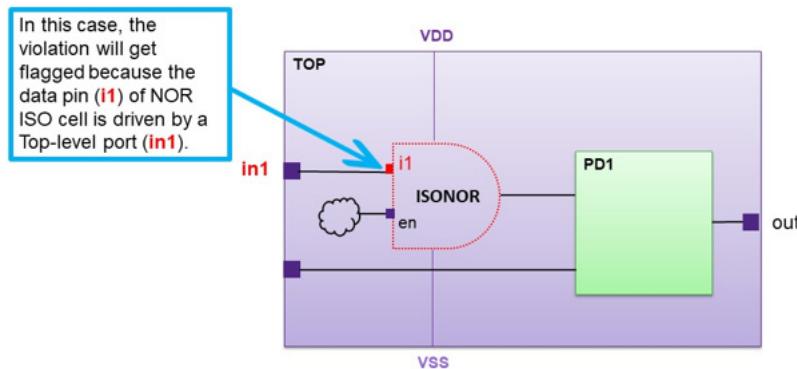
VC LP reports the ISO\_INST\_NOR and ISO\_STRATEGY\_NOR violations if a NOR style isolation cell or a strategy applied node is directly driven by an input pin of a top level port.

- ❖ ISO\_INST\_NOR

The ISO\_INST\_NOR violation is reported for all the NOR-ISO instances whose data pin is directly driven by a top-level port.

The ISO\_INST\_NOR violation has severity 'error' and is reported at the 'design stage'.

To check the driver of data pin of the NOR-isolation cell, VC LP traces the driver of data pin. While tracing, the tool skips only the module boundary ports and the connectivity operators.



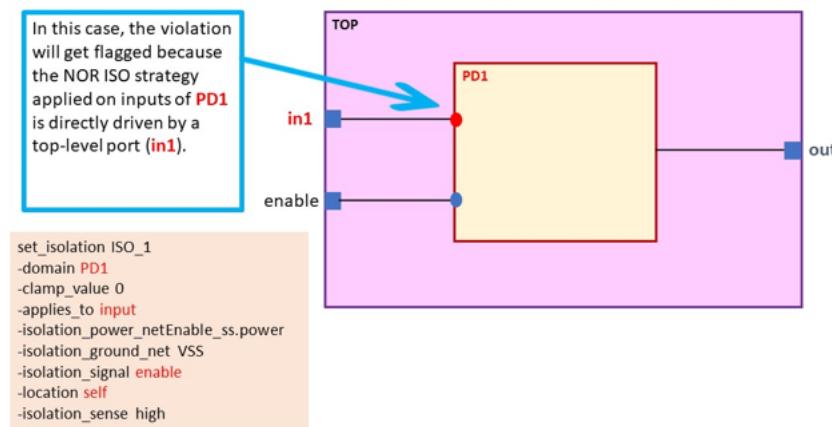
- ❖ ISO\_STRATEGY\_NOR

Previously, if any NOR isolation strategy applied on any domain were driven by the input pin of a top level port, there were no errors.

Starting with this release, VC LP reports the ISO\_STRATEGY\_NOR violation when a NOR isolation strategy applied on a domain is driven by the input pin of a top level port.

The ISO\_STRATEGY\_NOR violation has severity error and is reported at the 'UPF stage'.

## Example



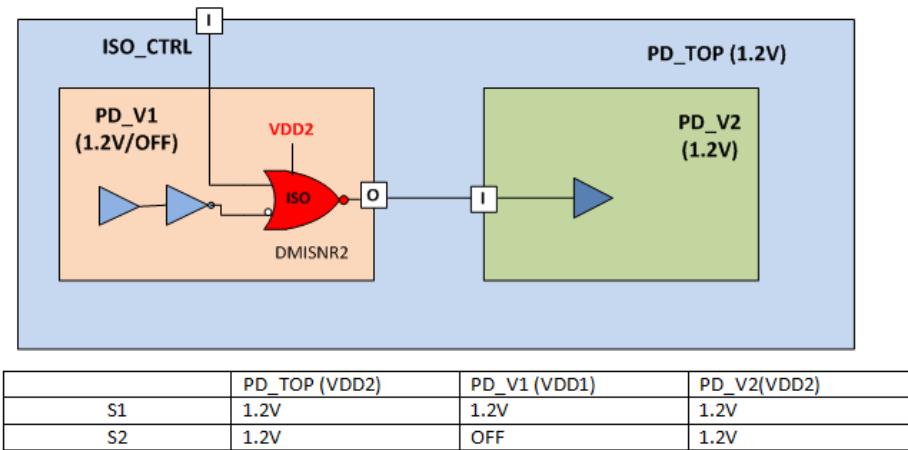
### 5.2.1.5 VC LP Checks on NOR-style Isolation Gates

As the NOR-style Isolation cell is able to clamp 0 to sink, even if the power of an Isolation cell is off, VC LP does not report any incorrect power rail related violation for this type Isolation cell.

If the NOR isolation cell ground is off and isolation cell power is on, then the ISO\_OUTPUT\_STATE/ISO\_SINK\_STATE messages are reported. If the NOR isolation cell power is off and ground is on, the ISO\_OUTPUT\_STATE/ISO\_SINK\_STATE messages are not reported.

When an ISO NOR cell is found in the design with isolation supply being ON while the data pin is driven by logic that is OFF, VC LP reports the ISO\_INST\_SOURCE violation.

**Figure 5-54 Example for ISO\_INST\_SOURCE Violation**



Previously, the PG\_STRATEGY\_CONN is reported when the design instance supply net does not match with supply net defined in UPF strategy. An exception is added while reporting the PG\_STRATEGY\_CONN violation for the NOR Isolation cell. For NOR-style isolation cells, this violation is reported only for the ground pin. This is because NOR isolation is always placed in the switched domain and it takes supply of the switched domain. For NOR-style isolation cells, the PG\_STRATEGY\_CONN violation is reported only when the design instance's ground supply net does not match with ground net defined in UPF strategy.

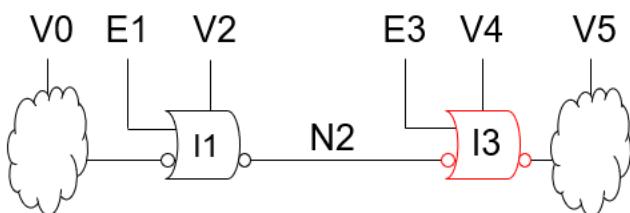
Consider the following example, where you specify the always on power, but DC performs an optimization to use NOR-style instead.

```
set_isolation ... -isolation_power_net VDD_always_on
```

VC LP checks that the NOR style isolation is electrically correct, even though the connected power net does not match with what is specified in the UPF. Therefore, the PG\_STRATEGY\_CONN is not reported for power pin of NOR isolation even though the instance supply net does not match with supply net defined in UPF strategy.

The ISO\_STRATEGY\_SOURCE/ISO\_INST\_SOURCE is reported when a nor-style isolation gate (I3) is on, but its immediate driver (V2) is off. This is a false violation if I1 is a nor-style gate which is clamping.

However, these checks are enhanced such that if lp\_signal\_supply information is available, it will be used. This may result in removing the violation, or reporting it on a different state.



```

lpss -signal E1 -supply V1
lpss -signal E3 -supply V3
(V1 and V3 are not needed in the diagram)

```

### 5.2.1.6 Limitations

- ❖ Parking checks are not supported.

### 5.2.1.7 Changes in Driver Analysis

The ISO\_CONTROL\_STATE check traces back from the isolation control signal given in isolation strategy after skipping buffer, inverter, and protection cell, and stops at first combo cell or top port to identify them as driver.

When the ISO\_CONTROL\_STATE check is run at netlist stage, there may be some scenario where buffer or inverter in the path may be off with respect to isolation strategy supply. Therefore, the driver tracing is changed to stop at first physical device (buffer / inverter/ protection device/combo/seq), and the first physical cell is identified as the driver.

To enable the change in the driver identification, set the following application variable to true:

```
set_app_var lp_mark_first_driver_control_check true
```

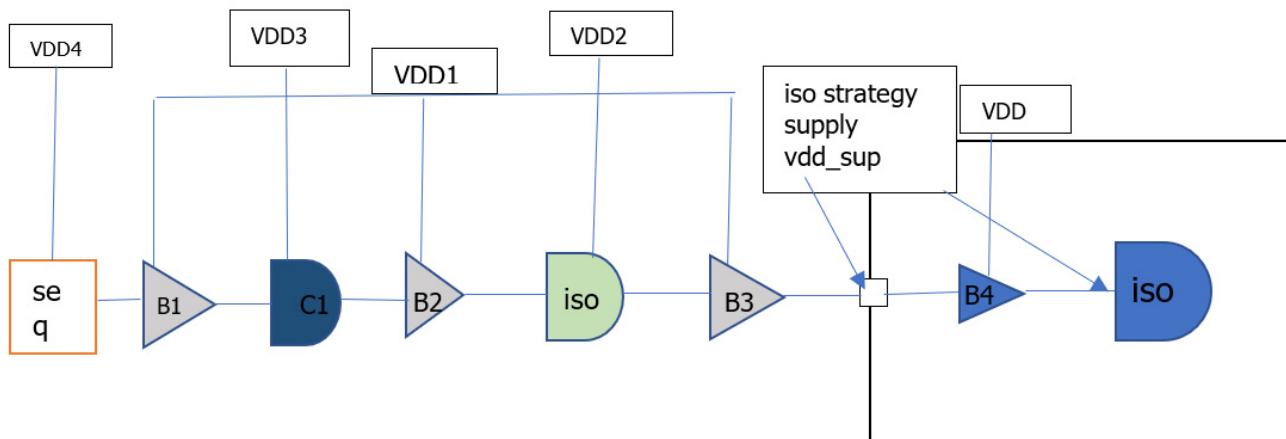
The change of the driver identification is applicable for the following tags:

- ❖ ISO\_CONTROL\_STATE
- ❖ ISO\_ASYNC\_STATE
- ❖ RET\_CONTROL\_STATE
- ❖ PSW\_CONTROL\_STATES

In case of NOR ISO strategy, identified driver is checked against the forwarded sink of iso strategy. While identifying the forwarded sink, tracing stops at the first device after iso strategy node, and makes the device as forwarded sink.

### Example

Consider the following design scenario:



- ❖ Assume VDD1 is less ON than vdd\_sub and VDD3 is less ON than vdd\_sub  
Previously, C1 is identified as the driver. Starting with this release, when lp\_mark\_first\_driver\_control\_check is set to true, B3 is identified as driver.
  - ❖ Assume VDD1 is less ON than vdd\_sub and VDD3 is not less ON than vdd\_sub  
Previously, C1 is identified as the driver. Starting with this release, when lp\_mark\_first\_driver\_control\_check is set to true, the ISO\_CONTROL\_STATE violation is not reported.

### **5.2.1.8 Aggressive Way of Identifying NOR ISO Strategy**

VC LP identifies a given isolation strategy as NOR iso strategy:

- ❖ If isolation supply set is not given in the strategy and clamp value of the strategy is zero
  - ❖ If clamp value of the strategy is zero and any one of the cell given in map\_isolation\_cell command is NOR ISO cell (alive\_during\_power\_down attribute)

When the `lp_ignore_map_command_nor_iso` application variable is set to true, VC LP changes the algorithm to identify NOR ISO strategy. By default, the `lp_ignore_map_command_nor_iso` application variable is set to false.

When the `lp_ignore_map_command_nor_iso` is set to `true`, an isolation strategy is identified as NOR ISO strategy if:

- ❖ Clamp value of the strategy is zero  
AND
  - ❖ Isolation supply set is not given  
AND
  - ❖ At least one cell given in map\_isolation\_cell command is NOR ISO cell (alive\_during\_power\_down attribute)

### **5.2.1.9 New Debug fields for ISO\_CONTROL\_STATE for NOR ISO case:**

In case of NOR ISO strategy, VC LP checks the driver supply of isolation control signal against the actual sink supply of NOR ISO strategy, instead of the strategy supply. The actual sink of the NOR ISO strategy is

identified as per the crossing. The following optional debug fields are reported if the ISO\_CONTROL\_STATE violation is reported for NOR ISO strategy:

- ❖ *IsNorIso*: Will be set as true for nor iso strategy
- ❖ *Logic Sink*: Actual sink whose supply has been compared against driver supply
- ❖ *SinkInfo*: Supply Info of the actual Sink.

## 5.2.2 NOR-style ELS Cells

VC LP supports NOR-style ELS cells for both clamp 0 and clamp 1. A NOR-style ELS cell is a single cell containing NOR isolation gate with a level shifter.

### 5.2.2.1 Prerequisites for a NOR-style ELS Cell

A liberty cell is an ELS cell, if the cell contains the `alive_during_power_up` attribute at the pin level.

- ❖ For clamp 1 cell:
  - ◆ The data pin has the `alive_during_power_up` attribute is set to false.
  - ◆ The cell has liberty attribute `is_isolation : true`, and `is_level_shifter : true`
  - ◆ The output matches function  $(y=I \mid ISO)$ .

The following is an example library snippet of a NOR-style ELS cell for clamp 1:

```
cell (ISOLS1_NORM) {
  pin(I) {
    direction : input;
    isolation_cell_data_pin : true;
    level_shifter_data_pin : true;
    alive_during_power_up : false;
    input_voltage_range (0.85, 1.32);
    related_power_pin : VDDIN;
    related_ground_pin : VSS;
  }
}
```

- ❖ For clamp 0 cell:
  - ◆ The data pin has the `alive_during_power_up` attribute is set to true.
  - ◆ The cell has liberty attribute `is_isolation : true`, and `is_level_shifter : true`.
  - ◆ The output matches function `AND_NOT (y=I * !ISO)`.
  - ◆ The output pin and isolation enable pin has liberty attribute `alive_during_partial_power_down: true`

The following is an example library snippet of a NOR-style ELS cell for clamp 0:

```
cell (ISOLS0_NORM) {
  pin(I) {
    direction : input;
    isolation_cell_data_pin : true;
    level_shifter_data_pin : true;
    input_voltage_range (0.85, 1.32);
    related_power_pin : VDDIN;
    related_ground_pin : VSS;
  }
}
```



#### Note

By default, if the `alive_during_power_up` attribute is not defined in the data pin, the `alive_during_power_up` attribute is considered to be true.

### 5.2.2.2 Enabling NOR-style ELS Checks

By default, the NOR-style ELS checks are enabled. To disable these checks, set the `enable_iso_nor` application variable to `false`.

```
%vc_static_shell>set_app_var enable_iso_nor false
```

By default, this application variable is set to true.

### 5.2.2.3 VC LP Checks on NOR-style ELS Cells

VC LP reports the ISO\_INST\_SOURCE violation for NOR-style ELS cells if the `alive_during_power_up` attribute is placed on the data pin, and its value is true.

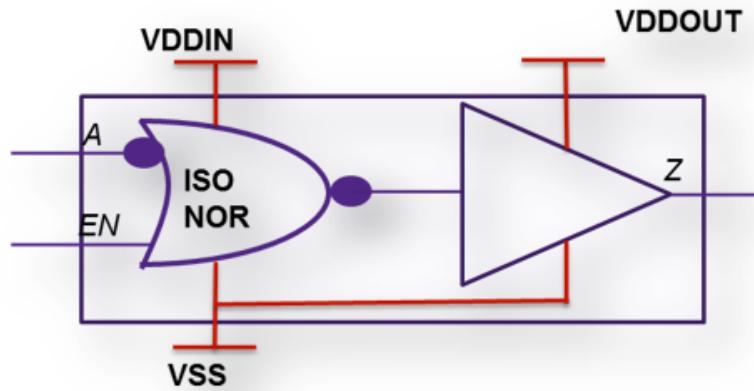
```
pin (D) {
    alive_during_power_up : true;
}
```

In the examples provided in this section, the data and enable are related to the VDDIN supply; the output is related to the VDDOUT supply (which will generally be at a different voltage); the VDDIN supply is the SCMR rail.

#### 5.2.2.3.1 NOR-style ELS Cells For Clamp 0

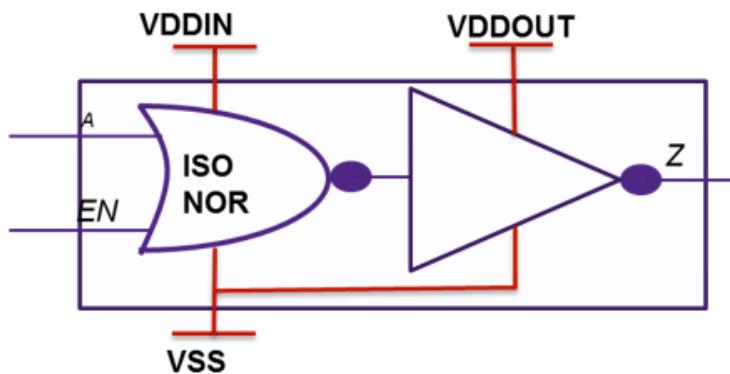
The cell which clamps to zero is standard NOR cell, as shown in [Figure 5-55](#). For this example, the cell's data pin attribute `alive_during_power_up` is not defined (which means that the value is true), hence ISO\_INST\_SOURCE is reported on the data pin.

**Figure 5-55 NOR-style ELS Cells with Clamp 0**



#### 5.2.2.3.2 NOR-style ELS Cells For Clamp 1

The cell which clamps to one is a standard NOR cell, followed by an inverter, as shown in [Figure 5-56](#). However, unlike the example in [Figure 5-56](#), this cell will not have an electrical short if the data input is floating when the cell is on. Therefore, the liberty model for this cell requires that the `alive_during_power_up` attribute is set to false.

**Figure 5-56 NOR-style ELS Cells with Clamp 1**

### 5.2.3 Support for NOR Style Isolation on Macro Outputs

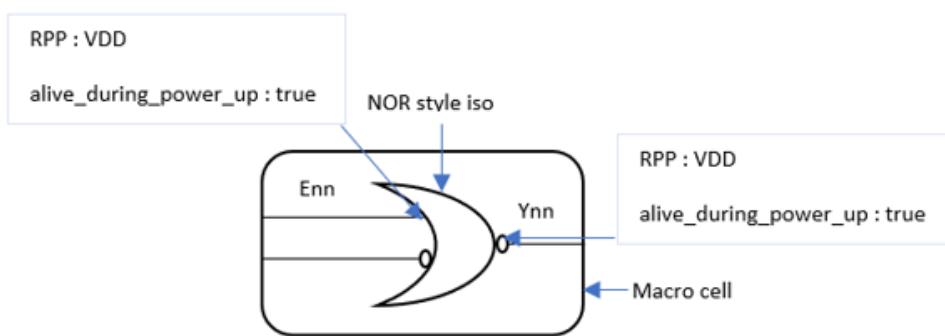
VC LP performs the same checks when a pin with the proper markings to indicate NOR style iso, appears as an output of a macro in a liberty file. VCLP performs NOR style iso related checks in following four scenarios.

- ❖ Enable is a single pin with same RPP as output
- ❖ Enable is an equation of all macro primary inputs
- ❖ Enable is missing
- ❖ alive\_during\_power\_up appears on a macro primary input

The scenarios are explained in the following section.

#### 5.2.3.1 Scenario 1: Enable is a single pin with same RPP as output

The following is an example standard cell nor style isolation gate with some key liberty attributes:



If several such gates appear as primary outputs of a macro, it can be represented as follows:



The following assumptions must be satisfied for this scenario.

- ❖ Enn, Ynn must have same RPP (related\_power\_pin)
- ❖ Enn must have alive\_during\_partial\_power\_down : true
- ❖ Ynn must have:
  - ◆ alive\_during\_partial\_power\_down : true
  - ◆ isolation\_enable\_condition : Enn

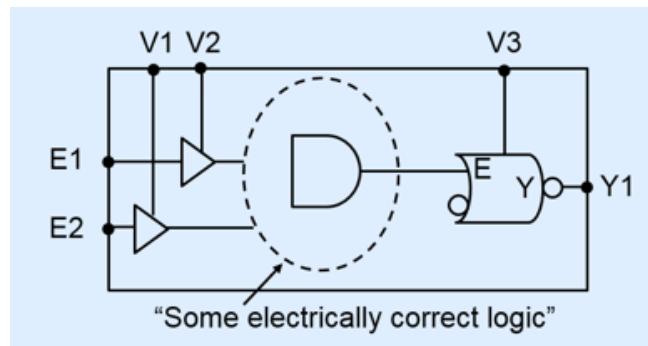
In other words, the isolation\_enable\_condition attribute must be present, and the condition must be a single primary input.

Currently in VCLP, we do sink forwarding from the standard NOR ISO cell enable pin to find the actual sink and do the on-ness check between control source and sink. This sink forwarding will be applied in a similar manner to the macro enables Enn as well.

### 5.2.3.2 Scenario 2: Enable is an equation of multiple primary inputs

In this case, there is some logic between the primary input pins and the enable pin of the NOR-style standard cell inside.

The following is an example isolation\_enable\_condition : E1 & E2.



The following assumptions must be satisfied for this scenario.

Ynn must have:

- ❖ alive\_during\_partial\_power\_down : true
- ❖ isolation\_enable\_condition : must be present but may be any function of other primary input pins. For example, Enn=E1&E2
- ❖ A new check ISO\_MULTINOR\_STATE has been implemented for the case where a macro output pin has alive\_during\_partial\_power\_down : true and multiple terms in isolation\_enable\_condition. When a pin is found matching both of these conditions, then a special rail order check is performed.

- ❖ This rail order check will trigger the violation, if there is a case where \*all\* of the related\_power\_pins of the enable inputs are off, and the sink is on.

*ISO\_MULTINOR\_STATE (Error, Design, Enabled by Default)*

*Description: Sink supply on, but supplies of nor style macro output off*

### 5.2.3.3 Scenario 3: Enable is missing

In cases, where `isolation_enable_condition` is missing, but the `a_d_p_p_d` attribute is present, then the output is handled as if its output is connected to an ideal always on supply. Any checking for the output pin is skipped.

The ISO\_EXPR\_MISSING violation is reported for scenarios where a macro output has the attribute `is_isolated : true`, but does not have the attribute `isolation_enable_condition`.

*ISO\_EXPR\_MISSING (Warning, Design, Disabled by Default)*

*Description: Isolation condition missing for pin with is\_isolated true*

### 5.2.3.4 Scenario 4: alive\_during\_power\_up appears on macro input

For standard cells, special handling is required at the input of a NOR-style isolation gate. Let us consider a scenario where the nor-style gate is powered on, but the driver of its data input is powered off. Here in this scenario, it is electrically incorrect due to an inverter at the data input of normal NOR-style cells.

VC LP checks for this electrical error with respect to the liberty attribute `alive_during_power_up`. If this attribute is present and true, the ISO\_INST\_SOURCE violation is reported.

#### 5.2.3.4.1 Handling of is\_isolated on outputs

The `lp_ignore_is_isolated_output` application variable is introduced with default value true.

The current behavior has not been changed. Normally, when `is_isolated : true` is found on an output pin, the pin is completely skipped for rail order checks.

But when this application variable is set to false, only pins with both `is_isolated : true` and `alive_during_partial_power_down : true` are skipped.

Pins which have `is_isolated : true` and do not have `alive_during_partial_power_down : true` are checked for rail order violations as normal. The `related_power_pin` of the isolated output is assumed to be the output power of the isolation cell. Then, when this power is off, ensure that the sink is off also.

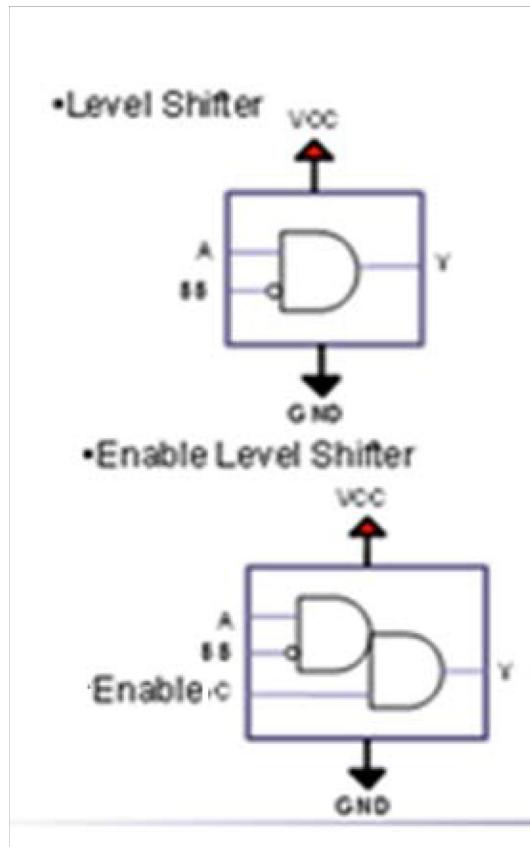
### 5.2.4 Differential LS

VC LP supports Differential Level Shifter (DLS) in a design. The following are the characteristics of a DLS:

- ❖ It does not need multiple voltage supplies for shifting, unlike a traditional Level Shifter
- ❖ It has two logical inputs acting as a differential input stage. One of the inputs is always inverted (in the example in [Figure 5-57](#), A and BB are the two inputs, and BB is inverted).

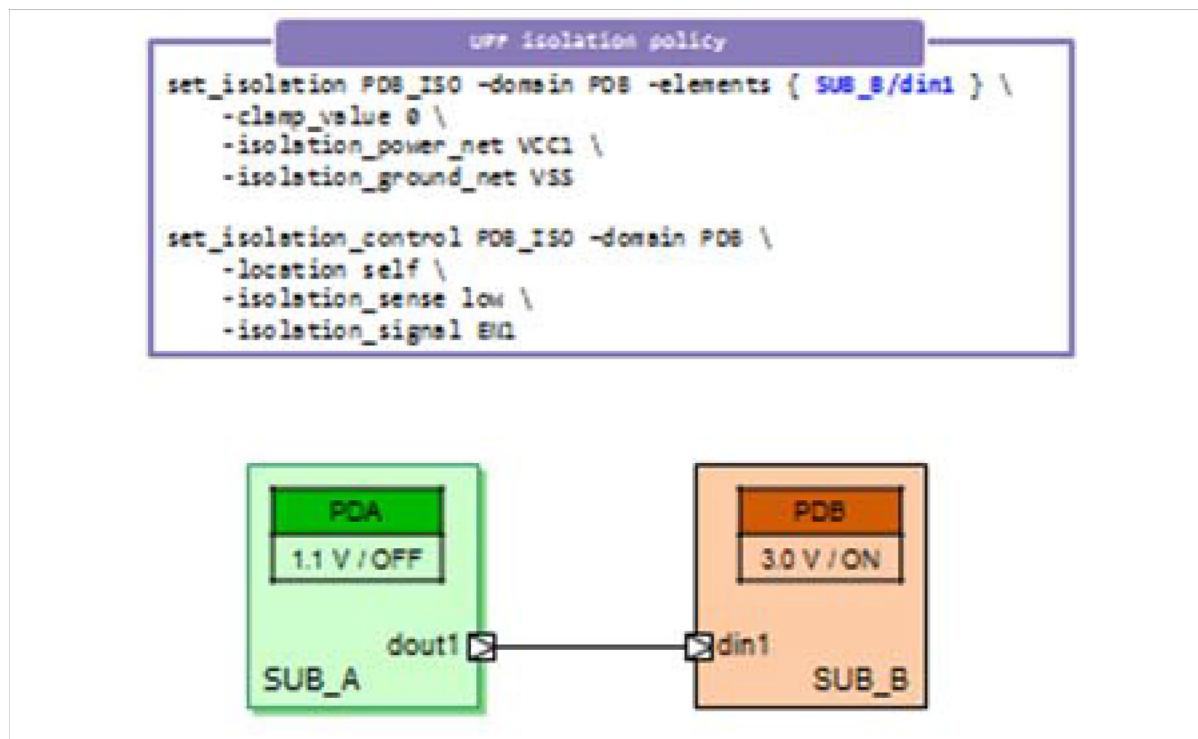
The advantage of adding the differential level shifter cells in a design is that it allows for easier power planning.

The following is an example of a DLS cell:

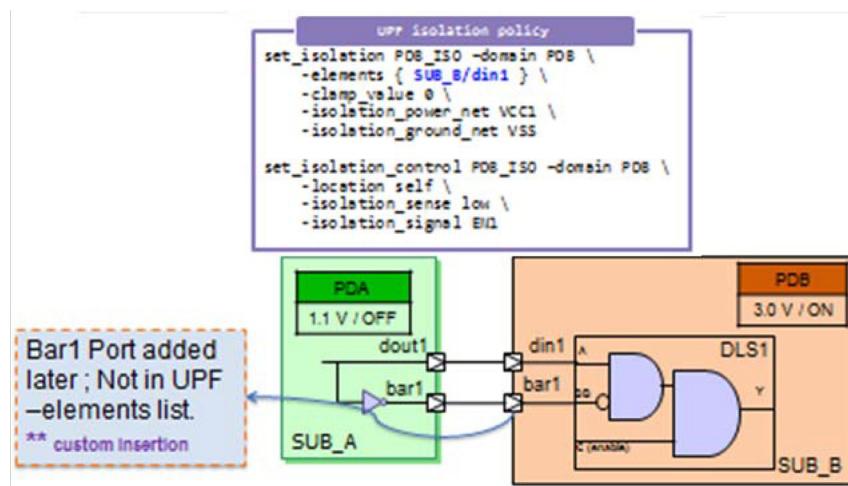
**Figure 5-57 Example of a DLS**

#### 5.2.4.1 DLS in the Implementation Flow

At the RTL phase, there is no concept of a DLS cell, as DLS cannot be represented in the UPF. As shown in [Figure 5-58](#), the crossing from SUB\_A/dout1 to SUB\_B/din1 is a just a plain wire and the UPF has an ISO policy applied to SUB\_B/din1.

**Figure 5-58 RTL Phase**

After synthesis, a DLS cell is inserted in the netlist as shown in [Figure 5-59](#). This is an example of the first RTL netlist. The ports SUB\_A/bar1 and SUB\_B/bar1 are created during synthesis to add a path for the inverted signal.

**Figure 5-59 Netlist Post Synthesis**

#### 5.2.4.2 Prerequisites for a Cell to be a DLS in VC LP

VC LP recognizes a cell to be a DLS, if the following prerequisites are met in the design.

- ❖ Liberty Attributes

The cell must have the following liberty attributes:

- ◆ The pin\_opposite or contention\_condition liberty attribute at the cell level.
- ◆ The is\_level\_shifter liberty attribute at the cell level.
- ◆ The Differential Enable Level Shifter (DELS) has the level\_shifter\_enable\_pin liberty attribute at the pin level.

[Figure 5-60](#) shows an example of the liberty definition of a DLS cell.

**Figure 5-60 Liberty Definition for DLS**

## Liberty Definition for D\_LS

```
cell(<cell_name>) {
    is_level_shifter : true ;
    pin_opposite("<pin_name>", "<pin_name>");
    contention_condition : "<boolean_expression>";
}

pin(<pin_name>) {
    direction : input;
    level_shifter_data_pin : true ;
    input_signal_level : <voltage_rail_name>;
    ...
}
pin(<pin_name>) {
    direction : input;
    level_shifter_data_pin : true ;
    input_signal_level : <voltage_rail_name>;
    ...
}
pin(<pin_name>) {
    direction : input;
    level_shifter_enable_pin : true ;
    ...
}
```

- ◆ The cell must be defined using the set\_attribute command.

```
set_attribute [get_lib_pins */DLS/A] complementary_pin "BB"
set_attribute [get_lib_pins */DLS/BB] master_complementary_pin "A"
```

When the complementary\_pin and master\_complementary\_pin attributes are not defined on the pins when the necessary liberty attributes for DLS are missing in liberty database, VC LP reports the DLS\_ATTRIBUTE\_MISSING violation.

- ❖ Design Connectivity Conditions

The following are the valid types of design connectivity conditions for a cell to qualify as a DLS:

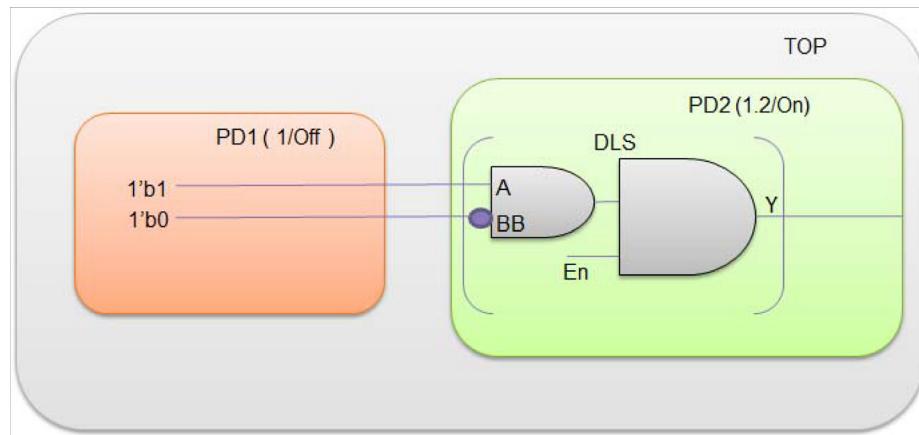
- ◆ Common root with inverse value

The driver root of the DLS input pin is Src, which is common, and there is an inverter on BB path which drives an inverse value.

**Figure 5-61 Legal Drivers for DLS cells**

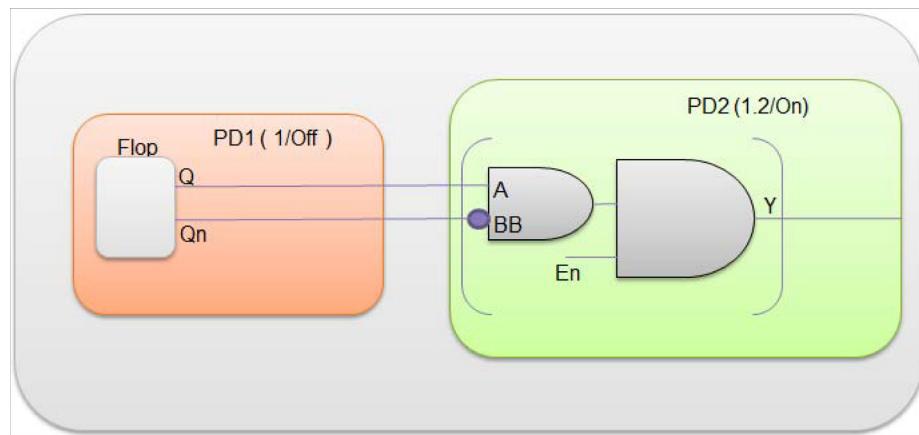
- ◆ Constants tied to inputs

The driver roots of DLS inputs are 1'b0 and 1'b1, which drives inverse value.

**Figure 5-62 Legal Drivers for DLS cells**

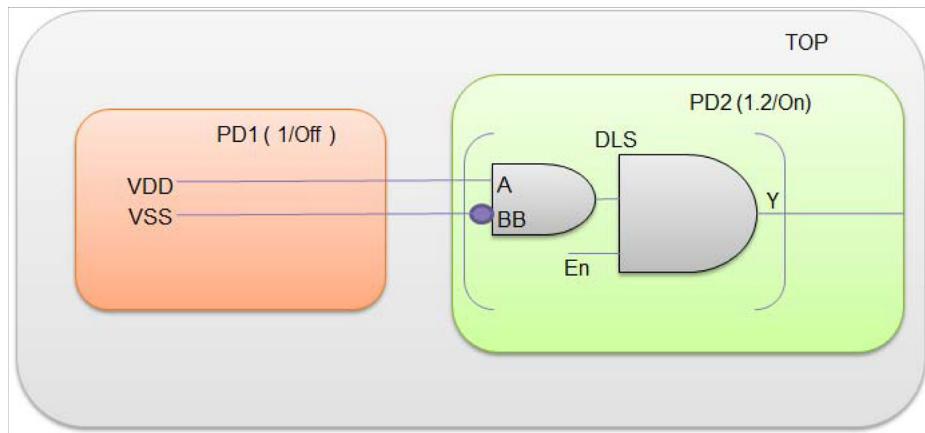
- ◆ Flop outputs Q/Qn

The driver roots of the DLS input pins are the Flop's Q/Qn, which drives inverse value.

**Figure 5-63 Legal Drivers for DLS cells**

- ◆ Power/Ground ports

The driver root of the DLS input pins are VDD/VSS, which drives inverse value.

**Figure 5-64 Legal Drivers for DLS cells**

If the design violates the prerequisite conditions from design connectivity point of view, the following warning messages are reported as part of the SETUP violations.

#### 5.2.4.3 DLS Violation Tags Messages

The following table provides the list of the violations reported for DLS cells:

**Table 5-5 Violation Tags DLS cells**

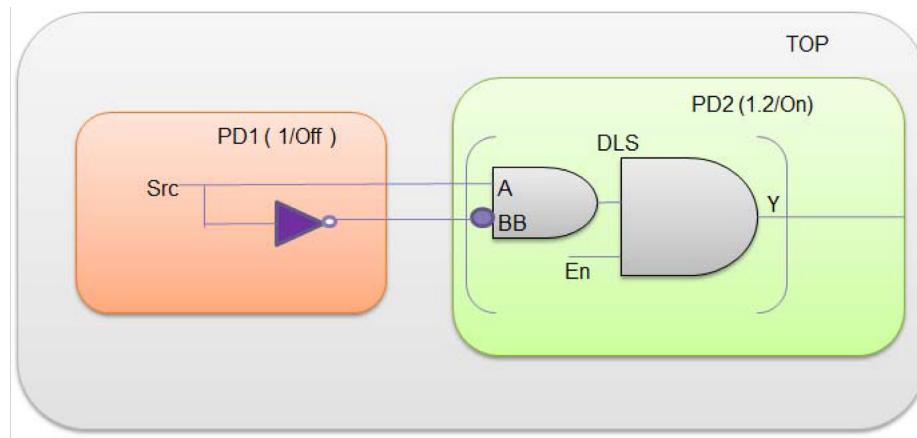
Parsing Message	Description
DLS_SETUP_ROOT	Incorrect connection has been created between DLS node and its' both the identified roots. The connection issue is indicated by the reason code. REASON_CODE list: <ul style="list-style-type: none"><li>• DLS_ROOT_DIFF: DLS node has two different drivers</li><li>• ROOT_POLARITY_SAME: Both the path from root to DLS node have same polarity</li><li>• FLOP_DRIVER_SUP_DIFF: Roots are output pin of flops, but having different supplies</li><li>• ROOT_CONST_SUPDIFF: Roots are literal constants but supplies are different</li></ul>
DLS_SETUP_PATH	Path connecting DLS node to one of its drivers has setup issue. The connection issue is indicated by the reason code. REASON_CODE list: <ul style="list-style-type: none"><li>• NON_BUF_NODE -&gt; The path contains a non buf/inv node</li><li>• INTR_SUP_DIFF -&gt; The intermediate node has different supply</li></ul>
DLS_SETUP_NOBOUNDARY	Path connecting DLS node to its driver is not crossing any power domain boundary.
DLS_SETUP_PG	Related power pin or related ground pin connection of complementary and master complementary pins of the DLS cell are not same.
DLS_ATTR_MISSING	complementary_pin or master_complementary_pin attribute is not defined for the DLS cell. Cell will not be treated as DLS cell.

#### 5.2.4.4 Other VC LP Checks on DLS Cells

At RTL, there are no changes in the existing violations, as the DLS cells cannot be identified from the UPF.

When a DLS is found in the design after netlist synthesis, although DLS is a split cell, VC LP recognizes the entire cell as a DLS and it suppresses certain checks for specials paths within the DLS, however, the entire set of LS related checks are still performed on the whole DLS cell treating DLS as a regular LS. The following are the checks that are suppressed on the DLS.

**Figure 5-65 Suppressed Checks for DLS**



In the example shown in [Figure 5-65](#),

- ❖ On the segment between driver-BB, VC LP does not report ISO\_STRAT\_MISSING, because at the RTL stage there is no such path between driver-BB in the design.
- ❖ On the crossover path passing through the BB pin, VC LP does not report ISO\_INST\_NOSTRAT for the Enabled DLS cell because at the RTL stage, the ISO strategy can be written on this path.

#### 5.2.5 MBIT Retention Cells

VC LP supports multi-bit retention cells. All the existing retention checks are supported for the multi-bit retention cells.

The following library snippet is an example for a multi-bit retention cell library:

```
cell ("MBIT_RR2") {
    retention_cell : "clock_low";
    pin (CLK) {
        ...
    }
    pin (RETN) {
        ...
    }
    bundle (D) {
        members ("D0", "D1");
        pin (D0) {
            ...
        }
        pin (D1) {
            ...
        }
    }
}
```

```

        }
        bundle (Q) {
            members("Q0", "Q1");
            pin (Q0) {
                ...
            }
            pin (Q1) {
            }
        }
    ...
}

```

## 5.2.6 Support for Multi-bit Protection (ISO/LS/ELS) Devices

By default, VC LP supports multi-bit protection (ISO/LS/ELS) devices present in the design. This impacts the crossover generation and strategy-cell association for paths where multi-bit protection devices are present. All the existing ISO\_\* / LS\_\* violations are supported/reported for multi-bit protection devices.

To disable this support, set the `handle_mbit_device` application variable to false.

## 5.2.7 Zero Pin Retention Cells

Zero pin retention cell is a standard retention cell, except that it has no save and restore pins. During retention, the clock must be clamped low and reset must be clamped high with always on signal. Zero pin retention cells must have the following Cell level liberty attribute.

```
retention_condition() {required_condition: "NRST &! CLK" ;}
```

NOR-style isolation cell is single rail device and it always clamps to zero when isolation control is high. It can be placed in switched (VDD off) domain and it still function correctly.

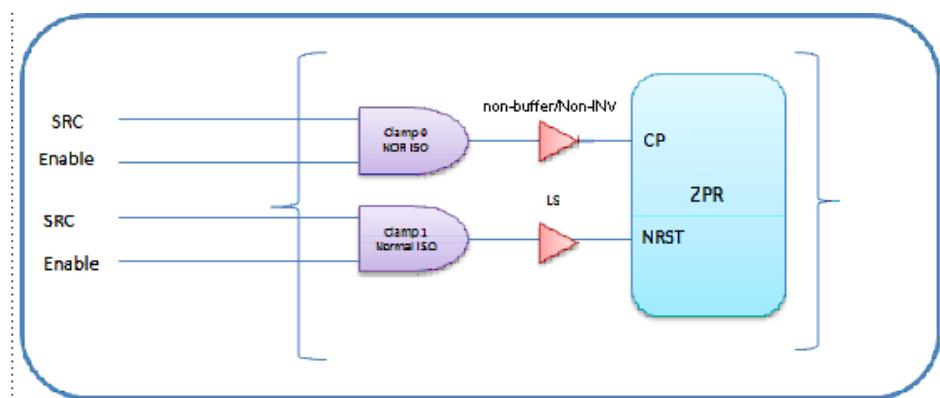
However, in designs where NOR-style isolation and Zero pin retention cells are together, the NOR-style isolation cell is expected to clamp the clock/reset pins of Zero pin retention flop cells.

Previously, VC LP did not support placing of cells between NOR-style isolation cell and Zero pin retention cell's clock/reset pin. You had to ensure that there was a clean wire connection between the NOR-style isolation cell and Zero pin retention cell. There was a requirement to remove this constraint as timing violations were reported on these paths. To fix these timing violations, AOB (Always-on Buffer) cells had to be placed between NOR-Style isolation cell and Zero pin retention flop.

VC LP enables you to add Always-On Buffer (AOB)/Buffer between NOR-style isolation cell and Zero Pin Retention's clock/reset pin. The following new checks are introduced for this support:

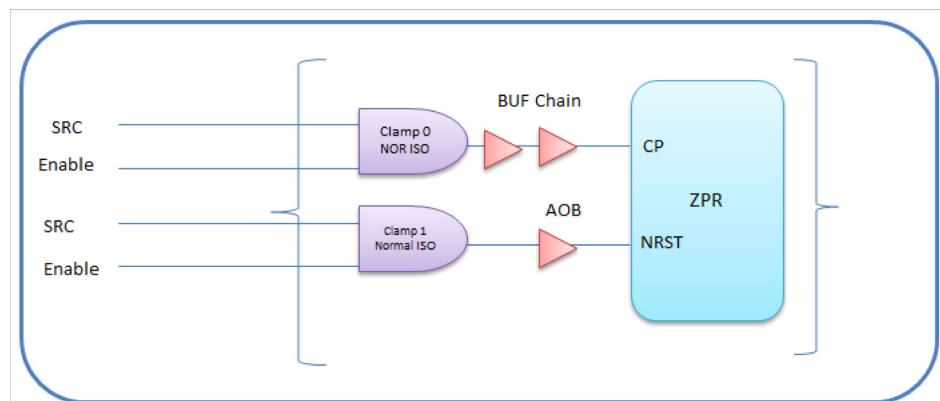
- ❖ RET\_CLAMP\_INCORRECT

If any non-buffer or non-inverter cell gets placed between NOR-style isolation cell and Zero pin Retention flop's clock/reset pin, then VC LP reports the RET\_CLAMP\_INCORRECT violation. The severity of this violation is Error.

**Figure 5-66 RET\_CLAMP\_INCORRECT Violation Example**

- ❖ RET\_CLAMP\_LEAKAGE

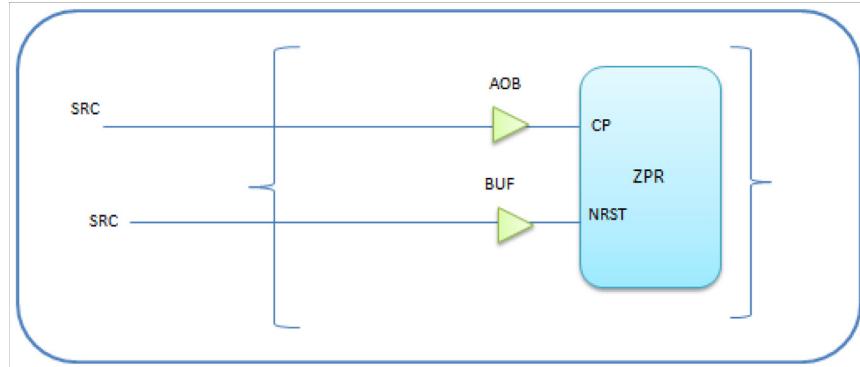
If the AOB cell's (Highlighted with Red) power/ground net is less ON compared to Zero pin retention flop's backup power/ground net, then VC LP reports the RET\_CLAMP\_LEAKAGE violation. The severity of this violation is error.

**Figure 5-67 RET\_CLAMP\_LEAKAGE Violation Example**

- ❖ RET\_CLAMP\_MISSING

From the Zero pin Retention Clock and Reset pin, VC LP does a back trace and if Clamp cells are not found on the path, then VC LP reports RET\_CLAMP\_MISSING violation.

**Figure 5-68 RET\_CLAMP\_MISSING Example**

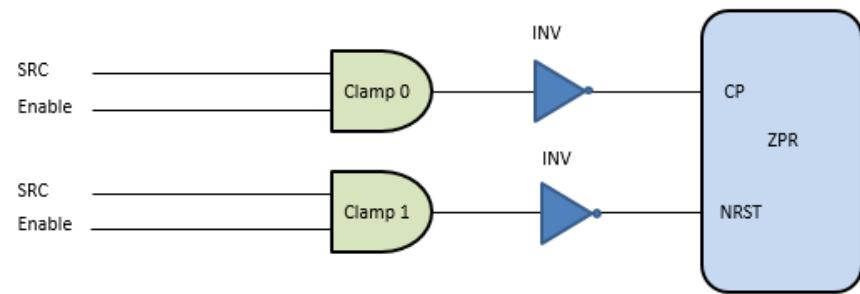


- ❖ RET\_CLAMP\_INVERT

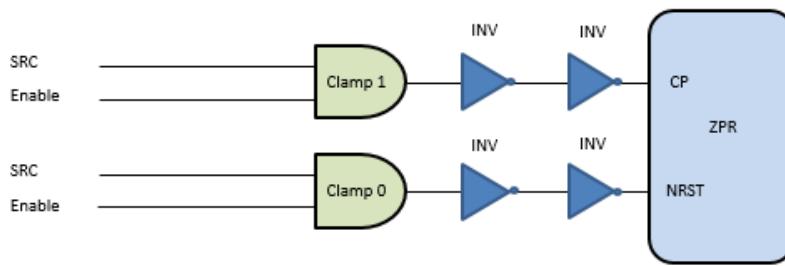
If any inverter cell gets placed between NOR-style isolation cell and Zero pin Retention flop's clock/reset pin, then VC LP reports the RET\_CLAMP\_INVERT violation. The severity of this violation is Error.

VC LP does not give RET\_CLAMP\_INVERT violation, when even number of inverters are used in between clamp cell and zero pin retention's clk/reset pins.

**Figure 5-69 RET\_CLAMP\_INVERT Example**



In the example in [Figure 5-70](#), the clamp cells are reversed means clamp 1 to CP pin and clamp 0 to NRST pin of ZPR. and if even number of inverters are used, then RET\_CLAMP\_INVERT violation is reported. However, if odd number of inverters are inserted, VC LP does not report RET\_CLAMP\_INVERT violation.

**Figure 5-70 RET\_CLAMP\_INVERT Example**

### 5.2.7.1 Support for map\_retention\_clamp\_cell Command

You can specify library cells in the zero pin retention (ZPR) clamp cells using the `map_retention_clamp_cell` command.

#### Syntax

```

vc_static_shell> map_retention_clamp_cell -help
Usage: map_retention_clamp_cell      # Specify library cells for mapping Zero Pin
Retention (ZPR) clamp cells
        -domain <domain_name>  (Specify the domain for retention strategies)
        [-clock_clamp_lib_cells <clock_clamp_lib_cells>]
                                (Specify library cells for mapping ZPR clamp cells on clock
                                path)
        [-async_clamp_lib_cells <async_clamp_lib_cells>]
                                (Specify library cells for mapping ZPR clamp cells on async
                                set/reset paths)
        <list_of_strategies>    (List of retention strategis)

```

The following new violations are reported for library cells in the zero pin retention (ZPR) clamp cells:

- ❖ UPF stage (`check_upf`)
  - ◆ RET\_MAPCLAMP\_UNAVAIL: This violation is reported when the library cell in `map_retention_clamp_cell` command does not exist.
  - ◆ RET\_MAPCLAMP\_MISMATCH: This violation is reported when the library cell in `map_retention_clamp_cell` command is not ISO/ELS type, or the listed library cells are correct type, but the strategy is not mapped to ZPR library cell.
- ❖ Design stage (`check_design`)
  - ◆ RET\_MAPCLAMP\_WRONG: This violation is reported when the clamp cell instance for a ZPR retention strategy does not match with the library cells listed in the `map_retention_clamp_cell` command.

### 5.2.8 Diodes

VC LP identifies a cell a diode if:

- ❖ LibCell has `user_function_class :: ANTENNA` attribute and `antenna_diode_type` attribute mentioned with any of these values - `power_and_ground`, `power`, `ground`
- ❖ LibCell has `user_function_class :: DIODE` attribute and `antenna_diode_type` attribute mentioned with any of these values - `power_and_ground`, `power`, `ground`
- ❖ LibCell has `antenna_diode_type` attribute in the Liberty library

- ❖ LibCell has does not contain the user\_function\_class :: ANTENNA or user\_function\_class :: DIODE attribute, however, the following variables are set:
  - ◆ set\_app\_var vss\_type\_diode\_cell\_name <cell\_name>
  - ◆ set\_app\_var vdd\_type\_diode\_cell\_name <cell\_name>
  - ◆ set\_app\_var vdd\_vss\_type\_diode\_cell\_name <cell\_name>

VC LP reports the DIODE\_CHECK\_IGNORE warning message, if diode checks are ignored. This warning is reported after executing any the following commands:

```
%vc_static_shell>check_lp -stage pg
%vc_static_shell>check_lp -stage pg -family diode
%vc_static_shell>check_lp -stage pg -force
%vc_static_shell>check_lp -stage pg -include_info
```

Warning Example:

[Warning] DIODE\_CHECK\_IGNORE: Checks for diode cell have been skipped.

The diode cell antenna/ANTENNA\_RVT\_LOCAL is used in the design but does not have any of the vdd\_type\_diode\_cell\_name, vss\_type\_diode\_cell\_name and vdd\_vss\_type\_diode\_cell\_name attributes.

DIODE\_CHECK\_IGNORE warning is reported in the following scenarios:

- ❖ LibCell has user\_function\_class :: ANTENNA attribute but no antenna\_diode\_type attribute is mentioned in the Liberty library
- ❖ LibCell has user\_function\_class :: DIODE attribute but no antenna\_diode\_type attribute is mentioned in the Liberty library

Application Variables

- ❖ To set the cell as a vss type diode, use the following application variable:  
`%vc_static_shell>set_app_var vss_type_diode_cell_name <cell_name>`
- ❖ To set the cell as a vdd type diode, use the following application variable  
`%vc_static_shell>set_app_var vdd_type_diode_cell_name <cell_name>`
- ❖ To set the cell as a vdd\_vss type diode, use the following application variable  
`%vc_static_shell>set_app_var vdd_vss_type_diode_cell_name <cell_name>`

## 5.2.9 Block/IP Level ISO Signals Connectivity Checks

VC LP checks the connectivity from the block/IP level isolation control signal to the correct SoC signal and the driver supply of the strategy node/data pin of the isolation cell. To enable this support, the reference\_toplevel\_isolation\_signal Tcl command is introduced. Using this command, you can specify the name of the correct isolation enable signal at the SoC level, and the correct driver supply of the isolation data pin.

### Use Model

```
vc_static_shell> reference_toplevel_isolation_signal -help
Usage: reference_toplevel_isolation_signal # specify the name of the correct
isolation enable signal at SoC level, and the correct supply of the driver of isolation
data pin
      -name <str>          (top level iso enable name)
      -supply <str>         (correct supply of the driver of isolation data pin)
```

VC LP has also introduced the following violation tags to check the connectivity of the block level isolation\_signal to the top level and the driver supply of the strategy node/data pin of the isolation cell.

- ❖ ISO\_STRATEGY\_REFERENCE

When a strategy has a defined `-isolation_signal`, that does not match with the signal specified in the `reference_toplevel_isolation_signal` command, the ISO\_STRATEGY\_REFERENCE violation is reported.

- ❖ ISO\_INST\_REFERENCE

When an isolation cell has an enable signal that does not match with the signal specified in the `reference_toplevel_isolation_signal` command, the ISO\_INST\_REFERENCE violation is reported.

- ❖ ISO\_STRATEGY\_SUPPLY

When `-isolation_signal` matches with a given isolation signal in the `reference_toplevel_isolation_signal` command, but the driver supply of the strategy node does not match with the supply specified in the `reference_toplevel_isolation_signal` command, then the ISO\_STRATEGY\_SUPPLY violation is reported.

- ❖ ISO\_INST\_SUPPLY

When the `-isolation_signal` of an ISO cell matches with the isolation signal specified in the `reference_toplevel_isolation_signal` command but the driver supply of the data pin of the isolation signal does not match with the supply specified in the tcl command, then the ISO\_INST\_SUPPLY violation is reported.

The severity of these tags is error and by default these tags are disabled. Use the `configure_tag -app LP` command to enable the tags.

The following violations are introduced to guide the application of isolation strategies on top level ports, assuming that the sub IPs in the design have also run these checks at their block level validation.

- ❖ ISO\_STRATEGY\_INPUT

It is not recommended to define an isolation strategy on the topmost input ports. This violation will be reported if an isolation strategy is detected to be defined on the primary input ports.

If this behavior is not intended, ensure not apply an isolation strategy on the primary inputs.

- ❖ ISO\_STRATEGY\_SINK

If the topmost instance is not defined as a soft macro with the `set_design_attribute -attribute is_soft_macro true` command, it is not recommended to define a path specific isolation strategy on its primary output ports with the `-sink` option. If this behavior is not intended, ensure remove the sink info.

- ❖ ISO\_STRATEGY\_DIFFSUPPLY

If a top most instance is not defined as a soft macro with the `set_design_attribute -attribute is_soft_macro true`, it is not recommended to define a path specific isolation strategy on its primary output ports with the `-diff_supply_only true` option. If this behavior is not intended, make sure remove the "`-diff_supply_only`" option.

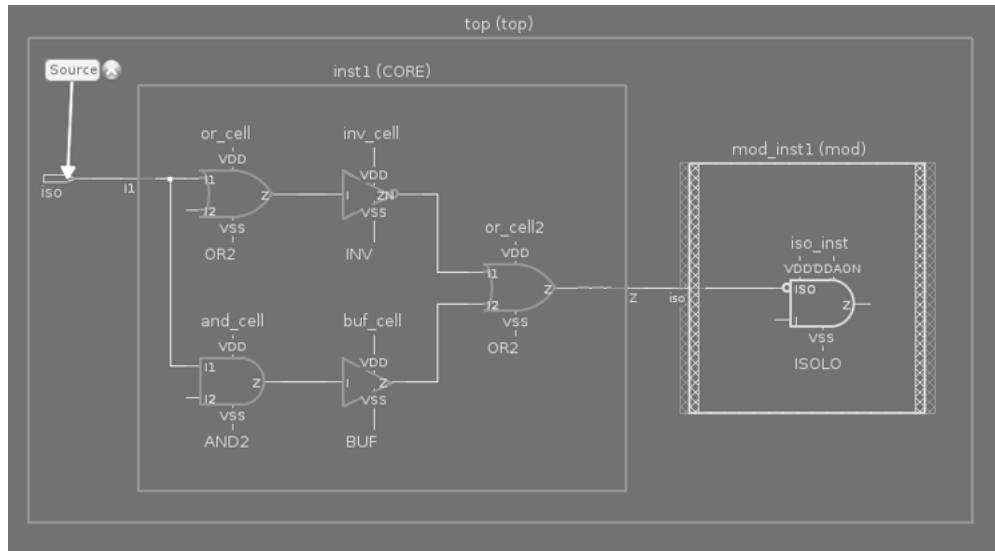
- ❖ ISO\_STRATEGY\_PATH

For a topmost instance which is not defined as a soft macro with the `set_design_attribute -attribute is_soft_macro true`, if its primary output port has any isolation strategy applied, and source supply is specified for the isolation strategy, the source supply is expected to be in the switch

path fanout of the isolation supply. Please add power switch strategies or cells to make sure the isolation supply is the power switch ancestors of the source supply.

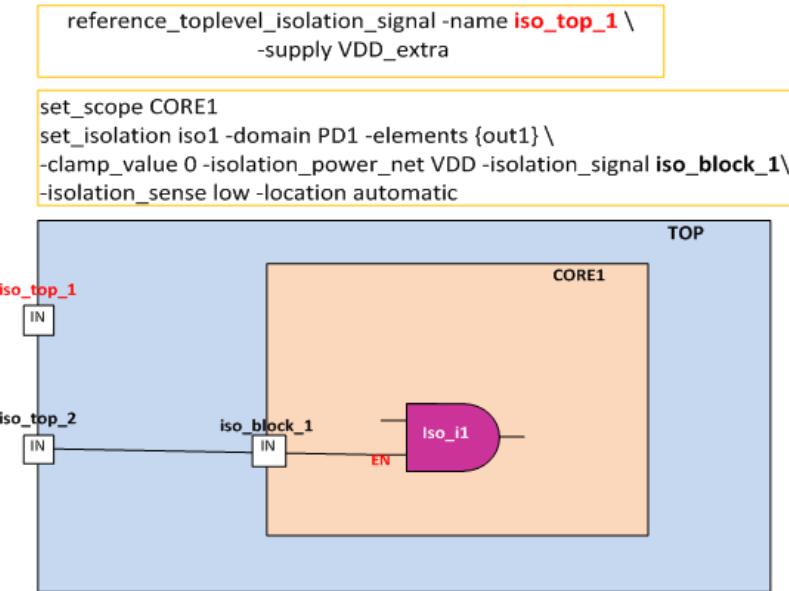
### 5.2.9.1 The reference\_toplevel\_iso\_skip\_combo Application Variable

The reference\_toplevel\_iso\_skip\_combo application variable specifies if the ISO\_INST\_REFERENCE and ISO\_STRATEGY\_REFERENCE should skip combinational cells whose output pin is related to signal input pin. The application variable is used to identify the correct reference top-level isolation signal. By default, the reference\_toplevel\_iso\_skip\_combo application variable is set to false. When the reference\_toplevel\_iso\_skip\_combo application variable is set to true, the ISO\_INST\_REFERENCE and ISO\_STRATEGY\_REFERENCE checks skips the combinational cells of inst1(CORE).



### 5.2.9.2 Example for ISO\_STRATEGY\_REFERENCE/ISO\_INST\_REFERENCE

Figure 5-71 shows an example scenario when the ISO\_STRATEGY\_REFERENCE/ISO\_INST\_REFERENCE violation is reported.

**Figure 5-71 ISO\_STRATEGY\_REFERENCE/ISO\_INST\_REFERENCE Violation Example****ISO\_STRATEGY\_REFERENCE Violation Example**

Tag : ISO\_STRATEGY\_REFERENCE  
 Description : Signal [DesignNets] reaching isolation control signal [UPFNet] defined for isolation strategy [Strategy] is not defined as a reference top level isolation signal.  
 Violation : LP:1  
 Strategy : CORE1/PD1/iso1  
 UPFNet  
 NetName : iso\_block\_1  
 NetType : Design/UPF  
 DesignNets  
 DesignNet  
 NetName : iso\_top\_2  
 NetType : Design

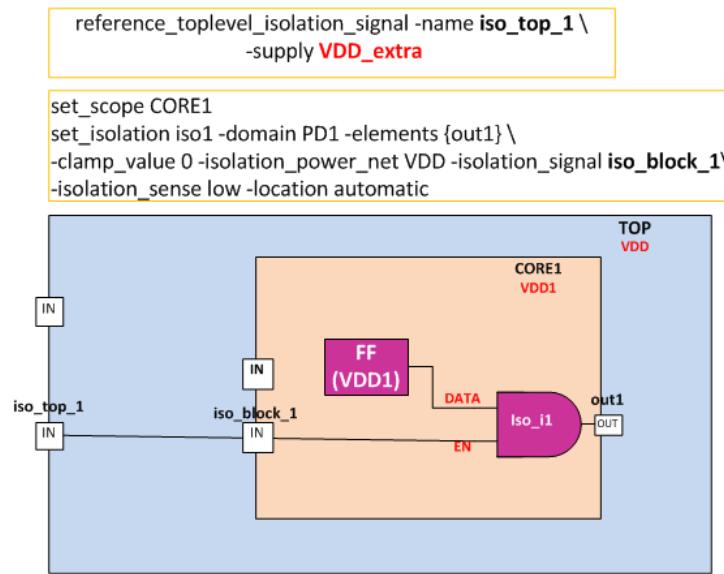
**ISO\_INST\_REFERENCE Violation Example**

Tag : ISO\_INST\_REFERENCE  
 Description : Signal [DesignNets] reaching isolation enable pin [CellPin] of isolation cell [Cell] is not defined as a reference top level isolation signal  
 Violation : LP:2  
 DesignNets  
 DesignNet  
 NetName : iso\_top\_2  
 NetType : Design  
 Instance : CORE1/iso\_i1  
 Cell : DMISAN2  
 CellPin : B

### 5.2.9.3 Example for ISO\_STRATEGY\_SUPPLY/ISO\_INST\_SUPPLY

Figure 5-72 shows an example scenario when the ISO\_STRATEGY\_SUPPLY/ISO\_INST\_SUPPLY violation is reported.

**Figure 5-72 ISO\_STRATEGY\_SUPPLY/ISO\_INST\_SUPPLY**



#### ISO\_STRATEGY\_SUPPLY Violation Example

Tag	: ISO_STRATEGY_SUPPLY
Description	: Supply [ReferSupply] specified for reference top level isolation signal [ReferToplevelIsoEn] does not match the source supply [UPFSupply] for domain boundary port [StrategyNode], on which isolation strategy [Strategy] is applicable
Violation	: LP:2
Strategy	: CORE1/PD1/iso1
StrategyNode	: CORE1/out1
ReferToplevelIsoEn	: iso_top_1
ReferSupply	: VDD_extra
UPFSupply	: VDD1

#### ISO\_INST\_SUPPLY Violation Example

Tag	: ISO_INST_SUPPLY
Description	: Supply [ReferSupply] specified for reference top level isolation signal [ReferToplevelIsoEn] does not match the source supply [UPFSupply] for isolation cell [Cell]
Violation	: LP:5
ReferToplevelIsoEn	: iso_top_2
ReferSupply	: VDD_extra
UPFSupply	: VDD1
Instance	: CORE1/iso_i1
Cell	: DMISAN2
CellPin	: B

## 5.2.10 Support for Local Bias For Isolation and Retention Cells

Electrical continuity requires that all logic elements in the same power domain share the same bias supplies. Previously, in the Synopsys UPF bias flow, you had supply a UPF which satisfies the following condition:

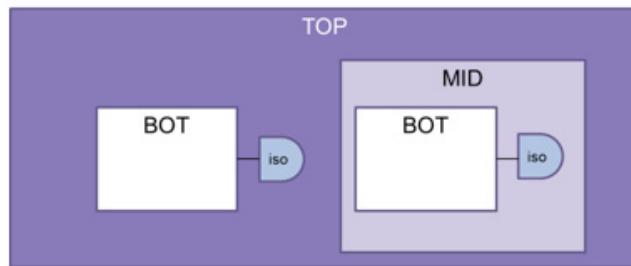
*The bias components of a supply set specified for an isolation/retention strategy must be the same as those of the primary supply of domain in which the power management cell is to be inserted; else, the design compiler issues an error otherwise and halts compile.*

Previously, to meet this condition, you had to provide a dedicated supply set using the back up power/ground supply, and the local bias had to be assembled for an isolation/retention strategy. This is an extra overhead when porting existing non-bias UPFs to the bias flow. At worst, adding such a supply set might not even be possible.

All supply sets were created with four supply functions (power, ground, nwell, and pwell). In many cases, the supply sets actually share the same nwell and pwell supplies (for example, the primary and isolation supplies in the same power domain must have the same bias), yet you had to perform the extra overhead of connecting/associating the bias functions of these supply sets to the same supply net.

Figure 5-73 shows an example of how it can be impossible to specify an isolation supply set to satisfy the bias continuity requirement. In this example, power domain BOT has two root cells, one nested in the power domain TOP, and the other in the power domain MID. Suppose the primary bias of TOP and MID are different, then it is impossible to specify an isolation supply set iso\_ss for the location -parent strategy iso1 which satisfies the bias continuity requirement of both TOP and MID.

**Figure 5-73 Bias Continuity**



```
set_isolation iso1 -domain BOT -location parent -isolation_supply_set iso_ss
```

VC LP assigns local bias to all logic elements in the same power domain to reduce overhead of manually adding them.

All standard and power management cells use the domain primary bias supply:

- ❖ When a two-function (power and ground) supply set is specified in a power management strategy (isolation/retention), the associated power management cell is assumed to use the local bias of the domain where it resides
- ❖ When a four-function supply set is specified in a power management (isolation/retention) strategy, the bias supplies in the supply set is ignored; local bias is used and a parser message with severity INFO is reported. This support is enabled by default. You can disable this support using application variables as discussed in the following sections.

## 5.2.10.1 Identification of Two and Four Function Supply Set

- ❖ By default, all supply sets have two supply functions only namely, power and ground.

- ❖ 4-function supply sets (power, ground, nwell and pwell) are created only when needed (that is, create-upon-reference). A bias supply function is referenced in the following scenarios:
  - ◆ When any bias function of a supply set is explicitly resolved to a supply net.
  - ◆ When any bias function of a supply set is referenced in a UPF command or a PG netlist.
  - ◆ When a supply set is associated with another supply set which has four functions.

These upgraded supply sets will always contain 4 functions: power, ground, nwell, pwell.

### Example

Consider the following UPF snippet:

```
create_supply_set SS_TOP
create_supply_set SS1
create_supply_set SS2
create_supply_set SS3 -function {nwell nwell_sn} -function {pwell pwell_sn} -update
create_supply_set SS4
create_power_domain PD_TOP -include_scope -supply {primary SS_TOP}
create_power_domain PD1 -elements {B1} -supply {primary SS1}
set_isolation ISO1 -domain PD_TOP -isolation_supply_set SS1 -elements {...}
set_isolation ISO2 -domain PD_TOP -isolation_supply_set SS2 -elements {...}
connect_supply_net SS4.nwell -ports {macro_cell/VNW}
```

In this example,

- ❖ SS\_TOP and SS1 are two-function supply sets.
- ❖ SS2 is a two-function supply set, as its bias functions have not been referenced anywhere.
- ❖ SS3 is a four-function supply set, as its bias functions are explicitly resolved.
- ❖ SS4 is a four-function supply set, as one of its bias functions is explicitly referenced in a connect\_supply\_net command.

## 5.2.10.2 Local Bias for Isolation Strategies

### 5.2.10.2.1 Using Two Function Supply Set

Consider following UPF example,

```
create_supply_set SS_TOP -function {power VDD_TOP} -function {ground VSS}
create_supply_set SS_ISO -function {power VDD_ISO} -function {ground VSS}
create_supply_set SS_PD1 -function {power VDD_PD1} -function {ground VSS}
create_supply_set SS_TOP -function {nwell nwell_top} -function {pwell pwell_top}
create_power_domain PD_TOP -include_scope -supply {primary SS_TOP}
create_power_domain PD1 -elements {B1} -supply {primary SS_PD1}
set_isolation ISO1 -domain PD1 -isolation_supply_set SS_ISO -elements {B1/out2} -
location parent
```

In above example,

SS\_ISO is a two-function supply set as it does not have pwell and nwell supply nets. ISO1 strategy has supply set SS\_ISO which does not have bias supply. Hence local bias is being used. Isolation cells inserted for ISO1 will continue to use the local bias of PD\_TOP. In this case, VC LP does not report a parser message.

### 5.2.10.2.2 Using Four Function Supply Set

Consider following UPF example where a four-function isolation supply set is used:

```
create_supply_set SS_TOP -function {power VDD_TOP} -function {ground VSS}
```

```

create_supply_set SS_ISO -function {power VDD_ISO} -function {ground VSS}
create_supply_set SS_PD1 - function {power VDD_PD1} -function {ground VSS}
create_supply_set SS_TOP -function {nwell nwell_top} -function {pwell pwell_top}
create_supply_set SS_ISO -function {nwell nwell_ISO} -function {pwell pwell_ISO}
create_power_domain PD_TOP -include_scope -supply {primary SS_TOP}
create_power_domain PD1 -elements {B1} -supply {primary SS_PD1}
set_isolation ISO1 -domain PD1 -isolation_supply_set TOP.primary -elements {B1/out1} -
parent
set_isolation ISO2 -domain PD1 -isolation_supply_set SS_ISO -elements {B1/out2} -parent

```

In above example, both SS\_TOP and SS\_ISO are four-function supply sets.

Previously,

- ❖ ISO1 strategy is related to TOP.primary and isolation cell inserted for ISO1 strategy use bias supplies defined in SS\_TOP supply set.
- ❖ ISO2 strategy is related to SS\_ISO and isolation cell inserted for ISO2 strategy use bias supplies defined in SS\_ISO supply set.

The isolations cells that are inserted for both ISO1 and ISO2 use the local bias of PD\_TOP. The following informational parser message is reported one each for ISO1 and ISO2, that the supply set bias supplies are ignored and the local bias supplies are used instead.

*UPF\_ISOLATION\_BIAS: Supply set <supply-set-name> for isolation strategy <isolation-strategy-name> has bias functions. Local bias will be used for checking instead.*

This support is enabled by default. You can set the `use_local_bias_for_isolation` application variable to false to disable this support.



### Note

ISO1 uses TOP.primary and isolation cell is placed in parent domain of PD1 (which is again TOP). Hence local bias is same as the bias supplies in isolation supply set, but the parser message is reported for ISO1.

#### 5.2.10.3 Local Bias for Retention Strategies

The same behavior (proposed above for isolation) is supported for retention strategies as well. If a four function supply set is specified for retention supply set (in retention strategy), then following information parser message is reported.

*UPF\_RETENTION\_BIAS: Supply set <supply-set-name> for retention strategy <retention-strategy-name> has bias functions. Local bias will be used for checking instead.*

This support is enabled by default. You can set the `use_local_bias_for_retention` application variable to false to disable this support.

#### 5.2.10.4 Impact on Existing Tags

When the `use_local_bias_for_isolation` or `use_local_bias_for_retention` application variable is set to true:

- ❖ If a supply set in isolation strategy contains bias functions, then UPF\_ISOLATION\_BIAS message is reported.
- ❖ If a supply set in retention strategy contains bias functions, then UPF\_RETENTION\_BIAS message is reported.
- ❖ The PG\_DOMAIN\_CONN violation is reported, when the bias pin of isolation cell is not connected to the local domain bias supply.

- ❖ The PG\_STRATEGY\_CONN violation is not reported when the bias pin of isolation cell is not connected to the bias supply specified in the supply set of the isolation strategy.
- ❖ There is no change in any other existing tag(s).

When the `use_local_bias_for_isolation` or `use_local_bias_for_retention` application variable is set to false:

- ❖ The UPF\_ISOLATION\_BIAS/UPF\_RETENTION\_BIAS message is not reported.
- ❖ The PG\_DOMAIN\_CONN violation is not reported, when the bias pin of isolation cell is not connected to local domain bias supply.
- ❖ The PG\_STRATEGY\_CONN violation is reported when the bias pin of isolation cell is not connected to bias supply specified in supply set in isolation strategy.

### 5.2.11 Support for PAD Cells

PAD cells are actually the ports for DUT in real silicon but because of the way verilog/vhdl has to be coded, the top module always have PAD cells as instances driving/receiving from top level ports.

The liberty snippet of a PAD cell is as follows:

- ❖ The cell should have “pad\_cell” attribute
- ❖ The cell pin should have “is\_pad” attribute

```
cell(PADBUS) {
  pad_cell : true ;
  bus("CTL_CLK") {
    bus_type : "blah"
    direction : input;
    is_pad : true ;
    related_power_pin : VIO;
    related_ground_pin : VSS;
    pin(CTL_CLK[0]) {
      related_power_pin : VDD
      related_ground_pin : VSS;
    }
    pin(CTL_CLK[1]) {
      related_power_pin : VDD;
      related_ground_pin : VSS;
    }
  }
}
```

VC LP supports `is_pad` attribute on bus bit when the `ignore_pad_cell_xovers` application variable is set to false. By default, this application variable is set to false.

When the `ignore_pad_cell_xovers` application variable is set to true:

- ❖ VC LP automatically detects crossovers between \*logic pins\* (that have `is_pad: true` attribute) directly driving or receiving top level ports and exclude them from doing ISO/LS checks.
- ❖ VC LP does not perform ISO/LS design checks for the path whose LogicSrc is top/input and LogicSink is PAD input (`is_pad true`) and vice-versa.

#### 5.2.11.1 Supply of Top port Supply That is Driving/Driven by a Pad Pin

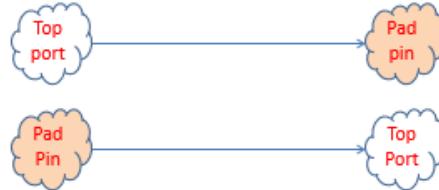
For a crossing between top level ports and pad cells UPF strategies may be present ,but it is not physically possible to insert isolation level shifters between a pad cell and top level port.

If a top level port is driving/driven by a pad pin voltage of top port should be taken as voltage of pad pin voltage. This support is controlled by application variable `assign_port_supply_from_padpin` which is set to true by default.

### Note

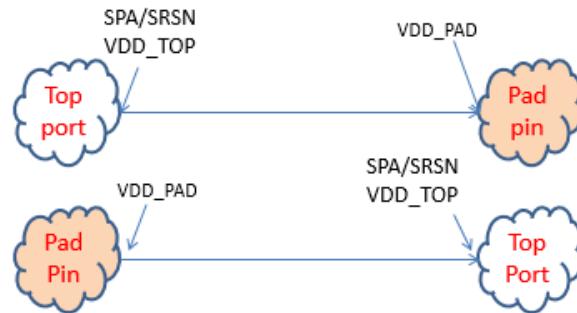
This is true, when the PAD cell is a global source or global sink. The crossover generation should not pass through a PAD cell. You can disable this feature by setting the `assign_port_supply_from_padpin` application variable to false.

- ❖ **Case 1:** When the pad cell is directly connected to the Top port.



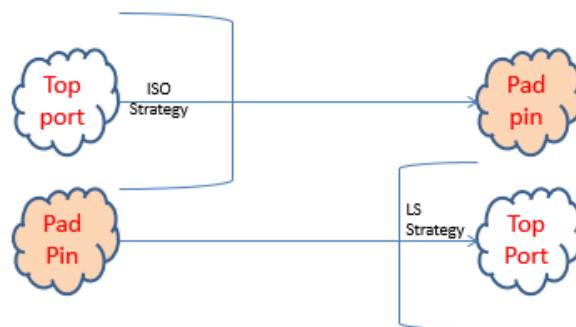
In the design Top, the ports are directly driving and receiving pad pins, VC LP overrides the Top port supply as PAD supply, and no ISO/LS/RAIL checks are performed on this crossover.

- ❖ **Case 2 :** SPA/SRSN is defined on the Top domain port



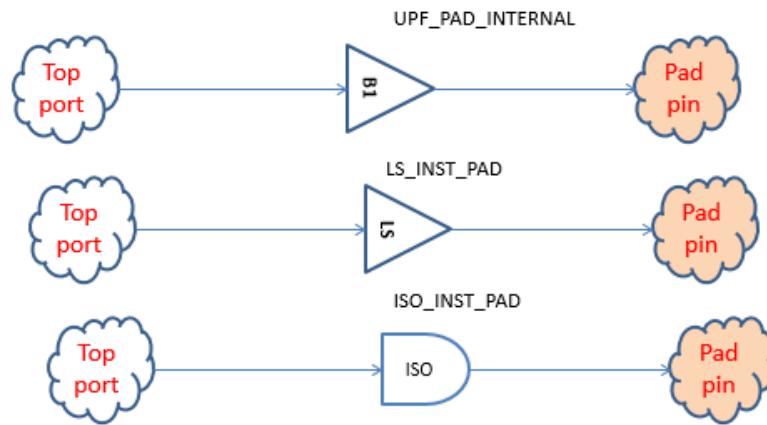
When the SPA/SRSN is defined on the Top port, but it is different than PAD pin, VC LP overrides the Top port supply as the VDD\_PAD supply and reports the UPF\_PORT\_SUPPLY violation, and no ISO/LS/RAIL checks are performed on these crossovers.

- ❖ **Case 3 : Isolation/Level Shifter Strategy is defined on the Top port to pad pin path**



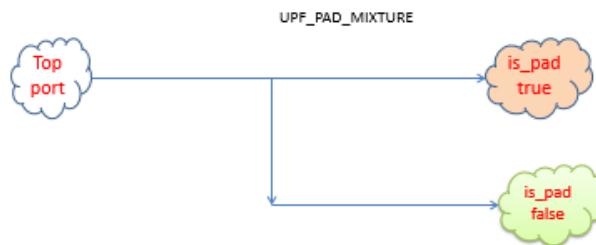
If the isolation and level shifter strategy is defined in the path of the Top port and PAD pin, VC LP reports ISO\_STRATEGY\_PAD and LS\_STRATEGY\_PAD. VC LP overrides the Top port supply as the PAD pin supply. All crossover based checks are performed considering the new supply. The LS/ISO\_INST\_MISSING violations are not reported on this path.

- ❖ **Case 4 : BUF/INV/Protection cell is driving pad pin**



If buffers/inverters or ISO-LS cells are present in the path of the Top port and the PAD pin, VC LP does not override the Top port supply with the pad-pin supply, and reports UPF\_PAD\_INTERNAL, LS\_INST\_PAD, and ISO\_INST\_PAD violations. All checks on this path are performed considering the actual supply.

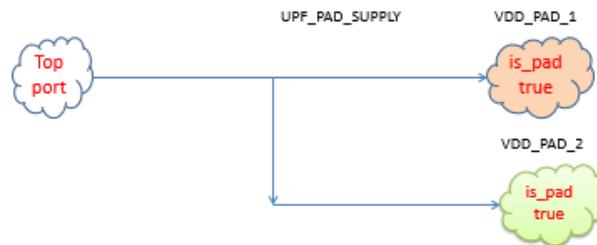
- ❖ **Case 5 : Top port connected to pad pin and standard cell.**



If the same Top port is directly connected to `is_pad true` and `is_pad false` pins, then VC LP reports the UPF\_PAD\_MIXTURE violation. VC LP does not perform checks on Top port to `is_pad true` path. On Top port to `is_pad false` path, VC LP performs checks considering the new supply.

VC LP reports the UPF\_PAD\_MIXTURE violation for each possible combination when non-pad cells are connected to pad cells combinations without any optimization.

- ❖ **Case 6 : Top port connected to pad pins having different supplies.**



If the same Top port is directly connected to `is_pad true` pins having different supply, then VC LP reports the UPF\_PAD\_SUPPLY violation. VC LP does not override the Top port supply. VC LP performs checks on the Top port to `is_pad true` path.

❖ **Case 7 : Pad cell is directly connected to Top PG Port**



In design Top, the PG ports directly driving and receiving pad pins. VC LP reports the UPF\_PORT\_PAD violation for this issue. VC LP does not override the supply of the Top ports. No ISO/LS/RAIL checks are performed on this crossover.

#### 5.2.11.2 Differences in VC LP Checks When assign\_port\_supply\_from\_padpin is Set to True/False

The following table summarizes the differences in the behavior when the `assign_port_supply_from_padpin` application variable is set to true/false.

Scenario	assign_port_supply_from_padpin false			assign_port_supply_from_padpin true		
	TopSupply	Crossover Generated	Checks On path	TopSupply	Crossover Generated	Checks On path
Top(VDD_TOP)----->PAD (VDD_PAD)	VDD_TOP	Yes	No ISO/LS checks performed.	VDD_PAD	Yes	No ISO/LS checks performed.
<ul style="list-style-type: none"> <li>• No ISO/LS/INV cell in the path</li> <li>• No ISO/LS Strategy in the path</li> <li>• No SPA/SRSN in the path</li> </ul>						
Top (VDD_TOP)(SPA/SRSN) - ----->PAD (VDD_PAD)	VDD_TOP	Yes	No ISO/LS checks performed.	VDD_PAD	Yes	No ISO/LS checks performed. The UPF_PORT_VOLTAGE violation is reported if SPA/SRSN is explicitly defined on the Top port.

<b>Scenario</b>	<b>assign_port_supply_from_padpin false</b>			<b>assign_port_supply_from_padpin true</b>		
	VDD_TOP	Yes	ISO/LS checks perform on this path	VDD_PAD	Yes	<ul style="list-style-type: none"> <li>ISO_STRATEG_Y_PAD</li> <li>LS_STRATEGY_PAD</li> <li>*_STRATEGY_REDUND</li> <li>*_STRATEGY_UNUSED</li> </ul>
Top (VDD_TOP)(ISO/LS Strategy) ----->PAD (VDD_PAD) • No ISO/LS/INV cell in the path						
Top (VDD_TOP)--- BUF/LS/ISO cell --->PAD (VDD_PAD)	VDD_TOP	Yes	ISO/LS/RAIL checks perform on this path. • ISO_INST_PAD • LS_INST_PAD	VDD_TOP	Yes	ISO/LS/RAIL checks perform on this path • ISO_INST_PAD • LS_INST_PAD • DESIGN_PAD_INTERNAL is reported if BUF/INV is present in the path.

### 5.2.11.3 Support for Pad pin Related checks for macro cells

The pad pin related checks for macro cells which have any pin with pin level attribute `is_pad:true` is enhanced. This is done to make it compatible with Library Compiler which supports `is_pad:true` for pins on macros that are connected to pad cells internally.

All functionality which works for normal pad cells also works for macro cells having pin with attribute `is_pad:true`.

## 5.2.12 Support for ISO/ELS Cells with No Enable Pin

VC LP enables you to use ISO/ELS cells with no enable pin, which are properly modeled with clamp value.

### 5.2.12.1 New Checks Introduced

#### 5.2.12.1.1 ISO\_ENABLE\_MISMATCH

The ISO\_ENABLE\_MISMATCH violation is reported when the strategy has no enable pin, but the cell has, or vice-versa.

The following is an example violation snippet of the ISO\_ENABLE\_MISMATCH Violation:

### 5.2.12.2 Impact on Existing Checks

#### 5.2.12.2.1 ISO\_CLAMP\_TYPE

ISO\_CLAMP\_INVERT violation tag renamed to ISO\_CLAMP\_TYPE to include this new scenario. The ISO\_CLAMP\_TYPE violation is reported when the strategy has no isolation enable, but the connected cell is an AND or OR type isolation gate, or vice-versa.

### 5.2.12.2.2 ISO\_MAP\_MISMATCH

The ISO\_MAP\_MISMATCH violation is enhanced to report violation when the strategy has no enable, but the cell has, or vice-versa.

### 5.2.13 Support For Fine Grain Power Switches

VC LP performs consistency checks between drivers of macros whose switched supplies are connected from the UPF. The PSW\_CONTROL\_TRACE violation is introduced for this support. VC LP performs the PSW\_CONTROL\_TRACE check only:

- ❖ For a 2 to 1 mux-like pg\_function.
- ❖ For a single pg pin input type fine grain PSW

The MACRO cells with function {IN} or {!IN} are allowed when analyzing the power switch enable signals paths. The following types of cells are allowed in this enable path:

1. Buffers or inverters
2. PSW or MACRO cells with function INPUT or NOT(INPUT).
3. Trace constraints settings of create\_trace\_constraint -trace\_through
4. UPF set\_port\_attribute feedthrough settings of:  

```
set_port_attribute -feedthrough
set_port_attribute -attribute function <input>
set_port_attribute -attribute function !<input>
```

The PSW\_CONTROL\_TRACE violation is reported when the connection of a signal in switch function and pg function is not consistent. This violation is reported at the UPF stage with severity *warning*. This tag is disabled by default. To enable it, use the configure\_tag -app LP -enable command.

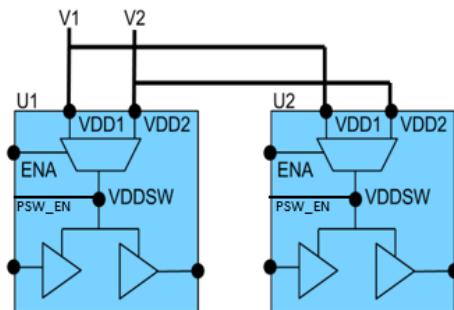
The PSW\_CONTROL\_TRACE violation performs the following checks:

- ❖ Consistency on PSW enable signal
- ❖ Match on the input supplies of the PSWs
- ❖ Match on the switch functions of the PSWs

#### Example

Consider the following design:

**Figure 5-74 Fine Grain PMUX**



And the following UPF snippet:

```

create_supply_net V_VIRT
connect_supply_net V_VIRT -ports {U1/VDDSW U2/VDDSW}
### or ###
set_equivalent -function_only -nets {U1/VDDSW U2/VDDSW}

```

And the following library cell:

```

cell (U1) {
    pg_pin (VDDSW) {
        pg_function : "(ENA VDD1)+(!ENA VDD2)";
        switch_function : "PSW_EN"; ...
    }
cell (U2) {
    pg_pin (VDDSW) {
        pg_function : "(ENA VDD2)+(!ENA VDD1)"; /* terms reversed */
        switch_function : "PSW_EN"

```

The PSW\_CONTROL\_TRACE violation is reported as the following matches are found.

U1/ENA connected to U2/ENA	MUX enable matching
U1/VDD1 is connected to U2/VDD2	Input supply matching
U2/VDD2 is connected to U2/VDD1	Input supply matching
U1/PSW_EN is connected to U2/PSW_EN	Switch function matching

The following is an example of the PSW\_CONTROL\_TRACE violation:

```

Tag : PSW_CONTROL_TRACE
Description : Connected macros output pins with same supply [UPFSupply] but do
not have same input supplies or their switch_function mismatch
Violation : LP:1
UPFSupply : tvdd10_csn
CellPinInfoList
  CellPinInfo
    OutputCellPin : U1/VDDSW
    InputCellPin : U1/ENA
  CellPinInfo
    OutputCellPin : U2/VDDSW
    InputCellPin : U2/ENA
ReasonCode : enable signals in pg_function do not match

```

## Example 2

```

create_supply_net V_VIRT
connect_supply_net V_VIRT -ports {U1/VDDSW U2/VDDSW}
### or ###
set_equivalent -function_only -nets {U1/VDDSW U2/VDDSW}

```

Consider the following lib Cell:

```

cell (i1) {
    pg_pin (VDDSW) {
        pg_function : "v1";
        switch_function : "SW_EN"; ...
cell (i2) {
    pg_pin (VDDSW) {
        pg_function : "v1";

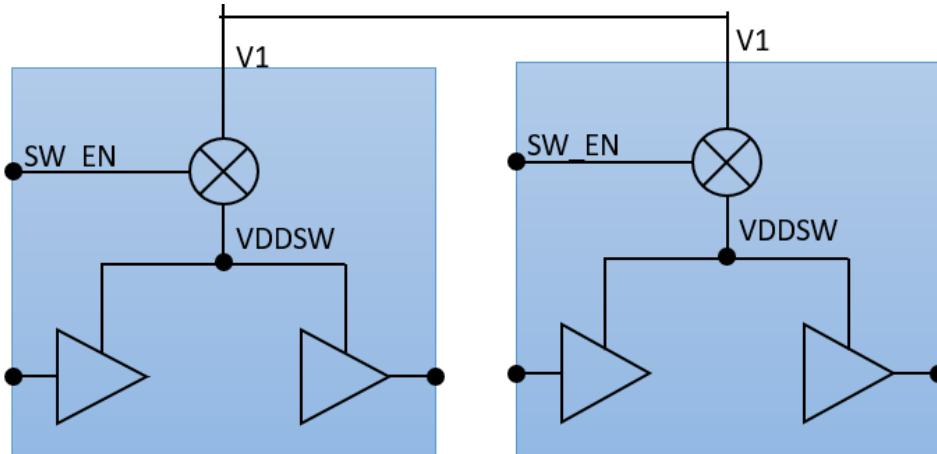
```

```
switch_function : "SW_EN" ...
```

If the VDDSW pin of an instance of i1 and the VDDSW pin of an instance of i2 are connected by a virtual net, then the following four traces will be checked.

- ❖ i1/SW\_EN connected to i2/SW\_EN
- ❖ i1/V1 is connected to i2/V1

If the connection of the enable signal in switch function and input supply are not in consistency, PSW\_CONTROL\_TRACE will be reported.



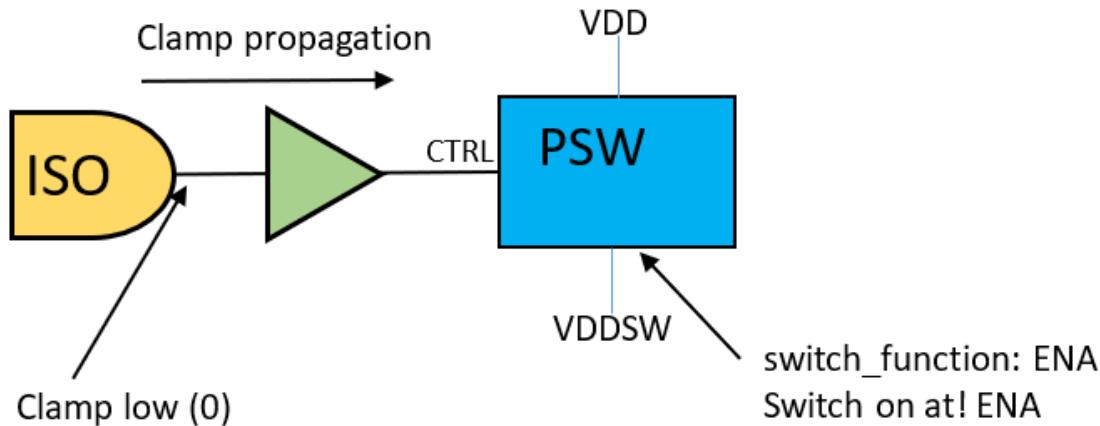
### 5.2.14 Support for Isolation Clamp Checks

VC LP support isolation clamp checks check whether isolation, PSW and retention control signals are properly isolated and functioning correctly.

Examples

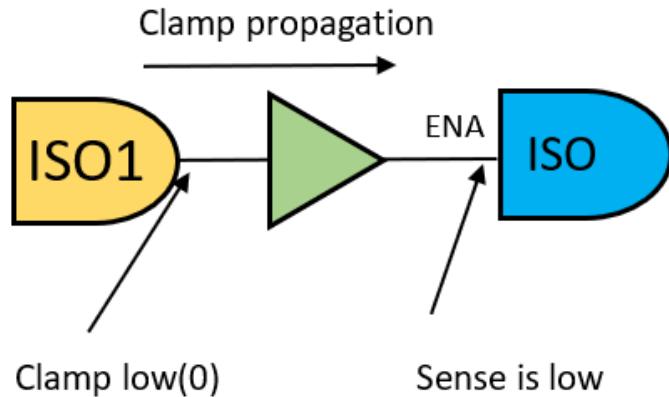
- ❖ Isolated PSW control

When isolation cell is clamping, PSW is on state (Transparent). That is not the correct behavior.



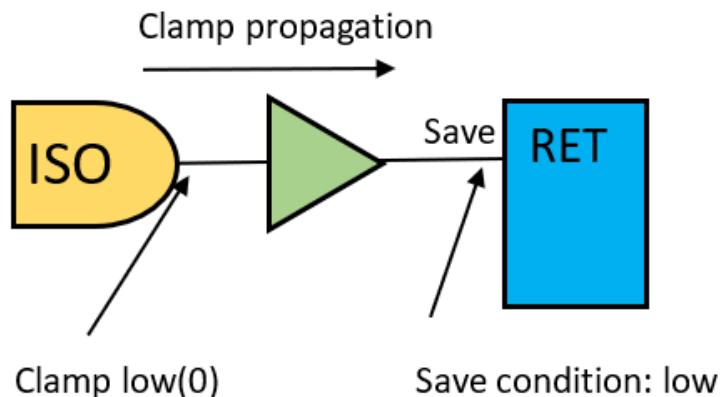
- ❖ Isolated ISO control

When isolation cell (ISO1) is clamping, ISO2 is in non-clamping state (Transparent). That is not the correct behavior.



- ❖ Isolated Retention control

When isolation cell is clamping, retention cell in save state. That is not the correct behavior. ISO clamp should not activate the retention cell states.



The following violations are introduced for this support:

1. PSW\_CONTROL\_CLAMP: For coarse grained PSW control end
2. PSW\_MACRO\_CLAMP: For macro PSW control end
3. ISO\_CONTROL\_CLAMP: For Isolation control end
4. ISO\_MACRO\_CLAMP: For macro cell with internal isolation
5. RET\_CONTROL\_CLAMP: For Retention control end (Save and Restore pins)

These tags are disabled by default and the severity is error. The violations are reported at the `check_lp` design stage.

- ❖ These tags are implemented to perform as crossover based checks. Hence, the clamp value does not propagate through combo logics.
- ❖ These tag are not depending with pst. Only clamp value and control port condition.

- ❖ New violations are performing for iso clamp values "1" and "0" only.

#### Note

- ❖ This checks are available under the VC-LP-FC-NT license.
- ❖ If PSW/ISO/RET control sink is not implemented (at RTL stage), user should enable generate\_upf\_ctrl\_signal\_crossover application variable to perform ISO clamp checks.

### 5.2.15 Support for AON Backplane Isolation Cells

The ISO\_SUPPLY\_SAME violation is introduced for AON backplane isolation cells. These backplane isolation cells are defined as the cells without a primary power pin and has only backup power pin. VC LP reports the new violation if the strategy supply of this cell is equivalent to its domain primary supply.

If it is power state table (pst) only equivalent, additional *PstEqui: True* debug field is reported. If they are electrically equivalent, this field is not reported. This is an Isolation family error violation checked at the design stage. This tag is disabled by default.

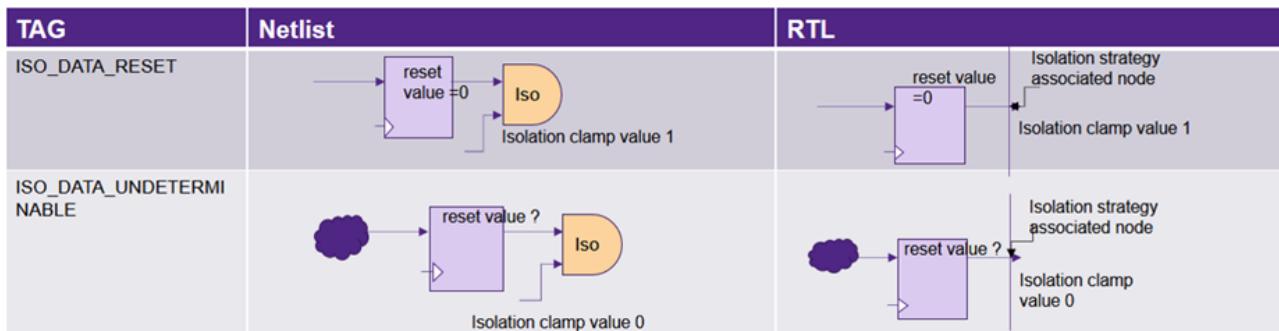
### 5.2.16 Support for ISO\_DATA\_RESET and ISO\_DATA\_UNDETERMINABLE

Incorrect clamp values with respect to reset values can cause functional issues in the design. Identifying these issues through simulation is time-consuming. The ISO\_DATA\_RESET and ISO\_DATA\_UNDETERMINABLE checks can help to identify these issues early and faster in the design flow. These checks are supported at the Netlist level and RTL level. Set the `lp_isodata_sequential_propagation` application variable to true to enable sequential constant propagation through sequential cells while performing these checks. By default, the `lp_isodata_sequential_propagation` application variable is set false.

For ISO\_DATA\_RESET and ISO\_DATA\_UNDETERMINABLE checks to run, the -family should have isolation and functional set for both RTL and netlist.

```
check_lp -stage all -family {functional isolation}
```

- ❖ ISO\_DATA\_RESET: This violation is reported when the cell clamp value does not match reset value of the driver.
- ❖ ISO\_DATA\_UNDETERMINABLE: This violation is reported when the isolation input data pin value after reset/set propagation of flops in the fan-in cone is undetermined.



In case of RTL, VC LP starts in the fan-in from the *policyAssocNode* and based on the set/reset values, the violations are reported. For the RTL checks, in debug fields instead of ISO cell instance, the *PolicyAssocNode* field is populated.

**Note**

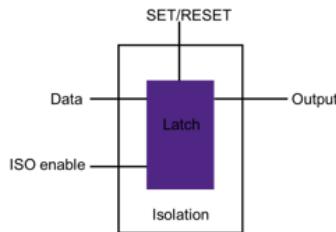
If there is an associated ISO cell for the policy, the RTL based check is ignored.

### 5.2.17 Support for ISO Latch With async set/reset

VC LP supports latch style isolation cells. With these type of isolation cells last value from the OFF domain can be passed to sink while isolating the source. These ISO latch cells can have asynchronous set/reset pins. Starting with this release, VC LP supports ISO latch cells with async set/reset

#### 5.2.17.1 Library Model

The following is a library model of a ISO latch cells with async set/reset.



The library modelling is similar to a latch cell, with enable pin as `iso_enable` and cell is with `is_isolation_cell` true. With set/reset triggered, the output is set to 1/0 regardless of the iso enable sense.

#### ISO latch with async reset: iso\_sense low and rst sense low

```

latch ("IQ","IQN") {
    enable : "G";
    data_in : "D";
    clear : "!RST";
    power_down_function : !VDD+VSS;
}
statetable ( " D   G   RST ", " Q   QN" ) {
    table : " -   -   L : - - : L   H, \
              -   L   H : - - : N   N, \
              H/L H   H : - - : H/L L/H";
}

```

#### ISO latch with async set: iso\_sense low and set sense low

```

latch ("IQ","IQN") {
    enable : "G";
    data_in : "D";
    preset : "!SET";
    power_down_function : !VDD+VSS;
}
statetable ( " D   G   SET ", " Q   QN" ) {
    table : " -   -   L : - - : H   L, \
              -   L   H : - - : N   N, \
              H/L H   H : - - : H/L L/H";
}

```

### 5.2.17.2 UPF Support

The following UPF command is supported:

```
set_isolation .... \
-clamp_value latch
[-async_set_reset {net_name <high | low>}]
[-async_clamp_value <0 | 1>]
```

#### Where

- ❖ `net_name` is the name of the set/reset control signal, and its sensitivity is specified by high/low
- ❖ `-async_clamp_value` specifies whether it is a set type or reset type latch 0 for reset and 1 for set.
- ❖ The `-async_set_reset` and `-async_clamp_value` options can be specified with `-update`. They can occur in any order.
- ❖ `-async_clamp_value` is optional, this can be specified later using `set_port_attribute SPA -ports {p1} -attribute {UPF_async_clamp_value 0 | 1}` similar to `UPF_clamp_value`
- ❖ These are not supported in LRM. These are Synopsys specific support.

#### Example

```
set_isolation ISO1 -domain uSUB1/PD_SUB -isolation_signal uSUB1/iso_en -isolation_sense
low -clamp_value latch -async_set_reset {uSUB1/w_set high} -async_clamp_value 1

set_isolation ISO4 -domain uSUB4/PD_SUB -isolation_signal uSUB4/iso_en -isolation_sense
low -clamp_value latch -async_set_reset {uSUB4/w_set low}

set_port_attributes -ports {uSUB4/in*} -attribute {UPF_async_clamp_value 0}
```

### 5.2.17.3 New LP violations

The following UPF violations are introduced:

UPF Check	Description
ISO_MAP_MISMATCH	This tag is already available, it is updated to cover new options
ISO_ASYNCCLAMP_MISSING	Strategy async set/reset is ignored
ISO_ASYNCCLAMP_OVERRIDE	Strategy async clamp value is overridden by set_port_attributes
ISO_ASYNC_PHASE	Isolation async signal is the same for multiple isolation strategies while their phases are different
ISO_ASYNC_UNCONN	Isolation async signal is unconnected
ISO_ASYNC_STATE	Isolation strategy supply on, but async set/reset supply off
ISO_ASYNC_VOLTDIFF	Isolation strategy supply voltage is different from async set/reset or its driver's supply voltage
ISO_STRATASYNC_GLITCH	Isolation async signal is not driven by a state signal

Design checks:

---

ISO_ASYNC_CONN	Strategy isolation async signal does not match isolation instance connection
ISO_ASYNC_INVERT	Strategy isolation async signal is opposite polarity from isolation instance
ISO_ASYNC_POLARITY	Isolation async signal reaches to instances with unknown polarity
ISO_ASYNC_UNDRIVEN	Isolation latch cell set/reset pin does not have any driver
ISO_ASYNC_GLITCH	Isolation latch cell set/reset pin is not driven by a state signal

---

In addition to this, tool will support all the checks similar to other control nets. Connectivity checks, architectural checks and so on.

#### 5.2.17.4 New Query Command Options and Attribute Support

The following new options are introduced in the following commands:

```
report_isolation_strategy <iso_strategy> -async_set_reset
get_isolation_strategies -async_set_reset <async_net>
get_isolation_strategy_elements <iso_strategy> -async_set_reset
```

The following new attributes are introduced:

**For isolation\_strategy**

async\_set\_reset, async\_sense, and async\_clamp\_value

**For crossover\_node**

SPA\_async\_clamp\_value

**Examples**

```
get_attribute [get_isolation_strategies uSUB1/PD_SUB/ISO1] async_set_reset
```

#### 5.2.17.5 Parser Messages Available for This Support

- ❖ *Error - [UPF\_ISOLATION\_OPTIONS\_MISMATCH] Set\_isolation options mismatch*  
*In set\_isolation command, with -async\_set\_reset, '-clamp\_value' must be specified, and the value must be 'latch'.*

**Example**

**Clamp value is not 'latch'**

```
set_isolation ISO1-domain PD1-isolation_signal iso_en-isolation_sense low
-clamp_value 1-async_set_reset{sethigh}-async_clamp_value 1
```

- ❖ *Error - [TCL\_OPT\_INVALIDONEOF] Invalid one-of value specified*  
*???????? Value 'posedge' for option '-async\_set\_reset' of command 'set\_isolation' is not valid. Specify one of: high, low.*

## Example

**Invalid sense for async set/reset net**

```
set_isolation ISO6-domain uSUB6/PD_SUB-isolation_signal uSUB6/iso_en-isolation_sense_low
-clamp_value latch -async_set_reset {uSUB6/w_set posedge}-async_clamp_value 1
```

- ❖ Error - [TCL\_OPTVAL\_INVALIDONEOF] Invalid one-of value specified  
???????? Failure reported from Tcl interpreter. Value 'x' for option '-async\_clamp\_value' is not valid. Specify one of: 0, 1.

## Example

**Invalid async clamp value**

```
set_isolation ISO6-domain uSUB6/PD_SUB-isolation_signal uSUB6/iso_en-isolation_sense_low
-clamp_value latch -async_set_reset {uSUB6/w_set high}-async_clamp_value x
```

- ❖ Error - [TCL\_OPT\_INVALIDONEOF] Invalid one-of value specified  
???????? Value 'x' for option '-attribute' of command 'set\_port\_attributes' is not valid. Specify one of: 0, 1.

## Example

**Invalid clmap value : SPA**

```
set_port_attributes -ports {uSUB6/in*} -attribute {UPF_async_clamp_value x}
```

- ❖ Error - [UPF\_OBJECT\_NOT\_FOUND\_ERROR] Object not found  
???????? Object 'xxx' of type 'LogicNet | LogicPort' specified with command option 'set\_isolation -async\_set\_reset' could not be resolved.

## Example

**async set/reset net not available**

```
set_isolation ISO6-domain uSUB6/PD_SUB-isolation_signal uSUB6/iso_en-isolation_sense_low
-clamp_value latch -async_set_reset {uSUB6/w_set1high}-async_clamp_value 1
```

- ❖ Error - [TCL\_INVALID\_ENUM\_VALUE] Invalid enum value  
???????? Enum value 'xxx' in option 'set\_isolation -async\_set\_reset' is invalid.

## Example

**Sense of async set/reset invalid**

```
set_isolation ISO6-domain uSUB6/PD_SUB-isolation_signal uSUB6/iso_en-isolation_sense_low
-clamp_value latch -async_set_reset {uSUB6/w_set1xxx}-async_clamp_value 1
```

- ❖ Error - [TCL\_OPT\_DEPENDS] Incorrect command options  
???????? '-async\_clamp\_value' requires that '-async\_set\_reset' must be specified in command 'set\_isolation'.

## Example

**async set/reset missing**

```
set_isolation ISO7-domain uSUB7/PD_SUB-isolation_signal uSUB7/iso_en-isolation_sense_low
-clamp_value latch -async_clamp_value 1
```

- ❖ Error - [UPF\_INVALID\_DUPLICATE\_OPTION] Invalid duplicate command option

## Example

### Duplicate updates

```
set_isolation ISO3-domain uSUB3/PD_SUB-isolation_signal uSUB3/iso_en-isolation_sense low-
clamp_value latch
set_isolation ISO3-domain uSUB3/PD_SUB-async_set_reset {uSUB3/w_set low}-update
set_isolation ISO3-domain uSUB3/PD_SUB-async_clamp_value 0-update
set_isolation ISO3-domain uSUB3/PD_SUB-async_set_reset {uSUB3/w_set high}-update
```

## 5.2.18 Support for Physical Variant Cells

VC LP supports physical variant cells. Physical variant cells are library cell variants which differ from the master cell only in mask color (small geometry). The timing data is the same for these variants.

In netlist or RTL+Lib cell flows, you can mark the cells as physical variants by using the library attribute “physical\_variant\_cells”

```
cell (MOD1) {
  physical_variant_cells : "MOD1_var1 MOD2_var2";
  -----
  -----
}
```

Previously, when `MOD1_var1` and `MOD2_var2` are instantiated, tool identified them as unresolved modules. Starting with this release, they can be mapped to the master cell `MOD2_var2`. The functional and timing behavior of the master cell are considered for the variant cells.

To enable physical variant cells in VC LP, set the `link_allow_physical_variant_cells` application variable, or the `-physicalVariantCells=1` Simon option.

Use the `physical_variant_cells` attribute for cells and lib cells to get the list of physical variant cells of the given cell or lib cell.

## Example

```
get_attribute [get_cells uSUB1/uBUF1] physical_variant_cells
get_object_name [get_lib_cells -of_objects [get_cells uSUB1/uBUF1]]
report_attributes -attributes physical_variant_cells [get_lib_cells
demo_std_12v_pg/MOD1] -class lib_cell
report_attributes -attributes physical_variant_cells [get_cells uSUB1/uBUF1] -class
cell
```

## 5.3 Advanced UPF Variations

### 5.3.1 Support for Restricting UPF Commands in DIUC

You can specify some constraints to not allow UPF command, UPF options, or enum values for UPF specified in VC LP. This means that, you can restrict users from specifying some options, which is not supported by other tools in the flow.

Use the `disallow_upf_command`, `disallow_upf_option`, and `disallow_upf_enum` commands to specify entire the UPF commands, UPF options, or enum values for UPF that does not need to be supported. You can specify a `upf_version` in which these commands need to be disallowed. If a `upf_version` is not specified, the commands will be disallowed in all the upf versions.

## Syntax

- ❖ disallow\_upf\_command <command> [-upf\_version <string>]
- ❖ disallow\_upf\_option <COMMAND2> <OPTION1> [-upf\_version <version> ]
- ❖ disallow\_upf\_enum <COMMAND3> <OPTION1> <ENUM1> [-upf\_version <version> ]

**Note**

The default upf version is considered to be upf 1.0.

**Examples**

- ❖ disallow\_upf\_command create\_supply\_set

Consider the following UPF:

```
create_supply_set SS\
    -function {power VDD}
```

If the command specified in the disallow\_upf\_command is used in the upf, the following error message will be reported.

*[Error] UPF\_LINT\_VIOLATED: Lint constraint is violated  
Command 'create\_supply\_set' is disallowed for any upf version.*

- ❖ disallow\_upf\_command create\_supply\_set -upf\_version 2.0

Consider the following UPF:

```
upf_version 2.0
create_supply_set SS\
    -function {power VDD}
```

If a upf\_version is mentioned in the disallow\_upf\_command, an error is reported only if the upf command is used under the specified upf\_version

*[Error] UPF\_LINT\_VIOLATED\_UPF\_VERSION: Lint constraint is violated for upf version  
Command 'create\_supply\_set' is disallowed for upf version '2.0'.*

- ❖ disallow\_upf\_option set\_level\_shifter domain

UPF:

```
set_level_shifter ls1 -domain top -applies_to inputs -rule low_to_high -location
self
```

If the command option specified in the disallow\_upf\_option is used in the UPF, the following error message is reported.

*[Error] UPF\_LINT\_VIOLATED: Lint constraint is violated  
Option '-domain' of command 'set\_level\_shifter' is disallowed for any upf version.*

- ❖ disallow\_upf\_option set\_level\_shifter domain -upf\_version 2.0

UPF:

```
upf_version 2.0
set_level_shifter ls1 -domain top -applies_to inputs -rule low_to_high -location
self
```

If a upf\_version is mentioned in the disallow\_upf\_option, an error is reported only if the upf command option is used under the specified upf\_version.

*[Error] UPF\_LINT\_VIOLATED\_UPF\_VERSION: Lint constraint is violated for upf version  
Option '-domain' of command 'set\_level\_shifter' is disallowed for upf version '2.0'.  
Please change related command/option/value.*

- ❖ `disallow_upf_enum set_level_shifter applies_to inputs`  
UPF:

```
set_level_shifter ls1 -domain top -applies_to inputs -rule low_to_high -location self
```

If the command option value specified in the `disallow_upf_enum` is used in the UPF, the following error message is reported.

*[Error] UPF\_LINT\_VIOLATED: Lint constraint is violated*

*Value 'inputs' for option '-applies\_to' of command 'set\_level\_shifter' is disallowed for any upf version.  
Please change related command/option/option value.*

- ❖ `disallow_upf_enum set_level_shifter applies_to inputs -upf_version 2.0`

UPF:

```
upf_version 2.0
set_level_shifter ls1 -domain top -applies_to inputs -rule low_to_high -location self
```

If a `upf_version` is mentioned in the `disallow_upf_enum` command, an error is reported only if the UPF command option is used under the specified `upf_version`.

*[Error] UPF\_LINT\_VIOLATED\_UPF\_VERSION: Lint constraint is violated for upf version*

*Value 'inputs' for option '-applies\_to' of command 'set\_level\_shifter' is disallowed for upf version '2.0'.  
Please change related command/option/option value.*

### 5.3.2 Support for `upf_set_once` Command

By default, when environment variables and Tcl variables are specified within the UPF files, and overridden by other UPF files subsequently loaded within the top level UPF, no indication was given that the original value of the variable was changed. You can specify protected variables for which the value will not be overridden by the subsequently setting the variable after initial definition using the `upf_set_once` command.

#### Use model

The `upf_set_once` command allows the user to specify a list of variables that needs to be protected from getting overridden within the UPF.

#### Syntax

```
upf_set_once < tcl variable names >
upf_set_once -env < environment variable names >
```

#### Example

```
upf_set_once -env ENV_VAR1
upf_set_once { TCL_VAR3 TCL_VAR4 }
```

If upf commands try to override the protected variables, the following warning message is reported:

Top.upf

```
set ::env(ENV_VAR1) val1b_top
load_upf sub.upf -scope sub1
```

sub.upf

```
set ::env(ENV_VAR1) val1b_hier
```

*[Warning] UPF\_SET\_ONCE\_VIOLATED: Rule upf\_set\_once violated*

*The upf\_set\_once variable 'env(ENV\_VAR1)' has been set to a different value 'val1b\_hier' than its initial value 'val1b\_top'. This command will be ignored.*

### 5.3.3 Support for Checking UPF Versions

When loading UPF files inside a parent upf, if there a mismatch between the upf versions, some inconsistencies may occur. For example, there could be differences in the way syntax is handled/allowed in different UPF versions, and using them together may be problematic. To prevent such issues, the `disallow_load_upf_version` command is introduced from this release onwards in DIUC flow that allows you to restrict loading upf files with specific upf version in a parent scope with a specific upf version.

#### Use model

```
disallow_load_upf_version -parent
{<upf version that could be in parent scope separated by space>}
-child <child upf version that needs to be disallowed >
-severity <ERROR/WARNING/INFO>
```

#### Example

tcl script

```
-----
disallow_load_upf_version -parent
{1.0 2.1} -child 3.0
```

```
top.upf
-----
upf_version 2.1
..
..

load_upf upf_3_0.upf
```

```
upf_3_0.upf
-----
upf_version 3.0
..
..
```

Here in the child upf, upf version is set as 3.0. It is loaded in the parent upf with version 1.0. Since the constraint is given as `disallow_load_upf_version -parent {1.0 2.1} -child 3.0` restricting version 3.0 inside 2.1, following parser error is flagged

*upf\_3\_0.upf:1: [Error] UPF\_LINT\_UPF\_VERSION\_DISALLOWED: Disallowed upf version with lint constraint  
The upf version '3.0' is disallowed in block when parent upf version is '2.1'.  
Please specify a version which does not violate the lint constraints specified.*



#### Note

The default value for severity is ERROR. You can change the severity of the parser message depending on the requirement.

### 5.3.4 Support for Allowing Specific enum Values in UPF Commands

Sometimes, you may have a requirement to disallow all the options in a command except few or allow only certain enum values, and disallow the rest of an option. In such cases, you should specify a number of disallow commands to achieve the requirement.

#### Example

```
set_isolation -location <>
```

It is needed to allow only location self. To achieve this using disallow commands, you should set the following two disallow commands

```
disallow_upf_enum set_isolation location parent -upf_version 2.0
disallow_upf_enum set_isolation location fanout -upf_version 2.0
```

Also, if in future, new options are introduced, you should modify the scripts to disallow those to allow only the required option.

To overcome this overhead, the allow\_\* commands are introduced to complement the disallow commands in such cases so that the user can achieve the requirement using one command.

```
allow_upf_enum set_isolation location self -upf_version 2.0
```

- ❖ allow\_upf\_enum

The allow\_upf\_enum command allows the specified enum of the specified option of the specified upf command per the version specified. If version is not specified, then the enum is allowed for the specified option of the specific command for all versions.

The specified command, option and enum is matched exactly, and only if an exact match is found, then the enum is allowed for the option for the command or an error is thrown.

#### Syntax

```
allow_upf_enum
<command_name>
<option_name>
<enum_option>
[-upf_version <version_val>]
```

#### Example

```
allow_upf_enum set_isolation location self -upf_version 2.0
```

- ❖ allow\_upf\_option

The allow\_upf\_option command allows the specified upf option of the specified upf command per the version specified. If version is not specified, then the option is allowed for the specific command for all versions.

The specified command and option is matched exactly and only if an exact match is found, then option is allowed for the command or an error is thrown.

#### Syntax

```
allow_upf_option
<command_name>
<option_name>
[-upf_version <version_val>]
```

#### Example

```
allow_upf_option set_isolation applies_to -upf_version 2.0
```

### 5.3.4.1 Precedence of allow\_upf\_option Over disallow\_upf\_option Commands

The following are the precedence of the `allow_upf_option` and `disallow_upf_option` commands.

- ❖ The `allow_upf_option` has precedence over `disallow_upf_option`. When the `allow_upf_option` has been specified for option, VC LP ignores the `disallow_upf_option` command for the same option.
- ❖ VC LP ignores the `disallow_upf_option` for any other option specified in a UPF command, when there is `allow_upf_option` specified for options in the same UPF command.

#### Example

Consider the following UPF snippet

```
upf_version 3.0
create_power_domain pd_always_on -exclude_elements {par_inst1}
allow/disallow command
allow_upf_option create_power_domain elements
allow_upf_option create_power_domain supply
allow_upf_option create_power_domain update
disallow_upf_option create_power_domain exclude_elements -upf_version 2.0
```

The following error messages are reported:

*top.upf:2: [Error] UPF\_LINT\_VIOLATED: Lint constraint is violated Option '-exclude\_elements' of command 'create\_power\_domain' is disallowed for any upf version. Please change related command/option/value.*

*top.upf:12: [Error] UPF\_LINT\_VIOLATED: Lint constraint is violated Option '-primary\_power\_net' of command 'set\_domain\_supply\_net' is disallowed for any upf version. Please change related command/option/value.*

In the example,

- ❖ For the same command (`create_power_domain`), the `allow_upf_option` takes precedence over the `disallow_upf_option` command.
- ❖ The `allow_upf_option` command are already mentioned for the `create_power_domain` command, and only options `-elements/-supply/-update` are allowed, therefore, for any other options of `create_power_domain`, the *[Warning] UPF\_LINT\_VIOLATED* is reported.
- ❖ If there is no `allow_upf_option` command for `create_power_domain`, VC LP checks the `disallow_upf_option create_power_domain exclude_elements -upf_version 2.0`, and the *[Warning] UPF\_LINT\_VIOLATED\_UPF\_VERSION* is reported.

### 5.3.5 Support for lp\_allow\_default\_version Application Variable

Synopsys tools support a feature for customers to create their own constraints for UPF commands by introducing `allow_upf_*` and `disallow_upf_*` commands. This support ideally requires setting `upf_version` in each UPF files. As customers start using newer UPF versions, setting `upf_version` in every UPF file is not feasible, therefore, VC LP has introduced the `lp_allow_default_version` application variable to setup the default UPF version.

By default, the application variable is set to 1.0. The values supported are: 1.0 2.0 2.1 3.0

Top UPF inherits the UPF version value from the `lp_allow_default_version` application variable when the `upf_version` is not set. Child UPFs loaded using the `load_upf` command inherits the parent UPF version when the `upf_version` is not set explicitly.

The UPF\_NO\_VERSION information message is introduced to notify which version is configured for the UPFs, which are not set with UPF version explicitly using "upf\_version"

### Example

Consider the following design hierarchy

DUT -> uSUB -> uMOD

The following Tcl snippet:

```
set_app_var lp_allow_default_version 3.0
disallow_upf_command set_isolation_control -upf_version 2.0
```

The following UPF in the design hierarchy:

DUT.upf	load_upf SUB.upf -scope uSUB set_isolation_control ISO -domain PD_DUT -isolation_signal iso_en -isolation_sense low
SUB.upf	upf_version 2.0 load_upf MOD.upf -scope uMOD set_isolation_control ISO1 -domain PD_SUB -isolation_signal iso_en -isolation_sense low
MOD.upf	set_isolation_control ISO2 -domain PD_MOD -isolation_signal iso_en -isolation_sense low

The following parser Messages are reported for this example

*MOD.upf:16: [Error] UPF\_LINT\_VIOLATED\_UPF\_VERSION: Lint constraint is violated for upf version Command 'set\_isolation\_control' is disallowed for upf version '2.0'. Please change related command option/value.*

*MOD.upf:0: [Info] UPF\_NO\_VERSION: upf\_version is not specified*

*File 'MOD.upf' doesn't specify upf\_version, assuming upf\_version '2.0', based on SUB.upf.*

*SUB.upf:25: [Error] UPF\_LINT\_VIOLATED\_UPF\_VERSION: Lint constraint is violated for upf version*

*Command 'set\_isolation\_control' is disallowed for upf version '2.0'.*

*Please change related command option/value.*

*DUT.upf:0: [Info] UPF\_NO\_VERSION: upf\_version is not specified*

*File 'DUT.upf' doesn't specify upf\_version, assuming upf\_version '3.0', based on appvar lp\_allow\_default\_version.*

### 5.3.6 Support for get\_upf\_scope Command

You can identify which UPFs are loaded at which scope using the `get_upf_scope` Tcl command.

Assume you have the hierarchical UPF:

Top.upf :

```
load_upf ABC.upf -scope ABC
load_upf BCD.upf -scope BCD
load_upf BCD.upf -scope CDE
```

The `get_upf_scope` Tcl command is introduced to make it possible to query the UPFs that are loaded at a relevant scope, and what are the relevant scopes for a given UPF.

### Syntax

```
get_upf_scope -< scope >/< upf >
```

### Examples

The command can be used for the mentioned hierarchical UPF as below.

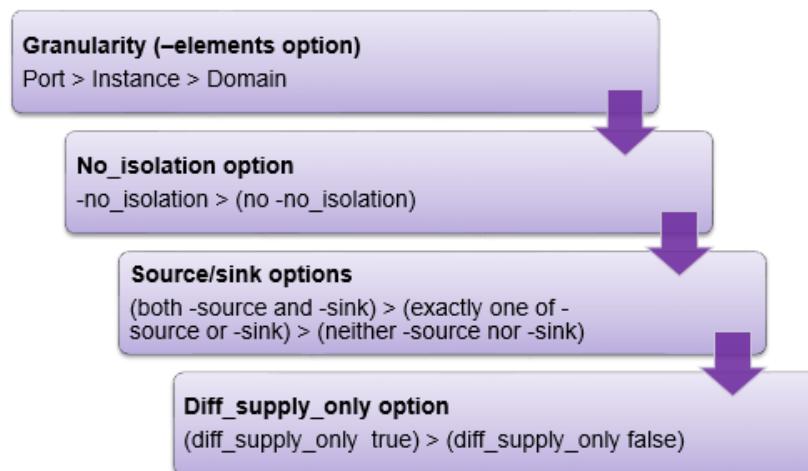
- ❖ `get_upf_scope -upf ABC.upf`  
`{"ABC"}`
- ❖ `get_upf_scope -upf BCD.upf`  
`{"BCD", "CDE"}`
- ❖ `get_upf_scope -scope ABC`  
`{"ABC.upf"}`
- ❖ `get_upf_scope -scope BCD`  
`{"BCD.upf"}`

### 5.3.7 Precedence for Choosing ISO Strategies

VC LP resolves the final ISO strategy that is applicable to a particular port, when there are multiple strategies are applicable as described in the following section.

Among multiple strategies present on a particular port, the most specific strategy gets associated with that port. VC LP analyzes the strategies based on the following filters and chooses the strategy based on the set precedences. The filters are mentioned in the decreasing order of precedence as shown in [Figure 5-75](#):

**Figure 5-75 Precedence for Choosing ISO Strategies**



- ❖ *Granularity based filter: [-elements option containing Port > Instance > Domain]*

Strategies written by specifying ports in the `-elements` of the strategy has higher priority than the strategies written by specifying the instance name in the `-elements` option. Strategies that do not have `-elements` option has the least precedence.

- ❖ *no\_isolation based filter: [-no\_isolation > (no -no\_isolation)]*

- Strategies written with `-no_isolation` gets higher priority over the strategies that do not have it.
- ❖ *The source/sink option based filter: [(both -source and -sink) > (exactly one of the -source or -sink) > (neither -source nor -sink)]*
- Strategies having both the `-source` and `-sink` option gets higher priority over which is having exactly one that is, either `-source` or `-sink`. Strategies which are not having any of the option `-source/-sink` gets the least precedence.
- ❖ *The diff\_supply\_only based filter: [(-diff\_supply\_only true) > (-diff\_supply\_only false)]*
- Strategies with `-diff_supply_only true` gets higher precedence over the ones having value false.

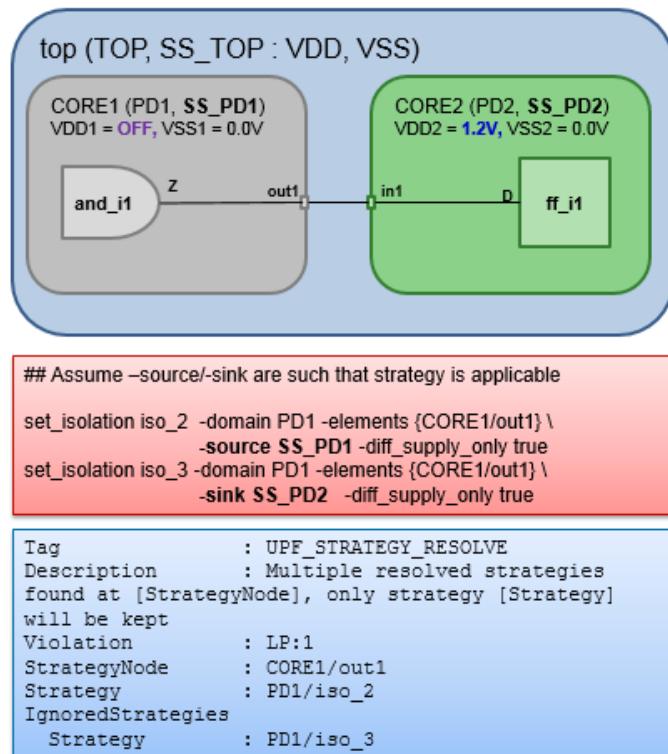
If there are two strategies with equal precedence after applying the precedence rules, VC LP reports the existing UPF\_STRATEGY\_RESOLVE violation and associates the first strategy written in the UPF. This violation is reported during the `check_lp -stage` UPF, with severity warning, and this violation is enabled by default.

### Example

Consider the following crossover (PD1 and PD2 are power domains) as shown in [Figure 5-76](#):

```
CORE1/and_i1/Z (PD1, OFF) --> CORE1/out1 (PD1, Boundary port) --> CORE2/in1 (PD2, Boundary port) --> CORE2/and_i1/A (PD2, ON)
```

**Figure 5-76 Example Design for UPF\_STRATEGY\_RESOLVE**



The following strategies are written in the decreasing order in which they apply on boundary port `CORE1/out1`. The number against each strategy indicates its precedence level.

---

```
1) set_isolation iso_1 -domain PD1 -elements {CORE1/out1} -no_isolation -source SS_PD1  
-sink SS_PD2  
2) set_isolation iso_2 -domain PD1 -elements {CORE1/out1} -no_isolation -source SS_PD1  
-diff_supply_only true  
2) set_isolation iso_3 -domain PD1 -elements {CORE1/out1} -no_isolation -sink SS_PD2 -  
diff_supply_only true  
3) set_isolation iso_4 -domain PD1 -elements {CORE1/out1} -no_isolation -source SS_PD1  
3) set_isolation iso_5 -domain PD1 -elements {CORE1/out1} -no_isolation -sink SS_PD2  
4) set_isolation iso_6 -domain PD1 -elements {CORE1/out1} -no_isolation -  
diff_supply_only true  
5) set_isolation iso_7 -domain PD1 -elements {CORE1/out1} -no_isolation  
6) set_isolation iso_8 -domain PD1 -elements {CORE1/out1} -source SS_PD1 -sink SS_PD2  
7) set_isolation iso_9 -domain PD1 -elements {CORE1/out1} -source SS_PD1 -  
diff_supply_only true  
7) set_isolation iso_10 -domain PD1 -elements {CORE1/out1} -sink SS_PD2 -  
diff_supply_only true  
8) set_isolation iso_11 -domain PD1 -elements {CORE1/out1} -source SS_PD1  
8) set_isolation iso_12 -domain PD1 -elements {CORE1/out1} -sink SS_PD2  
9) set_isolation iso_13 -domain PD1 -elements {CORE1/out1} -diff_supply_only true  
10) set_isolation iso_14 -domain PD1 -elements {CORE1/out1}
```

---




---

```

11) set_isolation iso_15 -domain PD1 -elements {CORE1} -no_isolation -source SS_PD1 -sink
SS_PD2
12) set_isolation iso_16 -domain PD1 -elements {CORE1} -no_isolation -source SS_PD1 -
diff_supply_only true
12) set_isolation iso_17 -domain PD1 -elements {CORE1} -no_isolation -sink SS_PD2 - -
diff_supply_only true
13) set_isolation iso_18 -domain PD1 -elements {CORE1} -no_isolation -source SS_PD1
13) set_isolation iso_19 -domain PD1 -elements {CORE1} -no_isolation -sink SS_PD2

14) set_isolation iso_20 -domain PD1 -elements {CORE1} -no_isolation -diff_supply_only
true
15) set_isolation iso_21 -domain PD1 -elements {CORE1} -no_isolation
16) set_isolation iso_22 -domain PD1 -elements {CORE1} -source SS_PD1 -sink SS_PD2
17) set_isolation iso_23 -domain PD1 -elements {CORE1} -source SS_PD1 -diff_supply_only
true
17) set_isolation iso_24 -domain PD1 -elements {CORE1} -sink SS_PD2 - -diff_supply_only
true
18) set_isolation iso_25 -domain PD1 -elements {CORE1} -source SS_PD1
18) set_isolation iso_26 -domain PD1 -elements {CORE1} -sink SS_PD2
19) set_isolation iso_27 -domain PD1 -elements {CORE1} -diff_supply_only true
20) set_isolation iso_28 -domain PD1 -elements {CORE1}
21) set_isolation iso_29 -domain PD1 -no_isolation -source SS_PD1 -sink SS_PD2
22) set_isolation iso_30 -domain PD1 -no_isolation -source SS_PD1 -diff_supply_only true
22) set_isolation iso_31 -domain PD1 -no_isolation -sink SS_PD2 - -diff_supply_only true
23) set_isolation iso_32 -domain PD1 -no_isolation -source SS_PD1
23) set_isolation iso_33 -domain PD1 -no_isolation -sink SS_PD2
24) set_isolation iso_34 -domain PD1 -no_isolation -diff_supply_only true
25) set_isolation iso_35 -domain PD1 -no_isolation
26) set_isolation iso_36 -domain PD1 -source SS_PD1 -sink SS_PD2
27) set_isolation iso_37 -domain PD1 -source SS_PD1 -diff_supply_only true
27) set_isolation iso_38 -domain PD1 -sink SS_PD2 - -diff_supply_only true
28) set_isolation iso_39 -domain PD1 -source SS_PD1
28) set_isolation iso_40 -domain PD1 -sink SS_PD2
29) set_isolation iso_41 -domain PD1 -diff_supply_only true
30) set_isolation iso_42 -domain PD1

```

---

After applying the precedence rules, if there exists multiple strategies with the same precedence level on particular boundary port, the UPF\_STRATEGY\_RESOLVE violation is reported. Consider a case, if only two strategies iso\_2 and iso\_3 mentioned above are present in UPF, then VC LP reports the UPF\_STRATEGY\_RESOLVE violation as follows:

Tag	:	UPF_STRATEGY_RESOLVE
Description	:	Multiple resolved strategies found at [StrategyNode], only
strategy [Strategy]	will be kept	
Violation	:	LP:1
StrategyNode	:	CORE1/out1
Strategy	:	PD1/iso_2

```
IgnoredStrategies
Strategy      : PD1/iso_3
```

Notes:

- ❖ Isolation strategy is having argument `-applies_to`, which can have values: inputs, outputs or both. The `-applies_to` option does not play any role in the isolation precedence. It only helps to filter out ports whether strategy has to be applied on inputs or outputs or both. The following two strategy have the same precedence and VC LP reports the UPF\_STRATEGY\_RESOLVE violation:

```
set_isolation iso_1 -domain PD1 -elements {CORE1/out1}
set_isolation iso_2 -domain PD1 -elements {CORE1/out1} applies_to outputs
```

- ❖ Other flags like `-location`, `-isolation_supply_set`, and so on that are supported for the isolation strategy do not play a role in isolation precedence.

### 5.3.8 Retention Strategy Precedence

The retention strategy policy association precedence rule in order of priority is:

1. Element based with `no_retention`
2. Element based
3. Domain based with `no_retention`
4. Domain Based

If the multiple same priority strategies are present, then the strategy which is provided first is given the priority. The RET\_STRATEGY\_MULTIPLE is reported only when two same precedence strategies are applied on the same element, and one is ignored in the following scenarios.

1. Both retention strategies are domain based applied on the same element.

**Example**

```
set_retention ret_PD1 -domain PD1
set_retention ret_PD11 -domain PD1
```

The RET\_STRATEGY\_MULTIPLE violation is reported as VC LP gives priority to lesser line number in case of same priority, and the `RET_PD1` is applied.

2. Both retention strategies are domain based with `no_retention` applied on the same element.

**Example**

```
set_retention ret_PD1 -domain PD1 -no_retention
set_retention ret_PD11 -domain PD1 -no_retention
```

The RET\_STRATEGY\_MULTIPLE violation is reported as VC LP gives priority to lesser line number in case of same priority, and the `RET_PD1` is applied.

3. Both retention strategies are Instance based strategies applied on same element.

**Example**

```
set_retention ret_PD1 -domain PD1 -elements CORE1/ff_i11
set_retention ret_PD11 -domain PD1 -elements CORE1/ff_i11
```

The RET\_STRATEGY\_MULTIPLE violation is reported as VC LP gives priority to lesser line number in case of same priority, and the `RET_PD1` is applied.

4. Both retention strategies are Instance Based and `no_retention` as well, applied on same element.

**Example**

```
set_retention ret_PD1 -domain PD1 -elements CORE1/ff_i11 -no_retention
set_retention ret_PD11 -domain PD1 -elements CORE1/ff_i11 -no_retention
```

The RET\_STRATEGY\_MULTIPLE violation is reported as VC LP gives priority to lesser line number in case of same priority, and the *RET\_PD1* is applied.

### Example

Consider the following design.

```
module top (in1,save1,restore1,out1,out2,out3);
core1 CORE1 (.in1(in1),.save1(save1),.restore1(restore1),.out1(w1), .out2(out2));
core2 CORE2 (.in1(w1),.save1(save1),.restore1(restore1),.out1(out1));
DMFD2 ff_top (.D(w1),.Q(out3));
endmodule

module core1(in1,save1,restore1,out1,out2);
assign w1 = out1;
DMFD2 ff_1 (.D(in1),.Q(out1));
DMFD2 ff_2 (.D(w1),.Q(out2));
endmodule

module core2(in1,save1,restore1,out1);
DMFD2 ff_i21 (.D(in1),.Q(out1));
endmodule
```

And the following UPF snippet.

```
create_power_domain TOP
create_power_domain PD1 -elements {CORE1}
create_power_domain PD2 -elements {CORE2}

set_retention ret_1 -domain PD1
set_retention ret_2 -domain PD1
set_retention ret_3 -domain PD1 -no_retention
set_retention ret_4 -domain PD1 -no_retention
set_retention ret_5 -domain PD1 -elements {CORE1/ff_1}
set_retention ret_6 -domain PD1 -elements {CORE1/ff_1}
```

The following are the resolved elements (per policy):

*Policy Name:PD1/ret\_5: CORE1/ff\_1*

*Policy Name:PD1/ret\_3: CORE1/ff\_2*

Two RET\_STRATEGY\_MULTIPLE violations are reported for this scenario. One is for *PD1/ret\_4* and the other one is for *PD1/ret\_6*.

### 5.3.9 Support for reference\_only Attribute

VC LP supports `reference_only` attribute is supported in the `set_design_attribute` command. This attribute enables you to mark supply nets and sets as reference only. This support limits the usage of supplies marked as reference only. The supply net name or supply set name can be given as a value to the attribute.

#### Syntax

```
set_design_attribute -element . -attribute reference_only\ {<supply_name>}
```

### Example

```
set_design_attribute -element . -attribute reference_only {SS2'}
```

### 5.3.9.1 Parser Error Messages Supported

The following parser messages are introduced for this support:

- ❖ If a supply\_set/net used in the reference\_only is not a simple name (non-hierarchical, non supply set handles), the following error message is reported:

*[Error] UPF\_REFERENCE\_ONLY\_SUPPLY\_NON\_SIMPLE\_NAME: Reference only supply only supports simple name*

- ❖ If the UPF command uses -elements {<element name other than .>}, the following error message is reported:

*[Error] UPF\_REFERENCE\_ONLY\_SUPPLY\_INVALID\_ELEMENTS: Reference only supply should use -elements {.}*

### 5.3.9.2 Support for Reference Supplies in set\_equivalent Command

Currently, the set\_equivalent command is allowed only for supplies connected to top ports or macros. These rules for set\_equivalent will be relaxed when set\_equivalent command is being used to associate reference only supplies with a real supply. The rules will be relaxed only in case where all of the supplies, or all but one of the supplies in set\_equivalent command are reference only.

### 5.3.9.3 Limitation With Support for reference\_only Attribute

The support on errors when reference\_only supplies are used in UPF command other than path based strategies like supply\_sets definitions and set\_port\_attributes are currently under validation.

### 5.3.10 Support for lower\_domain\_boundary Attribute

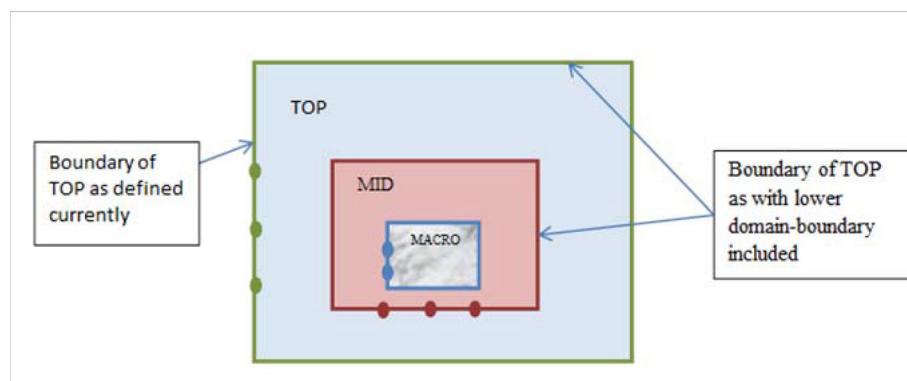
VC LP supports the lower\_domain\_boundary design attribute of the set\_port\_attributes UPF command.

```
set_design_attributes
-elements {<element_name>}
-attribute lower_domain_boundary <true/false>
```

The lower\_domain\_boundary design attribute:

- ❖ Must be written be in lower-case and is case-sensitive.
- ❖ Takes string values true (or TRUE) and false (or FALSE). The string value should either be all in lower-case or upper-case.
- ❖ Is disabled by default at the top scope.

The boundary of the power domain is defined as the logical boundary of the root cells of the power domain. The lower\_domain\_boundary attribute enables you to also include the lower boundary of a power domain.

**Figure 5-77 Lower Domain Boundary of Power Domain**

You can enable this support for the whole design using the following command at the top scope:

```
set_design_attributes -elements { . } -attribute lower_domain_boundary true
```

This feature is enabled for all the domains scoped at and below the top scope. Once enabled at the top most scope, the feature is enabled for the entire design.

### 5.3.10.1 Rules for setting lower\_domain\_boundary attribute

Adhere to the following rules while setting the `lower_domain_boundary` attribute to ensure that the same semantic is applied to the entire design and there is no mixing of blocks that have this feature enabled with blocks that have this feature disabled.

- ❖ The `lower_domain_boundary` attribute must set on the top most scope, only if there are no power domains in the design and the specified value matches with any explicit setting elsewhere in the design.
- ❖ The `lower_domain_boundary` attribute must set on a lower scope, only if there are no power domains at and below that lower scope and the specified value matches with the default or explicit setting on the top most scope.
- ❖ The `lower_domain_boundary` attribute can be changed on the top most scope, only if there are no power domains in the design and there are no explicit settings elsewhere in the design.

### 5.3.10.2 Impact on the UPF Commands

The following `set_isolation` and `set_level_shifter` commands have the following impact when the lower domain boundary feature is turned on for the design.

- ❖ The `-location parent/fanout` is not allowed in the `set_isolation_control` command.
- ❖ The `-location parent` is not allowed in the `set_level_shifter` command
- ❖ The `-applies_to` option of `set_level_shifter` and `set_isolation` commands considers lower boundary pins of the domain while filtering the objects or domain-level strategies.
- ❖ The `-elements` option identifies the specified pin and port elements as the lower boundary interface of the specified domain.

### 5.3.10.3 Impact on Policy Associations

- ❖ Isolation Policy Association
  - ◆ Case 1: Isolation policy with `-applies_to` without `-elements`

Any isolation policy that applies to the input pins/ports of the domain also applies to the output pins/ports of the root cells of the domains nested inside it. Any isolation policy that applies to the output pins/port of the domain also applies to the input pins/ports of the root cells of domains nested inside it.

- ◆ Case 2: Isolation policy with -elements

Isolation policies that specifically choose the pins to which they are applied do not change. They continue to apply to the pins/ports in the -elements option of the policy definition. However, the policies refer to the pins on the lower boundary of the domain in their -elements option. This will enable you to target an isolation policy specifically to the lower boundary.

- ◆ Case 3: Multiple isolation policies on the same boundary pin for back-to-back isolation cells

This case enables you to specify isolation policies such that two isolation cells can be inserted on the same power domain boundary pin. One of these cells will be inserted in the power domain and the other will be inserted in its surrounding domain.

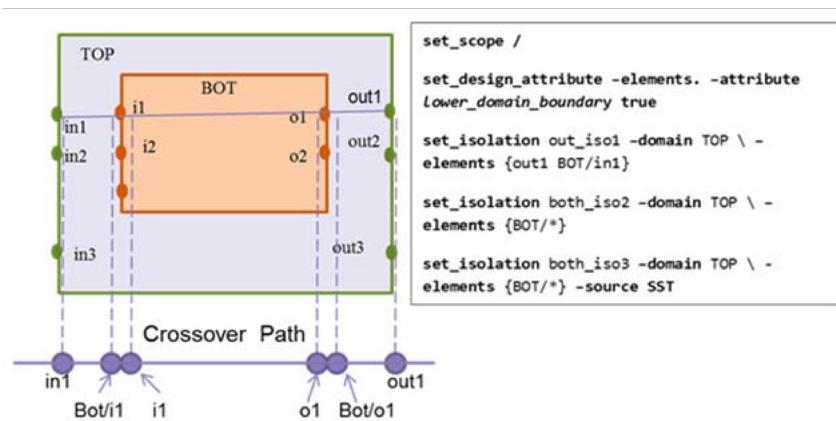
- ❖ Level Shifter Policy Association

The level-shifter policy association is same as the isolation policy association.

- ❖ Crossover Creation

For crossover tree creation, if the lower\_boundary\_domain is used, then for every intermediate boundary node, two policy\_assoc nodes are created; one for loconn and one for hiconn.

**Figure 5-78 Cross-Over Creation**



As shown in [Figure 5-78](#), for intermediate boundary ports, two policy association crossover nodes are created, one for hiconn and another for loconn.

### 5.3.11 VC LP Simplifies PST Using Voltage Triplets

Power State Tables (PST) can be large in size and very complex depending on the design and this can result in runtime performance issues. In such cases, VC LP describes port states and creates corresponding PST states for each process corner. This is achieved by grouping port states of different corners into voltage triplets for each supply port. This reduces the port size to one third of the original size and also enables you to reduce the resultant PST size thus improving the performance of the LP checks.

The following is an example for a simple PST:

```
add_port_state VDD1 -state {HV_MIN 0.9}
```

```

          -state {HV_NOM 1.0}
          -state {HV_MAX 1.1}
add_port_state VDD2 -state {HV_MIN 1.0}
          -state {HV_NOM 1.1}
          -state {HV_MAX 1.2}
add_port_state VSS -state {ON 0.0}

create_pst      PST           -supplies      {VDD1      VDD2      VSS}
add_pst_state  HV_STATE_MIN -pst PST -state { HV_MIN   HV_MIN   ON}
add_pst_state  HV_STATE_NOM -pst PST -state { HV_NOM   HV_NOM   ON}
add_pst_state  HV_STATE_MAX -pst PST -state { HV_MAX   HV_MAX   ON}

```

This PST can be optimized by using the following triplets:

```

add_port_state VDD1 -state {HV 0.9 1.0 1.1}
add_port_state VDD2 -state {HV 1.0 1.1 1.2}
add_port_state VSS -state {ON 0.0}

```

```

create_pst      PST           -supplies      {VDD1      VDD2      VSS}
add_pst_state  HV_STATE     -pst PST -state {HV      HV      ON}

```

By default, voltage triplets are not related and the actual voltage can take on any value within the range. VC LP checks all combinations of the best and worst case voltages when comparing two voltage triplets.

To treat triplets as voltage ranges and compare max to max, nom to nom, and min to min, the following command is used:

```
set_design_attributes -elements { . } -attribute correlated_supply_group "{VDD1 VDD2}"
```

## Use Model

In your UPF file, you can group supply nets into supply groups using the `set_design_attributes` command with the following syntax:

```
set_design_attributes -elements { . } -attribute correlated_supply_group
"<supply_group> <supply_group> ... <supply_group> "
```

where

```
<supply_group> = { <supply_net> <supply_net> ... <supply_net> }
```

Supply groups are merged if they contain the same supply net.

## Examples

- ❖ To describe two correlated supply groups:

```
set_design_attributes -elements { . } -attribute correlated_supply_group "{VDD1
VDD2} {VDD3 VDD4}"
```

- ❖ To turn on correlated voltage range for all supply nets in the design, you can group all supply nets together using '\*' as illustrated in the following example:

```
set_design_attributes -elements { . } -attribute correlated_supply_group "*"
```

- ❖ A supply net appears in more than one group (VDD1):

```
set_design_attributes -elements { . } -attribute correlated_supply_group "{VDD1 VDD2} {VDD1 VDD3}"
```

is equivalent to:

```
set_design_attributes -elements { . } -attribute correlated_supply_group "{VDD1 VDD2 VDD3}"
```

- ❖ Two supply nets belong to the same netgroup (VDD1 and VDD1'):

```
set_design_attributes -elements { . }
-attribute correlated_supply_group "{VDD1 VDD2} {VDD1' VDD3}"
```

is equivalent to:

```
set_design_attributes -elements { . }
-attribute correlated_supply_group "{VDD1 VDD2 VDD3}"
```

- ❖ A supply net is updated to another supply net in another group (VDD1 and SS1.power):

```
set_design_attributes -elements { . }
-attribute correlated_supply_group "{VDD1 VDD2} {SS1.power VDD3}"
create_supply_set SS1 -function {power VDD1} -
```

is equivalent to:

```
set_design_attributes -elements { . }
-attribute correlated_supply_group "{VDD1 VDD2 VDD3}"
```

### 5.3.12 Support for Logic Nets in –elements option of set\_design\_attributes command

Logic nets are allowed in –elements option of set\_design\_attributes command. Also, compatibility check are performed between logic net and the value of the –attribute option of set\_design\_attributes command.

Example

Consider LVDD1 is a logic net:

- ❖ set\_design\_attributes command with known SNPS attribute

```
set_design_attributes -elements { LVDD1 } -attribute
conservative_diff_supply_only_isolation true
```

The UPF\_ATTR\_INVALID\_OBJECT is reported as the attribute value is not compatible for logic net. Here logic net is identified as a valid value for –element.

- ❖ set\_design\_attributes with user defined attribute

```
set_design_attributes -elements { LVDD1 } -attribute ud_attribute true
```

The TCL\_INV\_ATTR message is reported as the attribute value is not recognized by tool. But still the logic net is identified as a valid value for –element.

### 5.3.13 Support for the terminal\_boundary Attribute

When block level designs are developed, external boundary conditions may be captured using set\_port\_attributes or set\_related\_supply\_net of the block boundary ports. When the block is integrated in a SoC top, typically verification happens in the full flat design. The verification tool does not have context of which blocks are implemented stand alone and then integrated. This information can be passed on to verification tools by marking the block boundaries as terminal\_boundary using

`set_design_attributes` in UPF. Using this attribute, verification tools can cross check and verify if the block implementation level assumptions of boundary conditions are over constrained or under constrained or at mismatch with the SoC level considerations.

When this attribute is set to `true` on an instance, the boundary ports of this instance become terminal points for any global net. Any hierarchical instance which has this attribute, is expected to be integrated into a bigger design.

## Use Model

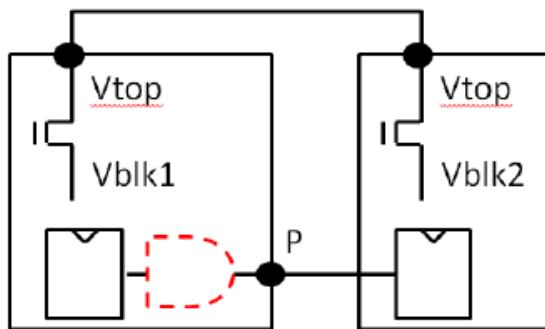
```
create_power_domain PDA -include_scope
set_design_attribute -elements { . } -attribute terminal_boundary true
```

The default value is `false`. It can be specified on any hierarchical instance which is a power domain boundary.

The following checks are introduced for the `terminal_boundary` option:

- ❖ A `driver/receiver_supply` attribute or `set_related_supply_net` on the primary input/output of a block is required where the `terminal_boundary` attribute is specified, else the `TERMINALBDRY_PORT_UNCONSTRAINED` message is issued for the `read_upf` command and the rest of checks are skipped.
- ❖ Based on the `driver/receiver_supply`, the constraints path is either marked as Overconstrained or Underconstrained.
  - ◆ *Overconstrained:* Consider the following example:

**Figure 5-79 Terminal Boundary Overconstrained Example**

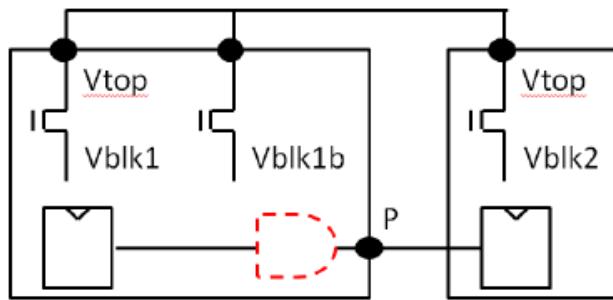


UPF for blk1:

```
create_power_switch -input Vtop -output Vblk1
set_domain_supply_net Vblk1
set_port_attributes P receiver_supply Vtop
set_isolation iblk -source Vblk1 -sink Vtop
-isolation_supply Vtop -location self
```

All the four combinations of `Vblk1 on, off` and `Vblk2 on, off` exists. The block isolation is always safe. In this case, no message is issued about the mismatch between the attribute supply and the real receiver supply as painful noise, as this is both expected and safe.

- ◆ *Underconstrained:* Consider the following example:

**Figure 5-80 Terminal Boundary Underconstrained Example**

```
UPF for blk1:
create_power_switch -input Vtop -output Vblk1
set_domain_supply_net Vblk1
set_port_attributes P receiver_supply Vblk1b
set_isolation iblk -source Vblk1 -sink Vblk1b \
-isolation_supply Vblk1b -location self
```

The system power state *Vblk1 off, Vblk1b on, Vblk2 on* demonstrates correct isolation. If the state *Vblk1 off, Vblk1b off, Vblk2 on* exists, this is an electrical error which cannot be detected at the block level. The SoC tools uses the attribute supply for this case, VC LP flags an error that the implementation result is passed as a block level implementation check, but the result is incorrect electrically at the SoC level.

The UPF\_SPADRIVER\_STATE/UPF\_SPARECEIVER\_STATE and UPF\_SPASUPPLY\_VOLTAGE violations are issued for the underconstrained cases.

The UPF\_SPASUPPLY\_CONN violation is also reported for the terminal\_boundary attribute. If all these violation are reported because of terminal\_boundary then debug field *TerminalBdry : true* is populated.

### The UPF\_SPARECEIVER\_STATE Violation Example

```
Tag          : UPF_SPARECEIVER_STATE
Description   : Supply of actual receiver is on, but SPA receiver supply
off for SPA constraint on [ConstraintNode]
Violation    : LP:1
ConstraintNode: P
ReceiverNode  : blk2/I1/A
ConstraintInfo
  PowerNet
    NetName   : Vblk1b
    NetType    : UPF
    PowerMethod: FROM_UPF_RECEIVER_SUPPLY
  ReceiverInfo
    PowerNet
      NetName   : Vtop
      NetType    : Design/UPF
      PowerMethod: FROM_UPF_POWER_DOMAIN
    TerminalBdry: true
    Internal   : True
  States
    State     : top_pst/s1
  UPFCommand
    Command   : set_port_attributes
    FileName  : constraint.upf
    LineNumber: 2
```



Similarly, the UPF\_SPADRIVER\_STATE violation is reported for the SPA -driver\_supply constraint.

### The UPF\_SPASUPPLY\_VOLTAGE Violation Example

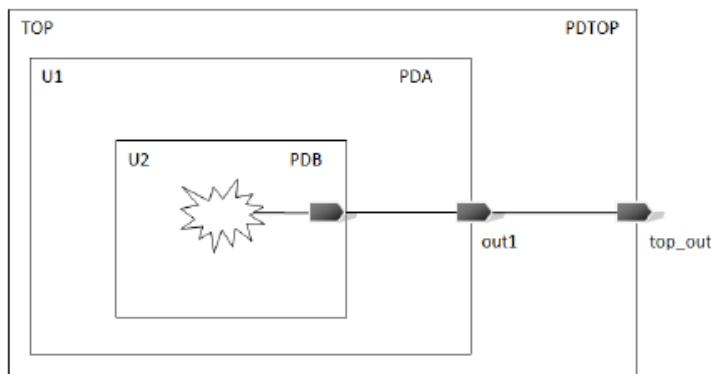
```

Tag : UPF_SPASUPPLY_VOLTAGE
Description : Voltage difference between supply of actual driver/receiver
and constrained supply for SPA constraint on [ConstraintNode]
Violation : LP:1
ConstraintNode : P
ReceiverNode : blk2/I1/A
ConstraintInfo
  PowerNet
    NetName : Vblk1b
    NetType : UPF
    PowerMethod : FROM_UPF_RECEIVER_SUPPLY
ReceiverInfo
  PowerNet NetName : Vtop
  NetType : Design/UPF
  PowerMethod : FROM_UPF_POWER_DOMAIN
TerminalBdry : true
  Internal : True
  UPFCommand
    Command : set_port_attributes
    FileName : constraint.upf
    LineNumber : 2

```

- ❖ Strategy association checks are handled differently on terminal boundary nodes. Consider the following example:

**Figure 5-81 Strategy Isolation Checks**



```

UPF for U1:
create_supply_set SSA
create_supply_set SSB
create_power_domain PDA -include_scope -supply {primary SSA}
create_power_domain PDB -elements U2 -supply {primary SSB}
set_design_attribute -elements {.} -attribute terminal_boundary true

```

```

UPF for top:
create_supply_set SSTOP
create_power_domain TOP -supply {primary SSTOP}

```

```

set_isolation iso1 -domain PDTOP -source SSB
set_isolation_control iso1 -domain PDTOP -location self
set_isolation iso2 -domain PDTOP -source SSA
set_isolation_control iso2 -domain PDTOP -location self

```

The design path is from *U2 to TOP*, so the SSB *iso1* strategy is implemented as the source.

As there is an terminal boundary attribute defined on *U1*. This means *U1* is implemented standalone and two paths *U1 to out1* and *out1 to top\_out* are considered instead of the single path. On the path from *out1 to top\_out*, the effective source supply is SSA. Therefore, the *iso2* strategy is implemented.

### **Limitation**

Handling of *-repeater\_supply* specified in a terminal boundary node is complex, and is not commonly used in customer designs. VC LP does not support handling *-repeater\_supply* in the current terminal boundary flow.

### **5.3.14 Support for `create_power_domain -exclude_elements`**

The support for *-exclude\_elements* is available for the `create_power_domain` command. Through this, the list of design elements to exclude from the effective element list can be specified under `create_power_domain` command and when the `create_power_domain -exclude_elements` option is specified, the instances listed in the `-exclude_elements` is not considered as part of the power domain.

#### **Syntax**

```
create_power_domain -exclude_elements <exclude_list>
```

Where

- ❖ `-exclude_elements <exclude_list>`: The list of design elements to exclude from the `effective_element_list`.

#### **Example**

```
create_power_domain PD1 -elements {A B C} -exclude_elements {B C}
```

Here, the effective element list of PD1 is A.

### **5.3.15 Support for `soft_macro` Attributes**

VC LP has been supporting `terminal_boundary` solution to bridge the gap between verification tool and implementation tool. When terminal boundary is enabled, VC LP breaks the crossover at terminal boundaries and do the check.

But terminal boundary solutions are lacking in

- ❖ Self-contained upf definition checking
- ❖ Supply connections being honored across the terminal boundaries

As a solution, VCLP supports `soft_macro` attributes

#### **Use Model**

```

set_design_attributes -models {..} -is_soft_macro TRUE
set_design_attributes -models {macro1 macro2} -attributes {UPF_is_soft_macro TRUE}

```

As a self-contained checker, additional UPF parser messages are introduced with this support:

- ❖ UPF\_MACRO\_UPDATE\_NOT\_ALLOWED {Macro object update not allowed}
- ❖ UPF\_MACRO\_WRONG\_LOCATION {Strategy defined at wrong location}

- ❖ UPF\_MACRO MODIFY\_FROM\_OUTSIDE {Macro object modify from outside}
- ❖ UPF\_SOFT\_MACRO\_INSTANCE\_NOT\_PARTITIONED {Top scope of soft macro not partitioned}
- ❖ UPF\_WITHIN\_MACRO\_ACCESS {Macro object accessed from outside}

The `-traverse_macros` option allows the `find_objects` command to descend and traverse all hierarchies without stopping at any boundary marked as `UPF_is_soft_macro` or `terminal_boundary` using the `set_design_attribute` command. Even with this option, the traversal will stop at standard cells and hard macro boundaries. If the `-traverse_macros` option is specified, then `-transitive TRUE` is implicitly specified. This `traverse_macros` command only supports the `-object_type model/inst` option.

- ❖ If the `-traverse_macros` option is specified with `-object_type` other than `model` or `inst`, the `UPF_FIND_OBJECTS_TRAVERSE_MACROS_INVALID_OBJECT_TYPE` parser warning is reported, and the `traverse_macros` option is ignored.

#### Example

```
find_objects . -pattern in1 -object_type port -traverse_macros
```

*[Warning] UPF\_FIND\_OBJECTS\_TRAVERSE\_MACROS\_INVALID\_OBJECT\_TYPE: Invalid object type for -traverse\_macros option*

*Option -traverse\_macros can be specified only with '-object\_type model/inst' in command 'find\_objects'. The option will be ignored.*

*'-traverse\_macros' can be specified only with '-object\_type model/inst'.*

- ❖ When `-traverse_macros` option is specified without `-object_type` `UPF_FIND_OBJECTS_TRAVERSE_MACROS_DEFAULT` parser warning message is reported. VC LP continues to return the default types outside macros, but only objects of type 'instance' inside macros will be returned.

#### Example

```
find_objects . -pattern * -traverse_macros
```

*[Warning] UPF\_FIND\_OBJECTS\_TRAVERSE\_MACROS\_DEFAULT: Only return some objects when -object\_type isn't specified*

*By default, instances, named processes, logic ports, and nets are returned when -object\_type is not specified. When -traverse\_macros is specified, tool continues to return the default types outside macros, but returns only objects of type 'instance' inside macros.*

The following violations are introduced to guide the application of isolation strategies on top level ports, assuming that the sub IPs in the design have also run these checks at their block level validation.

- ❖ ISO\_STRATEGY\_INPUT

It is not recommended to define an isolation strategy on the topmost input ports. This violation will be reported if an isolation strategy is detected to be defined on the primary input ports.

If this behavior is not intended, ensure not apply an isolation strategy on the primary inputs.

- ❖ ISO\_STRATEGY\_SINK

If the topmost instance is not defined as a soft macro with the `set_design_attribute -attribute is_soft_macro true` command, it is not recommended to define a path specific isolation strategy on its primary output ports with the `-sink` option. If this behavior is not intended, ensure remove the sink info.

- ❖ ISO\_STRATEGY\_DIFFSUPPLY

If a top most instance is not defined as a soft macro with the `set_design_attribute -attribute is_soft_macro true`, it is not recommended to define a path specific isolation strategy on its primary output ports with the `-diff_supply_only true` option. If this behavior is not intended, make sure remove the `-diff_supply_only` option.

- ❖ ISO\_STRATEGY\_PATH

For a topmost instance which is not defined as a soft macro with the `set_design_attribute -attribute is_soft_macro true`, if its primary output port has any isolation strategy applied, and source supply is specified for the isolation strategy, the source supply is expected to be in the switch path fanout of the isolation supply. Please add power switch strategies or cells to make sure the isolation supply is the power switch ancestors of the source supply.

### 5.3.16 Support for Leaf cells as Elements in the `create_power_domain` Command

VC LP considers leaf cells without function as a black box and allows it with the `create_power_domain -elements`.

#### Example:

#### Design:

```
module top(in);
  input in ;
  BUF u_BUF(.DiffClkIn0_H(in));
endmodule
```

#### UPF:

```
create_power_domain PD_P1 -elements { u_BUF}
```

In this example, BUF is leaf cell (without function).

VC LP allows this leaf cell to be used with the `create_power_domain PD_P1 -elements` command.

#### Liberty Snippet of a Leaf Cell:

```
cell(BUF) {
  always_on : true;
  ...
  pin(A) {
    related_power_pin : VDDG;
    related_ground_pin : VSSG;
    direction : input;
    capacitance : 1;
  }
  pin(Z) {
    related_power_pin : VDDG;
    related_ground_pin : VSSG;
    direction : output;
  }
}
```

As shown in the liberty for BUF cell, there is no function defined for output Z.

### 5.3.17 Support for Leaf Cell Pins in the -control\_port Switch of Power Switch Strategy

VC LP has introduced the `allow_pins_in_psw_control_port` application variable for supporting leaf cell pins in the `-control_port` switch of power switch strategy.

By default, the `allow_pins_in_psw_control_port` application variable is set to false, and VC LP reports error message when leaf cell pins are used in the `-control_port` switch of a power switch strategy.

#### Example

Consider the following UPF snippet:

```
create_power_switch SW_PD_P1 \
-domain PD_P1 \
-input_supply_port {vin VDD} \
-output_supply_port {vout VDDINT_P1} \
-control_port {sleep {u_pd_inv/ZN}}
```

The following error message is reported for this example when the `allow_pins_in_psw_control_port` application variable is set to false:

*test.upf:40: [Error] UPF\_OBJECT\_NOT\_FOUND\_ERROR: Object not found*

*Object 'u\_pd\_inv/ZN' of type 'Net' specified with command option 'create\_power\_switch -control\_port' could not be resolved. Resolved Path 'Instance /u\_pd\_inv(INVD1HVT:test.v:5)'. Unresolved Path 'ZN'.*

*Please specify a valid object.*

When the `allow_pins_in_psw_control_port` application variable is set to true, VC LP does not report an error when a leaf cell's pin is mentioned in the `-control_port` switch of power switch strategy.

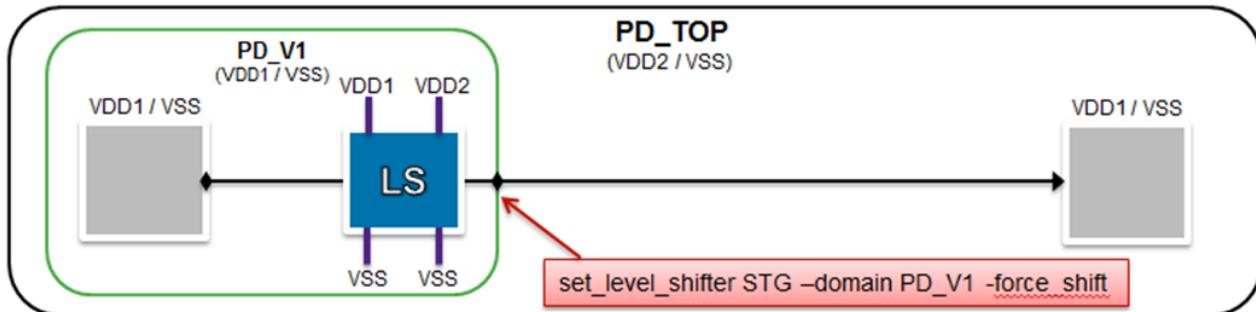
### 5.3.18 Support for -force\_shift Option in the set\_level\_shifter Command

The `set_level_shifter` command in UPF is a soft constraint and it adds level shifters only if required and it may not necessarily conform to the user strategy. However, if required, you can add level shifters using the `-force_shift` option in the `set_level_shifter` command.

The following are example situations where level shifters can be inserted using the `-force_shift` option:

- ❖ Cases where there is no voltage violation.
- ❖ Cases where there is no real driver/load.
- ❖ Cases where the driver is logic-zero.

**Figure 5-82 Example for Using -force\_shift option in set\_level\_shifter Command**



### 5.3.18.1 Errors and Warnings Messages

- ❖ If you use the `-force_shift` option with the `-no_shift` option, the following error message is displayed, and the parser exits.  
`[Error] Tcl_OPT_ATMOST_ONE_OF: Incorrect command options  
It is not legal to specify more than one of '(-no_shift, -force_shift)' in command 'set_level_shifter'.`
- ❖ If you use the `-force_shift` option with the `-threshold` option, the threshold value is dropped and the following warning message is displayed.  
`[[Warning] UPF_CMD_OPTION_COMBS_NOT_ALLOWED: Command option combinations not allowed  
Option '-threshold' is used in combination with '-force_shift' is not allowed. '-threshold' will be ignored.`

### 5.3.18.2 VC LP Checks for Level Shifters

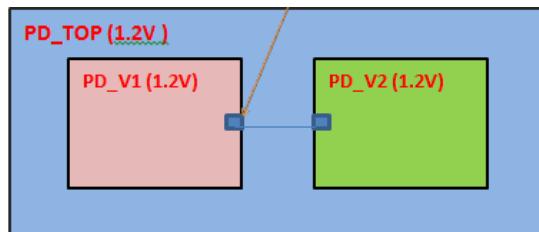
The following tags are introduced to check if level shifter strategies are written with the `-force_shift` option, when the level shifter is not required.

- ❖ LS\_STRATEGY\_FORCE

As shown in the example in [Figure 5-83](#), level shifter is not required on cross-over from source to sink, based on the supply states defined in the pst. However, in the UPF, level shifter strategy is written with the `-force_shift` option. The LS\_STRATEGY\_FORCE violation is reported for such cases.

```
set_level_shifter ls_V1_out -domain PD_V1 -force_shift -location automatic -applies_to outputs -rule both
```

**Figure 5-83 LS\_STRATEGY\_FORCE**



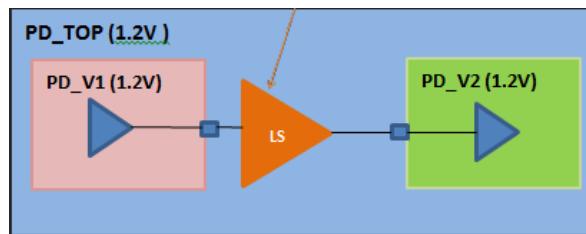
If you have intentionally used the `-force_shift` option in strategy, then you can ignore this violation, else correct the strategy.

The LS\_STRATEGY\_FORCE violation is enabled by default. The severity of this violation is *Warning*.

- ❖ LS\_INST\_FORCE

As shown in the example in [Figure 5-84](#), the level shifter instance is not required on the crossover from source to sink, based on the supply states defined in the pst. However, in the UPF, level shifter strategy is written with the `-force_shift` option. Also, a corresponding level shifter cell is present. The LS\_INST\_FORCE violation is reported for such cases.

```
set_level_shifter ls_V1_out -domain PD_V1 -force_shift -location parent -applies_to outputs -rule both
```

**Figure 5-84 LS\_INST\_FORCE Violation**

If you have intentionally used the `-force_shift` option in strategy, then you can ignore this violation, else correct the strategy.

The LS\_INST\_FORCE violation is enabled by default. The severity of this violation is *Warning*.

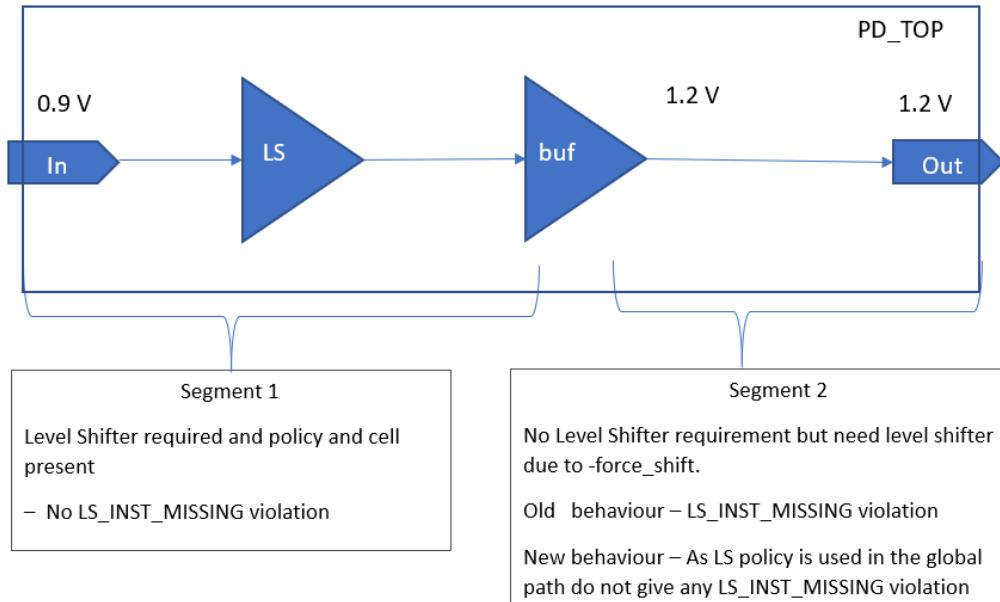
### 5.3.18.3 Impact on LS\_INST\_MISSING Violation

When `enable_global_ls_analysis` application variable is set to false, full path is split into multiple segment and LS\_INST\_MISSING check is performed on individual segments.

The LS\_INST\_MISSING violation is reported when a level shifter policy had `-force_shift` option and instance was not present in that segment without considering that the association node might be present in another segment.

When the `-force_shift` is provided, the LS\_INST\_MISSING violation is suppressed for the segment if the level Shifter strategy and cell is associated in some other segment of same path.

```
set_level_shifter PD_TOP_LS
-domain PD_TOP -location automatic -rule low_to_high
-elements Out
-force_shift
```



### 5.3.19 Support for voltage\_map attribute for Level Shifter Checks

The `voltage_map` attribute written at library pin is considered for level shifter checks in a crossing. This `voltage_map` attribute is considered whenever source and sink are PST equivalent. If source and sink are not PST equivalent, then the level shifter checks are based on the PST. To enable this support, set the `enable_voltage_map_ls_analysis` application variable is set to `true`.

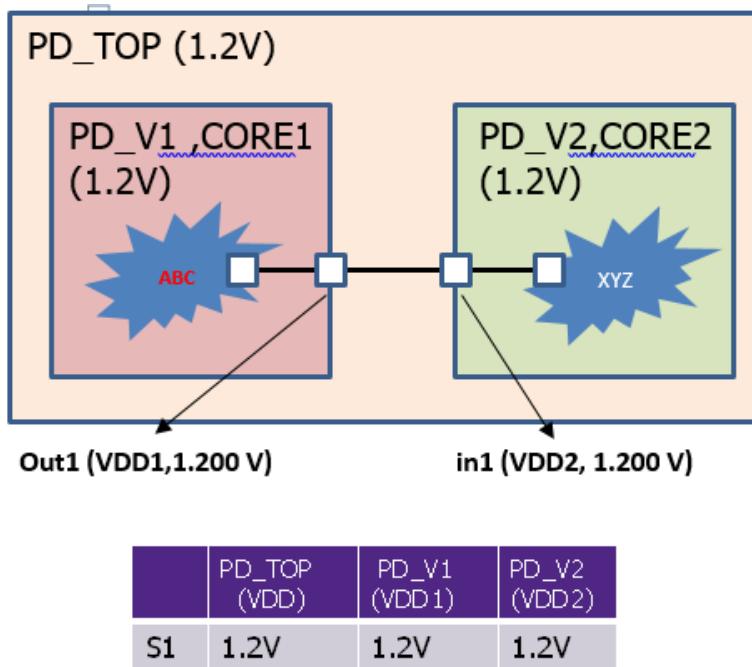
By default, the `enable_voltage_map_ls_analysis` application variable is set to `false`.



#### Note

This feature is a custom feature and is controlled by additional variables. For more details on this support, contact [mvsupport@synopsys.com](mailto:mvsupport@synopsys.com). This is a LCA feature.

**Figure 5-85 Example for voltage\_map attribute**



```
library(lib1)
{
  voltage_map(vdds, 1.00);
  cell(ABC)
  {
    pg_pin(vdds)
    {
      pg_type : internal_power;
      direction : internal;
      voltage_name : vdds;
    }
    pin(out1)
    {
      related_power_pin : vdds;
    }
  }
}
```

As shown in Figure 5-85, if the library pin has the `related_power_pin`, the `pg_type` of power pin must have `internal_power` and its direction must be `internal`. The `voltage_map` written for the above mentioned power pin must be considered for a crossing.

Figure 5-85 shows that there is a crossing between pin `out1` of instance `CORE1` and pin `in1` of instance `CORE2`. Both pins are at same voltage domain (as per UPF) and level shifter requirement is none. As per `voltage_map` written at `out1`, the level shifter requirement is `low_to_high`. The `LS_STRATEGY_MISSING` is reported for this scenario when the `enable_voltage_map_ls_analysis` application variables is set to `true`.



#### Note

If `connect_supply_net` (CSN) is already written on the pin, it gets the priority over `voltage_map`. CSN is already annotated on pins.

When the `voltage_map` attribute is considered for the level shifter checks, the `VoltageValueMap` debug field is added for the level shifter violations.

Tag : LS\_STRATEGY\_MISSING

```

Description      : Level shifter required on crossing from [Source] to [Sink], but
no strategy defined

Violation       : LP:4

Source         

  PinName       : CORE1/macro_vol_map_i1/Z
  Sink          : CORE2/and_i1/A
  PstRuleValue  : LH_TYPE
  SegmentSourceDomain : PD1
  SegmentSinkDomain  : PD2

LogicSource    


  PinName       : CORE1/macro_vol_map_i1/Z
  LogicSink    : CORE2/and_i1/A
  DomainSource : CORE1/out2
  DomainSink   : CORE2/in1

SourceInfo     


  PowerNet     


    NetName      : VDD1
    NetType      : UPF
    PowerMethod  : FROM_UPF_POWER_DOMAIN

  GroundNet     


    NetName      : VSS
    NetType      : UPF
    GroundMethod : FROM_UPF_POWER_DOMAIN
    VoltageValueMap : 1

SinkInfo       


  PowerNet     


    NetName      : VDD2
    NetType      : UPF
    PowerMethod  : FROM_UPF_POWER_DOMAIN

  GroundNet     


    NetName      : VSS
    NetType      : UPF
    GroundMethod : FROM_UPF_POWER_DOMAIN

```

The following violations are affected by the `voltage_map` library attribute:

- ❖ UPF\_CROSSOVER\_NOSTATE
- ❖ LS\_INST\_REDUND
- ❖ LS\_INST\_OPTIMIZE
- ❖ LS\_INPUT\_VOLTAGE
- ❖ LS\_OUTPUT\_VOLTAGE
- ❖ LS\_INST\_NOSTRAT
- ❖ LS\_INST\_OK
- ❖ LS\_NOTHRESH\_OK
- ❖ LS\_INST\_MISSING
- ❖ LS\_INST\_INCORRECT
- ❖ LS\_STRATEGY\_INCORRECT
- ❖ LS\_STRATEGY\_MISSING
- ❖ LS\_STRATEGY\_REDUND

- ❖ LS\_NOSTRAT\_OK



### Note

When the `lp_assume_floating_ls_source` application variable is set to true, VC LP enables checking of LS\_INST\_INCORRECT in case of floating LS/ELS input. The LS/ELS input is assumed as source for checking LS\_INST\_INCORRECT.

### 5.3.20 Support for -name\_suffix/name\_prefix in the set\_isolation/set\_level\_shifter Commands

VC LP supports the `-name_suffix` and `-name_prefix` in the `set_isolation`/`set_level_shifter` commands. These options do not influence strategy association.

- ❖ `set_isolation iso1 -domain PD1 -applies_to outputs -isolation_supply_set TOP.primary -clamp_value 0 -name_prefix lp_ -name_suffix _iso`
- ❖ `set_level_shifter ls1 -domain TOP -elements {out4 out5 out6} -rule low_to_high -name_prefix lp_ -name_suffix _ls1h`

VC LP reports the following violations if the name specified in the `-name_suffix` and `-name_prefix` options does not match with the associated instance. The severity of these violations is *Warning* and is enabled by default. VC LP reports this violation when instance and strategy are associated to each other.

- ❖ ISO\_INST\_PREFIX: This violation is reported when the isolation instance name does not match with the `name_prefix` option of the isolation strategy.

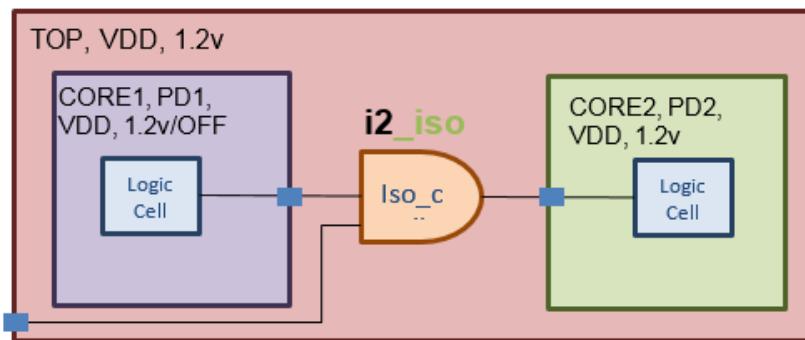
#### Example

Consider the following UPF snippet:

```
set_isolation iso1 -domain PD1 -applies_to outputs -isolation_supply_set TOP.primary -clamp_value 0 -name_prefix lp_ -name_suffix _iso -isolation_signal iso_en -isolation_sense low -clamp_value 0 -location parent
```

Consider the design as shown in [Figure 5-86](#).

**Figure 5-86 ISO\_INST\_PREFIX Example**



The following is the violation reported for this scenario:

```
Tag      : ISO_INST_PREFIX
Description : Isolation instance [Instance] does not match name_prefix option of
              strategy [Strategy]
Violation   : LP:26
Instance    : i2_iso
Strategy    : PD1/iso1
Option      : lp_
```

- ❖ ISO\_INST\_SUFFIX: This violation is reported when the isolation instance name does not match with the name\_suffix option of the isolation strategy.

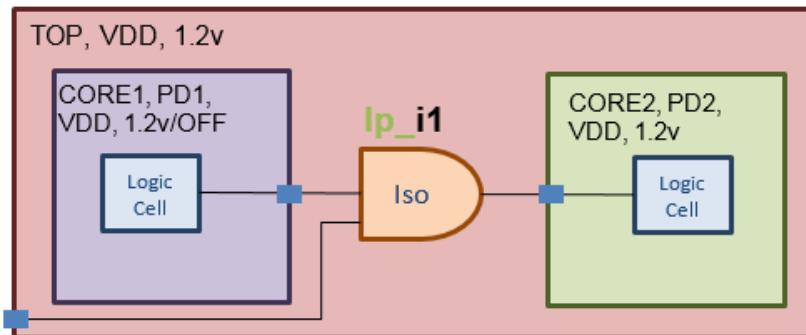
**Example:**

Consider the following UPF snippet:

```
set_isolation iso1 -domain PD1 -applies_to outputs -isolation_supply_set TOP.primary -
clamp_value 0 -name_prefix lp_ -name_suffix _iso -isolation_signal iso_en -
isolation_sense low -clamp_value 0 -location parent
```

Consider the design as shown in [Figure 5-87](#).

**Figure 5-87 ISO\_INST\_SUFFIX Example**



The following is the violation reported for this scenario:

```
Tag      : ISO_INST_SUFFIX
Description : Isolation instance [Instance] does not match name_suffix option of
strategy [Strategy]
Violation  : LP:25
Instance   : lp_i1
Strategy   : PD1/iso1
Option     : _iso
```

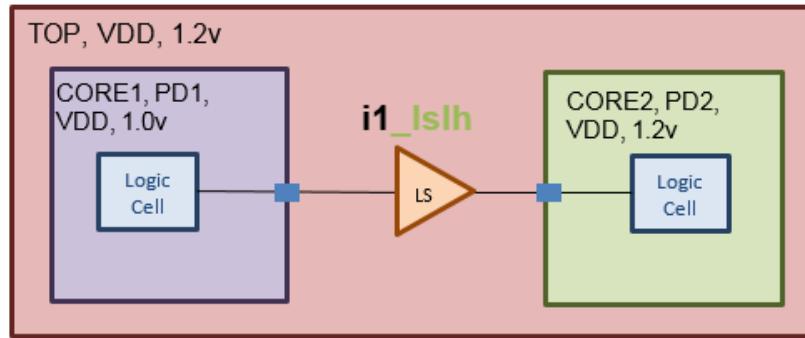
- ❖ LS\_INST\_PREFIX: This violation is reported when the level shifter instance name does not match with the name\_prefix option of the level shifter strategy.

**Example:**

Consider the following UPF snippet:

```
set_level_shifter ls1 -domain PD1 applies_to outputs -rule low_to_high -name_prefix lp_-
-name_suffix _lslh -location parent
```

Consider the design as shown in [Figure 5-88](#).

**Figure 5-88 LS\_INST\_PREFIX Example**

The following is the violation reported for this scenario:

```

Tag      : LS_INST_PREFIX
Description : Level shifter instance [Instance] does not match name_prefix option of
strategy [Strategy]
Violation   : LP:28
Instance    : i1_lslh
Strategy    : PD1/ls1
Option      : lp_

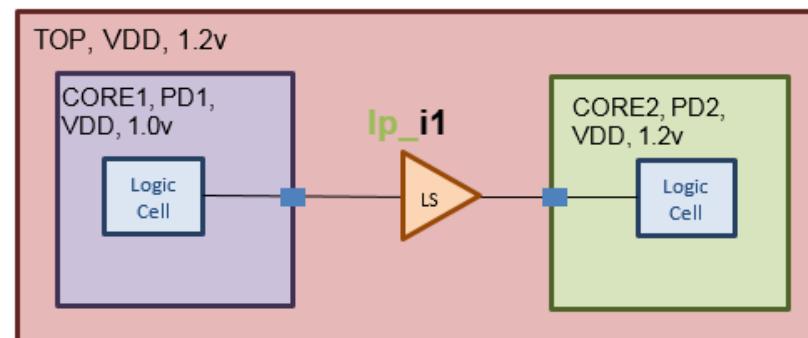
```

- ❖ **LS\_INST\_SUFFIX:** This violation is reported when the level shifter instance name does not match with the name\_suffix option of the level shifter strategy.

Consider the following UPF snippet:

```
set_level_shifter ls1 -domain PD1 applies_to outputs -rule low_to_high -name_prefix lp_
-name_suffix _lslh -location parent
```

Consider the design as shown in [Figure 5-89](#).

**Figure 5-89 LS\_INST\_SUFFIX Example**

The following is the violation reported for this scenario:

```

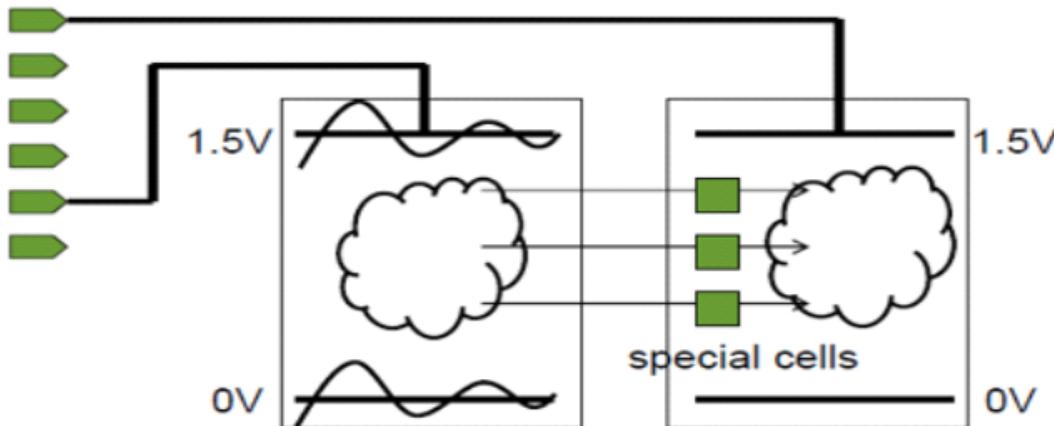
Tag      : LS_INST_SUFFIX
Description : Level shifter instance [Instance] does not match name_suffix option of
strategy [Strategy]
Violation   : LP:27
Instance    : lp_i1
Strategy    : PD1/ls1
Option      : _lslh

```

### 5.3.21 Detecting PST Equivalent Source and Sink Crossover

In modern SoC's there are many voltage sources which supply voltage for different parts of the chip, such as digital supply and analog supply. Although they may be defined to be at the same voltage level in the UPF power state table (PST), in reality there may be variations in voltage level due to different resistance in power grid.

**Figure 5-90** PST Equivalent Source and Sink Crossover



As shown in [Figure 5-90](#), as both supplies are at 1.5V, by default, VC LP does not report any warning or error messages as this is the desired design behavior. However, you may have to insert special cells such as diode buffers (for ESD protection), or same-voltage level shifter cells. VC LP reports violation for such scenarios if the following application variable is set to true.

```
%vc_static_shell> enable_reporting_pst_equivalent_crossovers
```

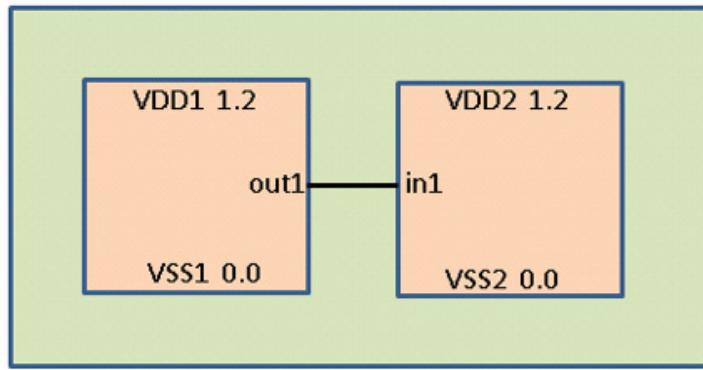
By default, the `enable_reporting_pst_equivalent_crossovers` application variable is set to false.

When the `enable_reporting_pst_equivalent_crossovers` application variable is set to true, VC LP checks if there is crossing between source and sink nodes which are PST equivalent, but are supplied by physically different rails. VC LP reports the following violations.

The violations have the severity warning and are reported at `check_lp -stage design`.

- ❖ LS\_SAMEVOLTXING\_PWRGND

This violation reported when the source and sink are PST equivalent but, the power and ground rail of the source and sink are physically different.

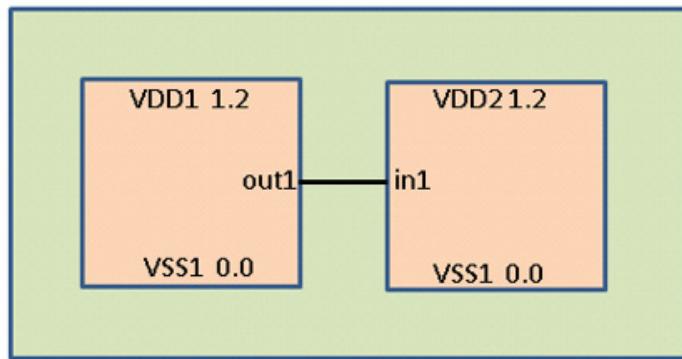
**Figure 5-91 Example for LS\_SAMEVOLTXING\_PWRGND Violation**

```

Tag : LS_SAMEVOLTXING_PWRGND
Description : Power and ground supplies for [Source] and [Sink] are at same
voltage but physical different
Violation : LP:1
Source
  PinName : CORE2/xor_i1/Z
  Sink : out1
  SegmentSourceDomain : PD2
  SegmentSinkDomain : TOP
  LogicSource
    PinName : CORE2/xor_i1/Z
    LogicSink : out1
    DomainSource : CORE2/out1
    DomainSink : out1
  SourceInfo
    PowerNet
      NetName : VDD2
      NetType : UPF
    PowerMethod : FROM_UPF_POWER_DOMAIN
    GroundNet
      NetName : VSS2
      NetType : UPF
    GroundMethod : FROM_UPF_POWER_DOMAIN
  SinkInfo
    PowerNet
      NetName : VDD
      NetType : UPF
    PowerMethod : FROM_UPF_POWER_DOMAIN
    GroundNet
      NetName : VSS
      NetType : UPF
    GroundMethod : FROM_UPF_POWER_DOMAIN
  
```

❖ **LS\_SAMEVOLTXING\_PWR**

This violation reported when the source and sink are pst equivalent but, only the power rails are physically different.

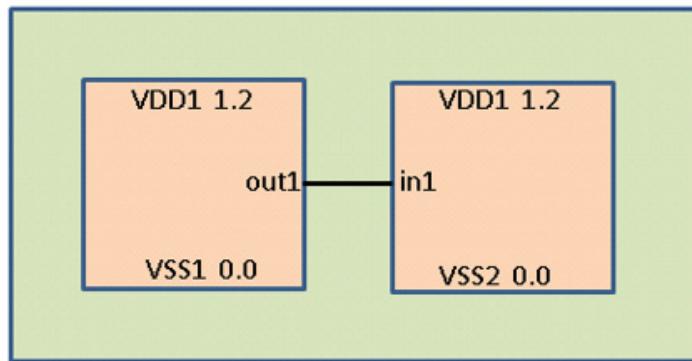
**Figure 5-92 Example for LS\_SAMEVOLTXING\_PWR Violation**

```

Tag : LS_SAMEVOLTXING_PWR
Description : Power supplies for [Source] and [Sink] are at same voltage but
physical different
Violation : LP:3
Source
  PinName : CORE2/xor_i1/Z
  Sink
  SegmentSourceDomain : PD2
  SegmentSinkDomain : TOP
  LogicSource
    PinName : CORE2/xor_i1/Z
  LogicSink
    : out1
  DomainSource : CORE2/out1
  DomainSink : out1
  SourceInfo
    PowerNet
      NetName : VDD2
      NetType : UPF
    PowerMethod : FROM_UPF_POWER_DOMAIN
    GroundNet
      NetName : VSS
      NetType : UPF
    GroundMethod : FROM_UPF_POWER_DOMAIN
  SinkInfo
    PowerNet
      NetName : VDD
      NetType : UPF
    PowerMethod : FROM_UPF_POWER_DOMAIN
    GroundNet
      NetName : VSS
      NetType : UPF
    GroundMethod : FROM_UPF_POWER_DOMAIN
  
```

#### ❖ LS\_SAMEVOLTXING\_GND

This violation reported when the source and sink are pst equivalent but, only the ground rails are physically different.

**Figure 5-93 Example for LS\_SAMEVOLTXING\_GND Violation**

Tag Description

```

Tag          : LS_SAMEVOLTXING_GND
Description   : Ground supplies for [Source] and [Sink] are at same voltage but
physical different
Violation     : LP:3
Source
  PinName      : CORE2/xor_i1/Z
Sink
  PinName      : out1
SegmentSourceDomain : PD2
SegmentSinkDomain   : TOP
LogicSource
  PinName      : CORE2/xor_i1/Z
LogicSink
  PinName      : out1
DomainSource
  Domain      : CORE2/out1
DomainSink
  Domain      : out1
SourceInfo
  PowerNet
    NetName    : VDD
    NetType     : UPF
    PowerMethod: FROM_UPF_POWER_DOMAIN
  GroundNet
    NetName    : VSS2
    NetType     : UPF
    GroundMethod: FROM_UPF_POWER_DOMAIN
SinkInfo
  PowerNet
    NetName    : VDD
    NetType     : UPF
    PowerMethod: FROM_UPF_POWER_DOMAIN
  GroundNet
    NetName    : VSS
    NetType     : UPF
    GroundMethod: FROM_UPF_POWER_DOMAIN

```

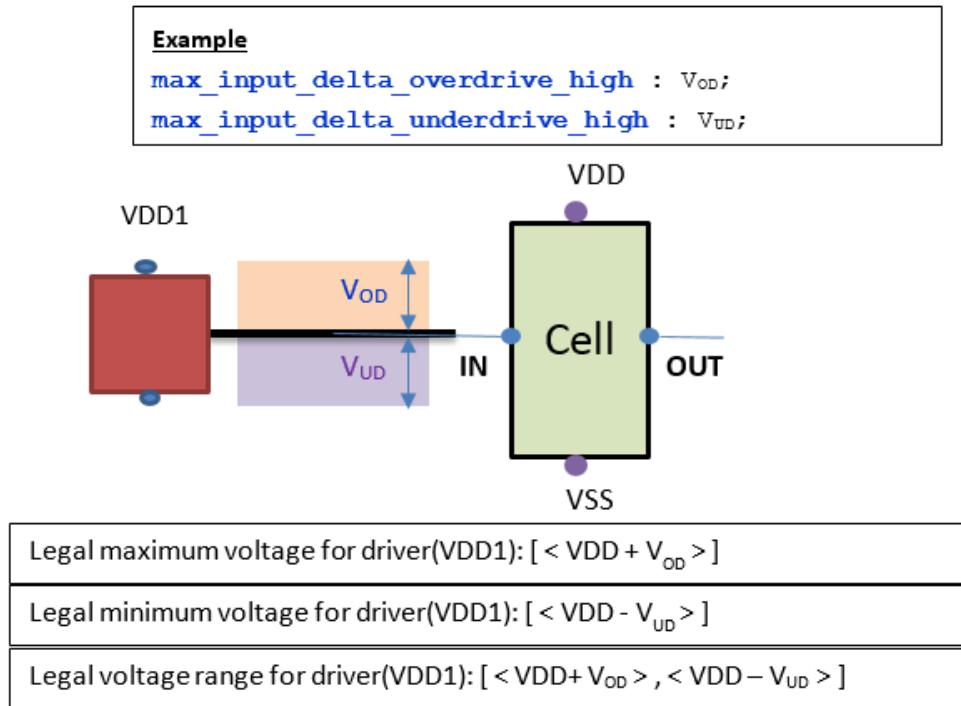
### 5.3.21.1 Support for Overdrive and Underdrive Attributes

VC LP supports the following new attributes to model overdrive and underdrive of a signal pin with respect to the root supply of the load.

- ❖ max\_input\_delta\_overdrive\_high
- ❖ max\_input\_delta\_underdrive\_high

VC LP supports the `input_signal_level` attribute which is a corner dependent attribute under the `lp_support_isl_check` application variable. The new attributes are introduced to remove this dependency.

The following figure illustrates the overdrive and underdrive modeling on a library cell's signal pin.



### Counter Example

$$VDD = 1.0 \text{ v}; \quad Vod = 0.2 \text{ v}; \quad Vud = 0.3 \text{ v}$$

$$\text{Legal voltage range of the driver} = (1.0 - 0.3) : (1.0 + 0.2) = 0.7 : 1.2$$

The following are the characteristics of the `max_input_delta_overdrive_high` and `max_input_delta_underdrive_high` attributes:

- ❖ These attributes take only floating values.
- ❖ The default value of this attribute is 0.0.
- ❖ Negative values are not supported for this attribute.
- ❖ The new attributes are checked only on input or inout pins.
- ❖ These new attributes do not impact the crossover generation or policy to cells association.
- ❖ You can define the new attributes only on power switch cells, level shifters, macros, and pad cells.
- ❖ If these attributes are present on any other type of cells, then VC LP does not flag any warning or error for these scenarios.
- ❖ VC LP considers `root_supply_net` of the load, based on the precedence (SRSN > PG > CSN > Strategy, > Domain).
- ❖ VC LP reads the new attributes and modify the checking of all tags where voltage violations are reported.
- ❖ Triplet support is available on both, none, and correlated supplies.

- ❖ No new checks are added for the new attributes.

### Use model

VC LP supports `max_input_delta_overdrive_high` and `max_input_delta_underdrive_high` liberty attributes written in the library. You need not define these attributes using the `set_attribute` Tcl commands. However, you can override the existing values in the library for these attributes using the `set_attribute` Tcl command.

You can set the `max_input_delta_overdrive_high` and `max_input_delta_underdrive_high` attributes on library pins as follows.

- ❖ Directly on lib cell pin

```
set_attribute [get_lib_pins <lib_pin_name>] max_input_delta_overdrive_high  
<value>  
set_attribute [get_lib_pins <lib_pin_name>] max_input_delta_underdrive_high  
<value>
```

#### Example

```
set_attribute [get_lib_pins lib1/cell1/d_in ] max_input_delta_overdrive_high 0.2  
set_attribute [get_lib_pins lib1/cell1/d_in] max_input_delta_underdrive_high 0.2
```

- ❖ Get lib pins through instance pins

```
set_attribute [get_lib_pins -of_objects <instant_pin_name>]  
max_input_delta_overdrive_high <value>  
set_attribute [get_lib_pins -of_objects <instant_pin_name>]  
max_input_delta_underdrive_high <value>
```

#### Example

```
set_attribute [get_lib_pins -of_objects macro1/A] max_input_delta_overdrive_high 0.2  
set_attribute [get_lib_pins -of_objects macro1/A] max_input_delta_underdrive_high 0.1
```

### Limitations

- ❖ The `remove_attribute` Tcl command is not supported for these attributes.

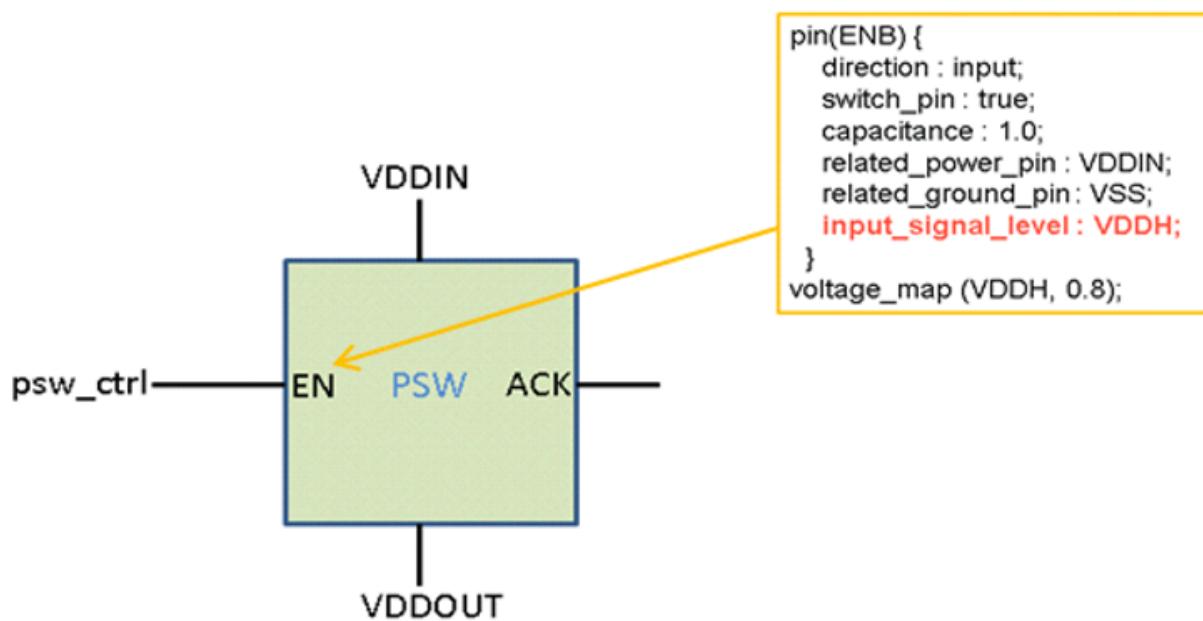
## 5.3.22 Support for `input_signal_level` for Level Shifting Checks

VC LP supports `input_signal_level` for power switch cell. The `input_signal_level` on enable pin of power switch helps to reduce leakage. Overdrive voltage on enable pin of PSW cell reduces leakage and switch goes into deep shut down mode.

When the `input_signal_level` is specified on the power switch cell pin, VC LP ignores `related_power_pin(RPP)` pin and considers the `input_signal_level` for level shifter checks.

The `input_signal_level` attribute is supported under the `lp_support_isl_check` application variable. By default, the application variable is set to false.

If you enable the `lp_support_isl_check` application variable, then VC LP honors the `input_signal_level` attribute, and in this mode the `input_signal_level` has higher priority over new attributes.

**Figure 5-94 Example for input\_signal\_level for Level Shifting Checks**

- In the example in [Figure 5-94](#), if the driver Supply (PST wise) which is driving the ENB input pin is lower than the `input_signal_level`, then the `DESIGN_VOLTAGE_LEVEL` violation is reported. Therefore, the `LS_INST_MISSING` check is ignored. The `DESIGN_VOLTAGE_LEVEL` violation disabled by default.

Tag Description:

```

Tag          : DESIGN_VOLTAGE_LEVEL
Description   : Enable pin with higher input_signal_level driven by immediate
source which is at low voltage
Violation     : LP:5
LogicSource
  PinName      : psw_ctrl
  LogicSink    : psw_i2/ENB
SourceInfo
  PowerNet
    NetName      : vddout
    NetType       : Design/UPF
  PowerMethod   : FROM_UPF_POWER_DOMAIN
  GroundNet
    NetName      : VSS
    NetType       : Design/UPF
  GroundMethod  : FROM_UPF_POWER_DOMAIN
  VoltageValueMap : 1.3
SinkInfo
  PowerNet
    NetName      : vddin
    NetType       : Design/UPF
  PowerMethod   : FROM_PG_NETLIST
  GroundNet
    NetName      : VSS
    NetType       : Design/UPF
  
```

```

GroundMethod      : FROM_PG_NETLIST
InputSignalLevel : 1.4
States
  State        : pst1/s1

```

- ❖ In the example in [Figure 5-94](#), if the driver Supply (PST wise) which is driving ENB input pin is higher than or equal to `input_signal_level`, then it is a valid design scenario. Therefore, previously the `LS_INST_MISSING` violation was ignored, and there is no violation reported on the same crossover.

### 5.3.23 Support for `iso_source` and `iso_sink` Attribute in `set_port_attributes`

The `set_port_attribute` UPF command helps designers to attach various power specific attributes on boundary ports of a power domain. The implementation tools can specify supply information for the boundary ports using the `set_port_attribute` command, and can perform synthesis in the hierarchical flow.

VC LP supports the `iso_source` and `iso_sink` attribute on the signal ports (not on UPF supply ports) in the `set_port_attribute` command. These attributes are different from the `driver_supply` and `receiver_supply` attributes, in the sense that they do not specify immediate drivers or receivers, but specify supplies of those beyond the isolation/level-shifter/AON cell.

#### Syntax

```

set_port_attribute -ports { out1 out2 out3 } -attribute iso_sink {sset2 }
set_port_attribute -ports { in1 in2 } -attribute -iso_source {sset1}

```

You cannot define both `iso_source` and `iso_sink` attributes for a port. You can define the `iso_source` attribute only for the input ports, and the `iso_sink` attribute only for the output ports.

If you define both `iso_sink` and `iso_source` attributes for a port, VC LP reports the following parser error message:

*[Error] UPF\_SPA\_BOTHSRCINK\_ATTRIBUTE: Port Attributes iso\_source and iso\_sink specified togetherPort Attributes iso\_source and iso\_sink specified together for design object 'top/in1'*



#### Note

VC LP issues `TCL_INV_ATTR` warning message for unknown attributes defined in `set_port_attributes` and `set_design_attributes` commands. Same message is issued for each and every ports, models and elements.

#### Example

```

set_port_attributes -ports {mod1/cell0/D cell1/CP } -attribute unknown true
<File_name>:<Line_Number>: [Warning] TCL_INV_ATTR: Invalid Attribute specified
Invalid Attribute 'unknown' specified in command 'set_port_attributes'. Attribute will be ignored.
Please refer to the 'User Guide' for supported attributes.

```

### 5.3.23.1 Support for `DERIVED_DIFF_ONLY` and `DERIVED_DIVERSE_ONLY` in `set_port_attributes`

In a top-down implementation flow, the `iso_source`/`iso_sink` attributes are added on block level ports to characterize source/sink isolation strategies while generating block level UPF.

VC LP also supports `DERIVED_DIFF_ONLY` and `DERIVED_DIVERSE_ONLY` attributes. The following rules are applicable for port attribute setting:

- ❖ When block is characterized, `iso_source` attribute is set for inputs and `iso_sink` attribute is set for outputs, ONLY in case of block level isolation strategies with location SELF.
- ❖ If the source/sink supply set is not available in the block (not even as a reference only supply set), then VC LP sets the `DERIVED_DIFF_ONLY` or `DERIVED_DIVERSE` as the value of the attribute.

- ◆ DERIVED\_DIVERSE: If the global net crossing the pin is not eligible for source/sink isolation insertion (reconverging fanouts, multiple drivers, inout ports, and so on), then DERIVED\_DIVERSE is set.
- ◆ DERIVED\_DIFF\_ONLY: If the source supply is different than all the sink supplies, then DERIVED\_DIFF\_ONLY is set.

Because multiple driven scenarios are not expected in design, only iso\_sink with value DERIVED\_DIFF\_ONLY/ DERIVED\_DIVERSE on output port impacts policy association. For value DERIVED\_DIVERSE, the strategy cannot be implemented when one of the following condition exists.

- The strategy is not path specified (with no -source/-sink/-diff\_supply\_only)
- Strategy is with -sink option.
- Strategy is with -diff\_supply\_only option

For value of DERIVED\_DIFF\_ONLY, the strategy cannot be implemented when one of the following condition exists.

- ❖ The strategy is not diff\_supply\_only based.
- ❖ The strategy is with option -diff\_supply\_only but also with option -sink.

The ISO\_STRATEGY\_DROPPED violation reports the DERIVED\_DIVERSE and DERIVED\_DIFF\_ONLY reason codes to indicate strategies dropped due to above usage scenarios. The same reason code will be seen in *vcst\_rtdb/lpdb/debug\_reports/DroppedPolicyReport.rpt*.

### 5.3.23.2 Supply Net Resolution in Presence of set\_port\_attributes

Whenever driver or receiver logic is available (in case of intermediate boundary ports which are not black box instances), the corresponding supply is used to resolve the related supply information. If the logic is not available (for top level ports), then following precedence is followed for the supply net resolution.

The isolation engine uses the following precedence for supply set resolution:

*iso\_source/iso\_sink port attributes > driver\_supply/receiver\_supply -port > driver\_supply/receiver\_supply -element > set\_related\_supply\_net > related supply as domain's primary*

The level shifter policy association and rail order checks (used by `check_lp -stage design` or `check_lp -stage upf`) takes the set\_port\_attribute information in following precedence:

*receiver\_supply -port > receiver\_supply -element > set\_related\_supply\_net > related supply as domain's primary*



**Note**  
The supply net resolution for level shifter checks does not consider the iso\_source or iso\_sink attribute. Unlike SRSN, these set port attributes on the libCell pin is ignored for supply net resolution.

### 5.3.23.3 Limitation

VC LP does not support the iso\_source and iso\_sink attributes for inout ports.

### 5.3.24 Support for set\_port\_attributes with -applies\_to and without -elements

Previously, Synopsys did not allow the use of set\_port\_attributes command with the -applies\_to option without the -elements option. But, UPF LRM does not specify any such restriction, therefore, starting with release, VC LP allows such SPA command.

Previously, when set\_port\_attributes UPF command with "-applies\_to" option but without the -elements option, Plato reports the flags error message.

For example:

```
set_port_attributes -ports $cpu_clamphigh_ports -applies_to outputs -clamp_value 1
```

Following error is flagged:

*TCL\_OPT\_DEPENDS: Incorrect command options*

*'-applies\_to' requires that '-elements' must be specified in command 'set\_port\_attributes'.*

The following behavior for SPA -applies without -elements.

SPA Command	Previous Behavior	New Behavior
SPA -ports <port list> -applies_to <>	Error  set_port_attributes -ports { clk_A } - applies_to outputs -driver_supply my_A - receiver_supply my_A  [Error] TCL_OPT_DEPENDS: Incorrect command options  '-applies_to' requires that '-elements' must be specified in command 'set_port_attributes'.	No Error (Filter out ports from the port list)
SPA -ports <> -applies_to <> - exclude_elements<>	Error  set_port_attributes -ports { clk_A } - applies_to outputs -exclude_elements {DOM_A_inst} -driver_supply my_A - receiver_supply my_A  [Error] TCL_OPT_DEPENDS: Incorrect command options  '-applies_to' requires that '-elements' must be specified in command 'set_port_attributes'.	No error (Exclude ports specified in exclude_elements & filter out ports whose direction is not consistent with any of the directions identified by the -applies_to option)
SPA -ports <> -applies_to <> -exclude_ports<>	Error  set_port_attributes -ports { clk_A } - applies_to outputs -exclude_ports {clk_A} - driver_supply my_A -receiver_supply my_A  [Error] TCL_OPT_DEPENDS: Incorrect command options  '-applies_to' requires that '-elements' must be specified in command 'set_port_attributes'.	No error

SPA Command	Previous Behavior	New Behavior
SPA -ports <> -elements <> -applies_to <>	No Error  set_port_attributes -elements {DOM_B_inst} -ports { clk } -applies_to outputs -driver_supply my_temp - receiver_supply my_temp	No Error  As per LRM, "It shall be an error if -ports is specified and -elements is also specified." Currently plato did not flag error when "-ports" and "-elements" are both specified. Discussed with Xteam, no new error message should be introduced in this project.
SPA -model <models> -applies_to <>	Error  set_port_attributes -model {DOM_A } - applies_to outputs -driver_supply my_A - receiver_supply my_A  [Error] TCL_OPT_DEPENDS: Incorrect command options  '-applies_to' requires that '-elements' must be specified in command 'set_port_attributes'.	No error, filter out ports of the macro according to -applies_to
SPA -model <models> -ports <port list> -applies_to <>	Error  set_port_attributes -model {DOM_A } -ports {clk_A } -applies_to outputs -driver_supply my_A -receiver_supply my_A  TCL_OPT_DEPENDS: Incorrect command options  '-applies_to' requires that '-elements' must be specified in command 'set_port_attributes'.	No error (Filter out ports whose direction is not consistent with any of the directions identified by the - applies_to option)
SPA -model <models> -ports <port list> -exclude_ports<> -applies_to<>	Error  set_port_attributes -model {DOM_A } -ports {clk_A } -exclude_ports {clk_A } -applies_to outputs -driver_supply my_A - receiver_supply my_A  TCL_OPT_DEPENDS: Incorrect command options  '-applies_to' requires that '-elements' must be specified in command 'set_port_attributes'.	No error  The condiate ports of the model will be filtered out based on -exclude_ports and then considers the applies_to option.

SPA Command	Previous Behavior	New Behavior
SPA -model <models> -elements<> -applies_to<>	Error  set_port_attributes -model {DOM_A } -elements {DOM_B_inst} -applies_to outputs -driver_supply my_A -receiver_supply my_A  [Error] TCL_OPT_ATMOST_ONE_OF: Incorrect command options  It is not legal to specify more than one of '(-model, -elements)' in command 'set_port_attributes'.  [Error] TCL_OPT_DEPENDS: Incorrect command options  '-applies_to' requires that '-elements' must be specified in command 'set_port_attributes'.	Error  As per LRM, "It shall be an error if -model is specified and -elements is also specified."  Flag "TCL_OPTION_ATMOST_OF _ONE_OF" only, and remove error message "TCL_OPT_DEPENDS" for this case
SPA -applies_to <>	Error  set_port_attributes -applies_to outputs -driver_supply my_A -receiver_supply my_A  [Error] TCL_OPT_DEPENDS: Incorrect command options '-applies_to' requires that '-elements' must be specified in command 'set_port_attributes'.	Error  TCL_OPT_ATLEAST_ONE_O F set_port_attributes -applies_to requies atleast one of "-model" or "-elements" or "-ports".
spa -exclude_elements/-exclude_ports -applies_to	Error  [Error] TCL_OPT_DEPENDS: Incorrect command options  '-applies_to' requires that '-elements' must be specified in command 'set_port_attributes'.	Error  test.upf:45: [Error] TCL_OPT_ATLEAST_ONE_O F: Incorrect command options  Atleast one of '(-elements, -ports, -model)' must be specified in command 'set_port_attributes'.

### 5.3.25 Support for set\_port\_attributes -is\_analog Attribute

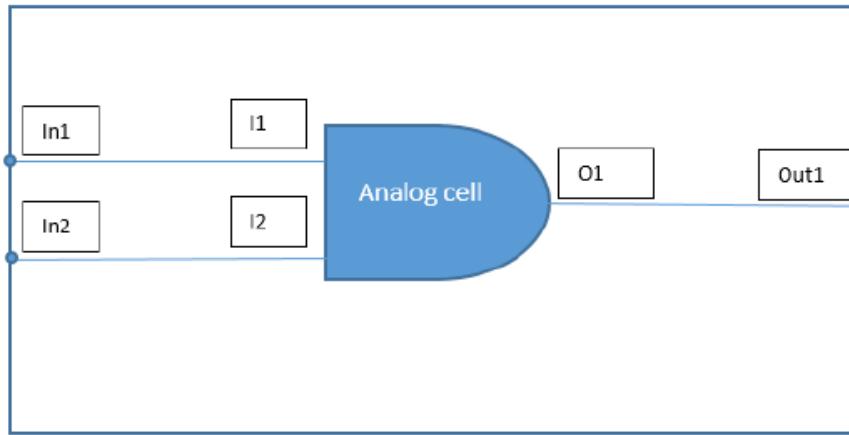
The set\_port\_attributes -is\_analog support is introduced for the top level ports. Using the set\_port\_attribute -is\_analog option, the top level ports can be marked as analog ports. This helps in removing the possible noise present in the top level ports.

#### Use Model

```
set_port_attributes -is_analog -ports <ports>
```

#### Example

Consider the following diagram. All the cell ports of the instance analog cell are analog ports. The cell connections are with top level ports.



Consider the following UPF:

```
set_port_attributes -is_analog -ports {In1 Out1}
```

For this example, one ANALOG\_NET\_INCORRECT violation is reported between In2 and I2.

#### Note

Currently tool supports both top level and hierarchical level ports in the -ports option. Hierarchical level ports support will be removed in a future patch release. False ANALOG\_PIN\_INCORRECT violations are reported for SPA -is\_analog for top level in-out ports. The ANALOG\_PIN\_INCORRECT violation is disabled by default.

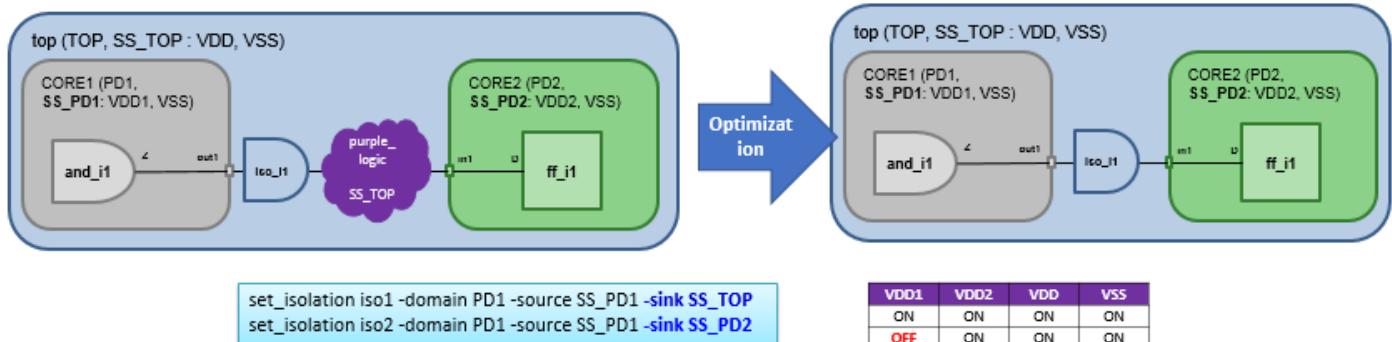
### 5.3.26 Support for Name Based ISO Strategy Association

The implementation tools might change/optimize the crossover paths, thereby changing the source/sink logic. This might cause the mismatch of source/sink logic before and after optimization, and can change the strategies applicable on a port. Hence, the isolation cell association cannot rely on the source/sink supply.

In such cases, VC LP enables you to use the name based isolation association. The isolation cell name is used to match with the isolation strategy. Use the following isolation cell naming convention for the particular strategy:

PREFIXsnps\_PDNAME\_\_STRATEGYNAME\_snps\_PINNAME\_NPOSTFIX

Consider the example design as shown in [Figure 5-95](#). Normally, without optimization, when the purple\_logic is working on supply SS\_TOP is present on the crossover, VC LP associates strategy *iso1* with the *iso\_i1* cell.

**Figure 5-95 Name Based ISO Strategy Association**

If during optimization, the purple\_logic is removed, then post optimization VC LP associates strategy iso2 with the cell iso\_i1 which might not be user intent.

If you specify the isolation cell instance name as snps\_PD1\_\_iso1\_snps, then VC LP associates strategy iso1 even in the absence of the purple\_logic.

### 5.3.27 Support for Name Based Association for ELS Cells

VC LP supports name-based association for ELS cells under the enable\_iso\_name\_assoc application variable. The enable\_iso\_name\_assoc application variable is disabled by default. Set the enable\_iso\_name\_assoc to true to enable name-based association for ELS cells.

The simple isolation cell uniform name is:

```
<iso_name_prefix>snps<power_domain_name>_<iso_strategy_name>_snps<pin_name><inst_index>
<iso_name_suffix>
```

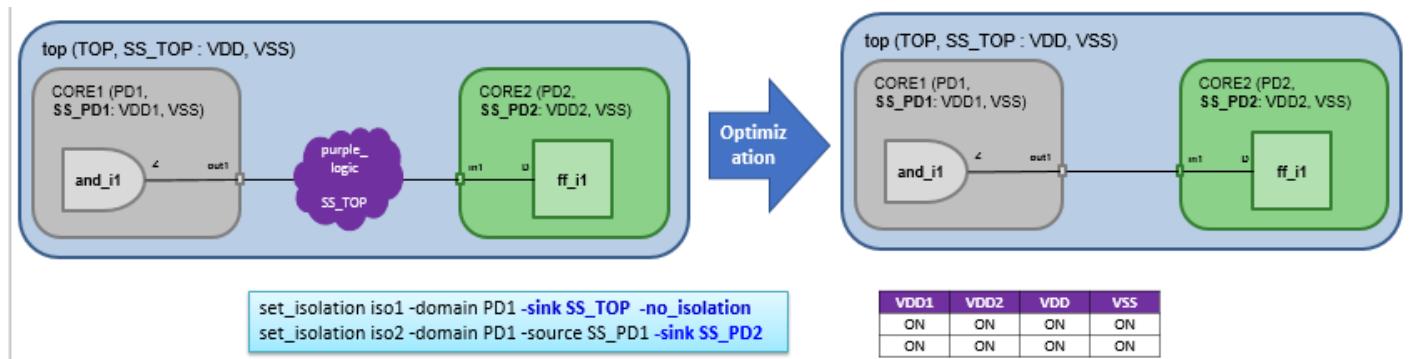
The ELS cell uniform name is:

```
<ls_name_prefix>snps<ls_power_domain_name>_<ls_strategy_name>_snps<iso_name_prefix>snps
<iso_power_domain_name>_<iso_strategy_name>_snps<pin_name>_<iso_name_suffix><ls_name_su
ffix>
```

### 5.3.28 Support for resolved\_iso\_strategy in set\_port\_attribute

The name based isolation association works fine for cases where ISO instance is present on path. For the cases where the ISO cell is not present on the path, the strategy which gets associated with the boundary node might get changed because of the optimization.

In the example in [Figure 5-96](#), the -no\_isolation strategy iso1 is associated with boundary port CORE1/out1 before optimization. After optimization, the strategy iso2 is associated with boundary port CORE1/out1.

**Figure 5-96 Example for resolved\_iso\_strategy in set\_port\_attribute**

For such cases, you can use the `set_port_attributes -ports` command to specify the isolation strategy association. The UPF command ensures the strategy iso1 gets associated with the boundary port CORE1/out1.

```
set_port_attributes -ports CORE1/out1 -attribute resolved_iso_strategy PD1__iso1
```

If you do not want to associate any strategy with a particular boundary port, then attribute value can be specified as `smps_none`.

```
set_port_attributes -ports CORE1/out1 -attribute resolved_iso_strategy smps_none
```

### 5.3.28.1 Parser Messages Introduced for set\_port\_attribute -attribute resolved\_iso\_strategy

- ❖ If the port specified in the `-ports` option of the SPA command do not exists in design, VC LP reports the following parsing message `UPF_OBJECT_NOT_FOUND_WARN` parser warning message.

#### Example

```
set_port_attributes -ports {CORE1/out1} -attribute resolved_iso_strategy PD1_is01
```

VC LP reports the following parsing message:

[Warning] `UPF_OBJECT_NOT_FOUND_WARN: Object not found`

*Object 'CORE1/out1' of type 'Port | Pin' specified with command option 'set\_port\_attributes -ports' could not be resolved. Resolved Path 'Instance /CORE1/ff1:upf\_object\_not\_found\_spa\_ports.v:11'. Unresolved Path 'out1'.*

*Please specify a valid object.*

- ❖ The `resolved_iso_strategy` attribute can take two values : `smps_none` or `PDNAME__STRATEGYNAME`. If you do not follow the specified format, VC LP reports the `INCORRECT_VALUE_OF_SPA_RESOLVED_ISO_ATTRIBUTE` parser message:

#### Example

```
set_port_attributes -ports {CORE1/out1} -attribute resolved_iso_strategy PD1_is01
```

VC LP reports the following parsing message:

[Error] `INCORRECT_VALUE_OF_SPA_RESOLVED_ISO_ATTRIBUTE: Attribute value 'PD1_is01' specified in 'set_port_attributes' command with attribute name 'resolved_iso_strategy' does not follow recommended format, attribute ignored for rule checking`

*Please specify attribute value as `smps_none` or in <domain-name>\_<strategy-name> format*

- ❖ If a strategy provided in the attribute value does not exist, then VC LP reports the STRATEGY\_NOT\_FOUND\_IN\_SPA\_RESOLVED\_ISO\_ATTRIBUTE message.

#### **Example**

The strategy iso3 do not exist in UPF file for power domain PD1, and the following UPF command is present:

```
set_port_attributes -ports {CORE1/out1} -attribute resolved_iso_strategy PD1__iso3
```

VC LP reports the following parsing message:

*[Error] STRATEGY\_NOT\_FOUND\_IN\_SPA\_RESOLVED\_ISO\_ATTRIBUTE: Isolation strategy 'iso3' specified in 'set\_port\_attributes' command with attribute name 'resolved\_iso\_strategy' does not exist for domain 'PD1', attribute ignored for rule checking*

*Please specify valid isolation strategy defined for the domain*

- ❖ If the domain name does not exist, then VC LP reports the DOMAIN\_NOT\_FOUND\_IN\_SPA\_RESOLVED\_ISO\_ATTRIBUTE parsing message.

#### **Example**

The power domain PD3 does not exist in the UPF, but the following UPF command is specified:

```
set_port_attributes -ports {CORE1/out1} -attribute resolved_iso_strategy PD3__iso
```

VC LP reports the following parsing message:

*[Error] DOMAIN\_NOT\_FOUND\_IN\_SPA\_RESOLVED\_ISO\_ATTRIBUTE: Domain 'PD3' specified in 'set\_port\_attributes' command with attribute name 'resolved\_iso\_strategy' does not exist, attribute ignored for rule checking*

*Please specify valid domain name*

- ❖ If strategy specified in the SPA command does not have -no\_isolation, then VC LP reports the INCORRECT\_STRATEGY\_IN\_SPA\_RESOLVED\_ISO\_ATTRIBUTE parsing message.

#### **Example**

The strategy iso does not have -no\_isolation:

```
set_port_attributes -ports {CORE1/out1} -attribute resolved_iso_strategy PD1__iso
```

VC LP reports the following parsing message:

*[Error] INCORRECT\_STRATEGY\_IN\_SPA\_RESOLVED\_ISO\_ATTRIBUTE: Isolation strategy 'iso' specified in 'set\_port\_attributes' command with attribute name 'resolved\_iso\_strategy' does not contain 'no\_isolation'*

*It is expected that strategy specified in attribute 'resolved\_iso\_strategy' contains no\_isolation*

- ❖ If the port specified in the -ports of SPA command does not belong to the power domain specified, VC LP reports the UPF\_SPA\_PORT\_NOT\_ON\_DOMAIN\_IN\_SPA\_RESOLVED\_ISO\_ATTRIBUTE parser message.

#### **Example**

The following command port CORE2/in1 does not belong to power domain PD1:

```
set_port_attributes -ports {CORE2/in1} -attribute resolved_iso_strategy PD1__iso
```

VC LP reports the following parsing message:

[Error] UPF\_SPA\_PORT\_NOT\_ON\_DOMAIN\_IN\_SPA\_RESOLVED\_ISO\_ATTRIBUTE: Port 'CORE2/in1' specified in 'set\_port\_attributes -ports' command does not belong to domain 'PD1' specified with attribute name 'resolved\_iso\_strategy', attribute ignored for rule checking

Port specified in -ports should belong to domain specified in value of attribute resolved\_iso\_strategy

- ❖ If there are multiple SPA command for a particular port, VC LP reports the UPF\_ATTR\_VALUE\_OVERRIDE parsing message, and considers the latter one.

#### Example

```
set_port_attributes -ports {CORE1/out1} -attribute resolved_iso_strategy PD1_iso
set_port_attributes -ports {CORE1/out1} -attribute resolved_iso_strategy PD1_isol
```

[Warning] UPF\_ATTR\_VALUE\_OVERRIDE: Overriding attribute value

Attribute 'resolved\_iso\_strategy' has already been applied on 'top/CORE1/b' with value 'PD1\_iso' at upf\_attr\_value\_override.upf,50. Replacing with attribute value 'PD1\_isol'.

Please remove the invalid value to remove the warning.

### 5.3.29 Support for -feedthrough and unconnected Attributes in set\_port\_attributes

Prior to L-2016.06, VC LP was not able to recognize unconnected and feedthrough pins present inside lib cell. VC LP was considering such lib pins as the end point for LP analysis and it was causing unnecessary ISO/LS requirements.

VC LP supports -feedthrough/-unconnected attributes in set\_port\_attributes command to define unconnected and feedthrough lib pins present inside lib cell.

#### 5.3.29.1 Feedthrough Modeling

A port or pin can be modeled as feedthrough using the UPF based modeling and set\_port\_attributes - unconnected modeling.

##### 5.3.29.1.1 Use Model for UPF based modeling

###### Use Model

```
set_port_attribute -model M -feedthrough -ports {A Z}
```

Where:

- ❖ The -model can refer to macro, pad library cells or black box elements.
- ❖ The -ports are normal pin/ports of a model. It cannot refer any functional pins (like is\_isolated, save, restore, and so on)

When you run the above upf command, the crossover goes through for feedthrough ports (does not stop at feedthrough ports) and the related\_power\_pin/related\_ground\_pin is set to null.

#### Strategy and Crossover Handling for set\_port\_attributes -feedthrough

- ❖ All port-based policies on feedthrough ports are applied normally, as done for all other ports, even though it is a feedthrough port.
- ❖ The path-based strategy is applied as per the source and sink computation after feed-through information is taken into account.

Crossover continues through the feedthrough ports, and these ports/pins are not kept in crossover tree, unless there is a policy on those ports. The crossover behavior is just like any feed-through ports in hierarchical modules.

### 5.3.29.1.2 Use Model for set\_port\_attributes -unconnected

```
set_port_attribute -model M1 -unconnected -ports {B}
```

Where:

- ❖ -model can refer to macro, pad library cells, black box, user-defined module or lib cells elements.
- ❖ -ports are normal pin/ports of a model. It cannot refer any functional pins (like is\_isolated, save, restore and so on).

When you run the above UPF command, the crossover stops earlier for the unconnected ports, and the related\_power\_pin/related\_ground\_pin attribute is set to null.

### Strategy and Crossover Handling set\_port\_attributes -unconnected

Whether the policy is used in the analysis is based on setting of the mode handle\_hanging\_crossover application variable. However, if in the process of shortening a crossover, the specified pin is eliminated from crossover path, then the policies associated at the pin also gets eliminated from analysis.

- ❖ If handle\_hanging\_crossover is set to false, then these pins can be source or sink of a crossover supply as the default domain supply.
- ❖ If handle\_hanging\_crossover is set to true, then this port will be removed from crossover, as it is done for any other hanging net and ports for hierarchical modules.

In short, the set\_port\_attributes -unconnected command is treated in the same way as the design unconnected net.

### 5.3.29.2 Precedence Rules

- ❖ If on same port both (a) set\_port\_attribute (SPA) -feedthrough/-unconnected and (b) set\_related\_supply\_net (SRSN) is specified, then SRSN takes precedence, and SPA -feedthrough/-unconnected is ignored

*[Warning] UPF\_SPA\_FEEDUNCONN\_IGNORED: SPA '-feedthrough/-unconnected' on port will be ignored.*

*set\_port\_attribute '-feedthrough/-unconnected' option specified on port 'in1' will be ignored because set\_port\_attribute or set\_related\_supply\_net has already specified on port.*

- ❖ Similarly, if set\_port\_attribute -driver\_supply/-receiver\_supply/-repeater\_supply is specified, and also set\_port\_attribute -feedthrough/-unconnected is specified, then feedthrough/unconnected specification will be ignored, and warning message will be issued.

*[Warning] UPF\_SPA\_FEEDUNCONN\_IGNORED: SPA '-feedthrough/-unconnected' on port will be ignored.*

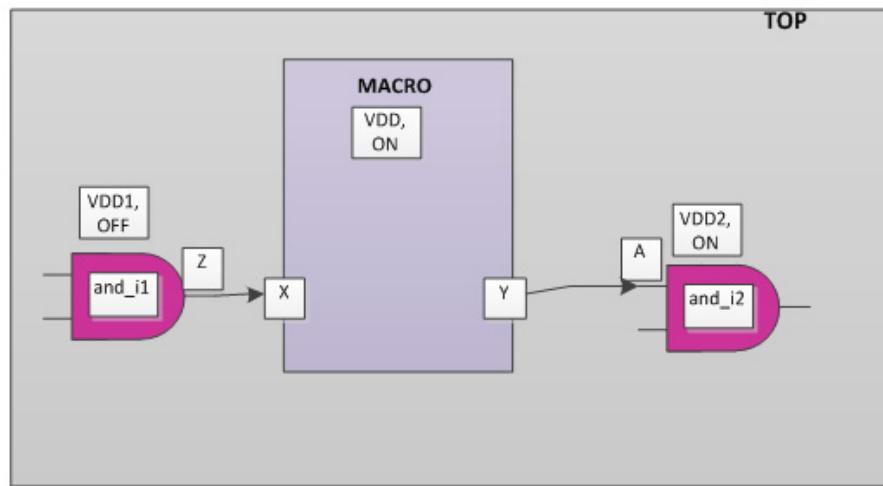
*set\_port\_attribute '-feedthrough/-unconnected' option specified on port 'in1' will be ignored because set\_port\_attribute or set\_related\_supply\_net has already specified on port.*

- ❖ If same port of a cell is specified as unconnected and feedthrough in UPF, then error message will be issued and parser stops.

*[Error] UPF\_ATTRS\_CONFLICT: Conflicting design attributes specified  
Attribute '-feedthrough' conflicts with attribute '-unconnected'*

### 5.3.29.3 Examples

**Figure 5-97 Example for set\_port\_attributes -feedthrough**



For the example in [Figure 5-97](#), if you do not specify the `set_port_attributes -feedthrough` attributes, VC LP reports ISO\_STRATEGY\_MISSING and ISO\_INST\_MISSING from and\_i1/Z to MACRO/X.

```

Tag : ISO_STRATEGY_MISSING
      Description : Isolation required on crossing from [Source] to [Sink], but
strategy missing
Violation : LP:1
Source
  PinName : and_i1/Z
SegmentSourceDomain : TOP
Sink
  PinName : MACRO/X
SegmentSinkDomain : PD1
LogicSource
  PinName : and_i1/Z
LogicSink
  PinName : MACRO/X
  
```

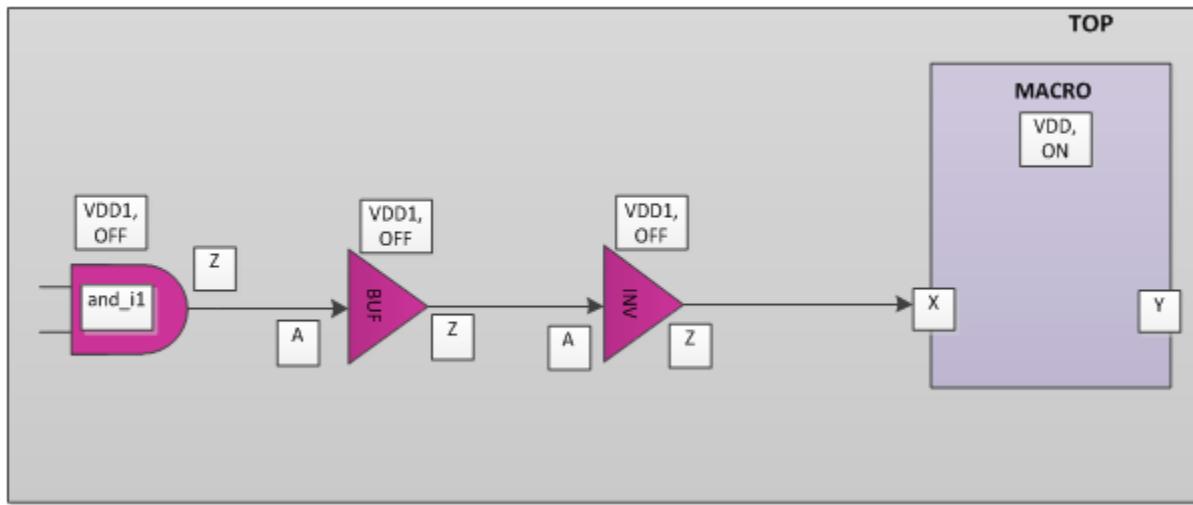
If you specify the following upf command,

```
set_port_attributes -feedthrough -model M -ports {X Y} // M is macro cell name
```

then VC LP reports ISO\_STRATEGY\_MISSING and ISO\_INST\_MISSING from and\_i1/Z to and\_i2/A.

```

Tag : ISO_STRATEGY_MISSING
      Description : Isolation required on crossing from [Source] to [Sink], but
strategy missing
Violation : LP:1
Source
  PinName : and_i1/Z
SegmentSourceDomain : TOP
Sink
  PinName : and_i2/A
SegmentSinkDomain : TOP
LogicSource
  PinName : and_i1/Z
LogicSink
  PinName : and_i2/A
  
```

**Figure 5-98 Example for set\_port\_attributes -unconnected**

For the example in [Figure 5-98](#), if you do not specify the `set_port_attributes -unconnected` attributes, VC LP reports ISO\_INST\_MISSING and ISO\_STRATEGY\_MISSING with LogicSource: and\_i1/Z and LogicSink : MACRO/X.

```

Tag          : ISO_STRATEGY_MISSING
Description   : Isolation required on crossing from [Source] to [Sink], but
strategy missing
Violation     : LP:1
Source
  PinName      : and_i1/Z
SegmentSourceDomain : TOP
Sink
  PinName      : MACRO/X
SegmentSinkDomain  : PD1
LogicSource
  PinName      : and_i1/Z
  LogicSink    : MACRO/X
  
```

If you specify the following UPF command, and if the `handle_hanging_crossover` is set to false

```
set_port_attributes -unconnected -model M -ports {x} // M is macro cell name
```

```
set_app_var handle_hanging_crossover false
```

then VC LP reports ISO\_STRATEGY\_MISSING and ISO\_INST\_MISSING with LogicSource : and\_i1/Z, and LogicSink : MACRO/X with LogicSinkUnconnected : True

```

Tag          : ISO_STRATEGY_MISSING
Description   : Isolation required on crossing from [Source] to [Sink], but
strategy missing
Violation     : LP:1
Source
  PinName      : and_i1/Z
Sink
  PinName      : MACRO/X
SegmentSourceDomain : TOP
SegmentSinkDomain  : PD1
LogicSource
  PinName      : and_i1/Z
  LogicSink    : MACRO/X
  
```



```
LogicSinkUnconnected : True
```

---

If you specify the following upf command, and if the handle\_hanging\_crossover is set to true,  
`set_port_attributes -unconnected -model M -ports {X} // M is macro cell name`  
`set_app_var handle_hanging_crossover true`  
then VC LP does not report ISO\_\*\_MISSING violation for this crossover as MACRO/X is dropped from the crossover.

 **Note**

ISO\_STRATMISSING\_NOBOUNDARY is reported for cases where an Isolation Strategy is not provided, but needed, on a port that is not on a power domain boundary. Typically this happens when you have a Macro/LDB living inside/below a power domain and a crossover was detected to/from the Macro/LDB boundary, which does not line up with a power domain boundary (that is, the macro instance was not put in the elements list of the power domain or a design attribute like lower\_domain\_boundary is not specified). ISO\_STRATEGY\_MISSING is flagged for cases where an Isolation Strategy is not provided, but needed, on a port that IS ON a power domain boundary. Therefore, the scenarios covered by ISO\_STRATMISSING\_NOBOUNDARY will not be covered by ISO\_STRATEGY\_MISSING violation.

### 5.3.30 Support for Feedthrough/Unconnected SPA Attribute Block Level Consistency Check

Currently, for SPA function attribute (buffered/inverted feed through), unconnected and feed through, you can only specify black box, macro pad, and libcell using the -model option. The TOP module is not allowed, it throws a warning and attributes are not retained. During block level validation, the specified models at the -model at top level will become the TOP module. Therefore, during block level validation, the TOP module should be allowed to be specified for -model option for SPA for the mentioned attributes to be applied to TOP module. Starting with this release, the attributes feed through, function, unconnected with model TOP will be read on block level flat run and if attributes are not matching design scenarios and the following violations are reported.

The following new warning violations are introduced for these scenarios.

- ❖ UPF\_SPAUNCONN\_MISSING: TOP level output port does not have a driver and without -unconnected specification.
- ❖ UPF\_SPAUNCONN\_MISSING: TOP level input port does not have a load and without -unconnected specification.
- ❖ UPF\_SPAFEED\_MISSING: Wired feed through without UPF SPA -feedthrough specification.
- ❖ UPF\_SPAFUNC\_MISSING: Buffered feedthrough without SPA -function specification.
- ❖ UPF\_SPAUNCONN\_INCORRECT: Top port has a valid driver, but `set_port_attributes -unconnected` is set in UPF
- ❖ UPF\_SPAUNCONN\_INCORRECT: Top port has a valid load, but `set_port_attributes -unconnected` is set in UPF
- ❖ UPF\_SPAFEED\_INCORRECT: Top ports do not form wire feedthrough, but `set_port_attributes -feedthrough` is set in UPF
- ❖ UPF\_SPAFUNC\_INCORRECT: Top ports do not form powered feedthrough, but `set_port_attributes -attribute` function is set in UPF

These tags are disabled by default. To enable the checks, specify using the `configure_lp_tag -tag <tag name> -enable`.

These checks are performed at `check_lp -stage` under a new family *HierConsistency*. The VC-LP-ULTRA license is required to invoke these checks. The license will be checked out `check_lp -stage upf` if any of the four checks are enabled through the `configure_lp_tag` command.

This flow helps to improve the block level verification by adding more information to the UPF and makes the block ready for black box sign off.

### Example

```

Tag           : UPF_SPAUNCONN_MISSING
Description   : Top port [TopLevelPort] has no driver. Please model it using
set_port_attributes -unconnected
Violation    : LP:4
TopLevelPort : inout1
DesignDirection : INOUT

Tag           : UPF_SPAUNCONN_MISSING
Description   : Top port [TopLevelPort] has no load. Please model it using
set_port_attributes -unconnected
Violation    : LP:5
TopLevelPort : save

Tag           : UPF_SPAFUNC_MISSING
Description   : Top port [TopLevelPortList] form powered feedthrough
[FeedthroughType]. Please model it using set_port_attributes -attribute function
Violation    : LP:7
TopLevelPortList
  TopLevelPort : in1
  TopLevelPort : out1
FeedthroughType : Buffer

```

### Limitations.

- ❖ UPF\_FEEDTHRU\_MODEL\_TYPE\_INVALID and UPF\_INVALIDMODEL parser errors will be flagged when top module is specified in SPA -model. They can be downgraded to warning from below commands.

```

configure_read_tag -family upf -tag UPF_FEEDTHRU_MODEL_TYPE_INVALID -severity warning
configure_read_tag -family upf -tag UPF_INVALIDMODEL -severity warning

```

### 5.3.31 UPF 2.0/2.1 Enhancements in Commands

VC LP supports the following UPF 2.0/2.1 options in the following UPF commands.

- ❖ The `-update` option with `-elements` in `create_power_domain`, `set_isolation`, `set_level_shifter` and `set_retention` UPF commands
- ❖ The `-update` option with `-exclude` elements in `set_isolation` and `set_level_shifter` UPF commands
- ❖ The `-exclude_elements` option in `set_isolation` and `set_level_shifter` UPF commands.
- ❖

Also, the `-exclude_elements` option is added in `report_isolation_strategy` and `report_level_shifter_strategy` VC LP Tcl commands.

### 5.3.31.1 Enhancements to the `create_power_domain` Command

Previously, you could add only supplies using the `-update` option. Now, you can also add elements using the `-update` option.

#### Syntax

```
create_power_domain domain_name
[-elements element_list]
[-update]
```

- ❖ `-update`: Use `-update` to add elements to a previously created domain.

#### Examples

```
create_power_domain TOP -include_scope
create_power_domain PD1 -elements {CORE1}
create_power_domain PD1 -elements {CORE2} -update
```

### 5.3.31.2 Enhancements to the `set_isolation` Command

VC LP supports `-exclude_elements` option and also `-update` option with `-elements` and `-exclude_elements` option in the `set_isolation` strategy.

#### Syntax

```
set_isolation strategy_name
-domain domain_name
[-elements element_list]
[-exclude_elements exclude_list]
[-update]
```

- ❖ `-exclude_elements exclude_list`: A list of instances or ports to which the strategy does not apply
- ❖ `-update`: Use `-update` to add elements in `-elements` and `-exclude_elements` to a previously defined isolation strategy

#### Examples:

```
set_isolation iso1 -domain PD1 -elements {CORE1/*} -exclude_elements {CORE1/out1}

set_isolation iso2 -domain PD2 -elements {CORE2} -exclude_elements {CORE2/out1}
set_isolation iso2 -domain PD2 -elements {CORE2/CORE22} -update

set_isolation iso3 -domain PD3 -elements {CORE3}
set_isolation iso3 -domain PD3 -exclude_elements {CORE3/out1} -update
```

### 5.3.31.3 Enhancement to the `set_level_shifter` Command

VC LP supports `-exclude_elements` option and also `-update` option with `-elements` and `-exclude_elements` option in the `set_level_shifter` strategy.

#### Syntax

```
set_level_shifter strategy_name
-domain domain_name
[-elements element_list]
[-exclude_elements exclude_list]
[-update]
```

- ❖ `-exclude_elements exclude_list`: A list of instances or ports to which the strategy does not apply

- ❖ -update: Use -update to add elements in -elements and -exclude\_elements to a previously defined level shifter strategy.

### Examples

```
set_level_shifter ls1 -domain PD1 -elements {CORE1/*} -exclude_elements {CORE1/out1}
set_level_shifter ls2 -domain PD2 -elements {CORE2} -exclude_elements {CORE2/out1}
set_level_shifter ls2 -domain PD2 -elements {CORE2/CORE22} -update

set_level_shifter ls3 -domain PD3 -elements {CORE3}
set_level_shifter ls3 -domain PD3 -exclude_elements {CORE3/out1} -update
```

#### 5.3.31.4 Enhancement to the set\_retention upf Command

VC LP supports -update option with -elements option in the set\_retention strategy.

### Syntax

```
set_retention retention_name
-domain domain_name
[-elements element_list]
[-update]
```

- ❖ -update: Use -update to add elements in -elements of a previously defined retention strategy.

### Examples

```
set_retention ret1 -domain PD1 -elements {CORE1/ret1}
set_retention ret1 -domain PD1 -elements {CORE1/ret2} -update
```

#### 5.3.31.5 Enhancement to the report\_isolation\_strategy and report\_level\_shifter Commands

The -excluded\_elements option is added in report\_isolation\_strategy and report\_level\_shifter\_strategy Tcl commands. Use this option to get a report of the excluded elements.

### Examples

```
%vc_static_shell> report_isolation_strategy -exclude_elements
Isolation Strategy : iso1
Exclude Elements   : CORE2
%vc_static_shell> report_level_shifter_strategy -exclude_elements
Level Shifter Strategy : ls1
Exclude Elements       : CORE2
```

#### 5.3.32 Support for set\_retention/set\_isolation UPF 3.0 Versions

VC LP supports UPF 3.0 version of the following commands. You can specify any version or mixed version of the following commands:

UPF 2.0/2.1	UPF 3.0
set_retention [-retention_supply_set ret_supply_set]	set_retention [-retention_supply ret_supply_set]
set_isolation [-isolation_supply_set supply_set_list]	set_isolation [-isolation_supply supply_set_list]



This feature has Backward compatibility. So user can use both style

### Limitations

The Pixl2 parser does not support this feature. Only the Plato parser is supports this feature.

### 5.3.33 Support for -update option for -source and -sink option in set\_isolation command

VC LP supports the -update option for the -source and -sink option in the set\_isolation command.

#### Example

```
set_isolation ISO1 -domain PD_SUB -elements {uSUB1/in1} -isolation_signal iso_en -  
isolation_sense low -clamp_value 1  
set_isolation ISO1 -domain PD_SUB -source PD_DUT.primary -sink PD_SUB.primary -update
```

Since this not supported by all SNPS tools, a parser message is introduced to notify the user.

*[Info] UPF\_ISOLATION\_SOURCE\_SINK\_UPDATED: unset source/sink filter of isolation updated  
Unset 'source' filter of isolation strategy: 'ISO1' has been updated.  
This feature might not be supported across all tools of Synopsys.*

In addition, if you update the previously defined source sink values, the following error message is reported:

*[Error] UPF\_ISOLATION\_SOURCE\_SINK\_UPDATE\_INVALID: source/sink filter of isolation cannot be updated  
The '<source/sink>' filter of isolation strategy: '<ISO strategy>' has already been set with a different value in <UPF file>.   
Cannot update source/sink filter if already set in a previous set\_isolation command.*

### 5.3.34 Support for clamp\_value in set\_port\_attributes Command

The -clamp\_value option is supported in the set\_port\_attributes command. Using the set\_port\_attribute -clamp\_value command, you can define clamp\_value on the isolation strategy ports.

The set\_port\_attribute -clamp\_value command simplifies the UPF for ports with the same isolation strategy but with just different clamp\_value, and makes the clamp\_value independent of the isolation.

The set\_port\_attribute -clamp\_value does not impact isolation strategy applied to a port, but only the clamped value of the port. The clamp\_value specified by the port attribute is looked upon as the hard constraint and it has precedence over the one specified by isolation strategy.

- ❖ If clamp\_value is set on a port, whether the associated isolation strategy has the -clamp\_value or not, the clamp\_value set in set\_port\_attribute will be used.
- ❖ If set\_port\_attribute -clamp\_value is not defined, the clamp\_value defined in the associated isolation strategy defined is used.
- ❖ If both set\_port\_attribute -clamp\_value and the associated isolation strategy has no -clamp\_value defined, then the default value:0 is used.

#### Syntax

```
set_port_attributes      # set_port_attributes  
[-ports port_list]  
[-elements <element_list>]  
[-applies_to direction]  
[-clamp_value value]    (indicates the value that specified ports shall be clamped to:  
Values: 0, 1, latch)
```

The following are examples of set\_port\_attribute -clamp\_value used with -ports, -elements, and -applies\_to options.

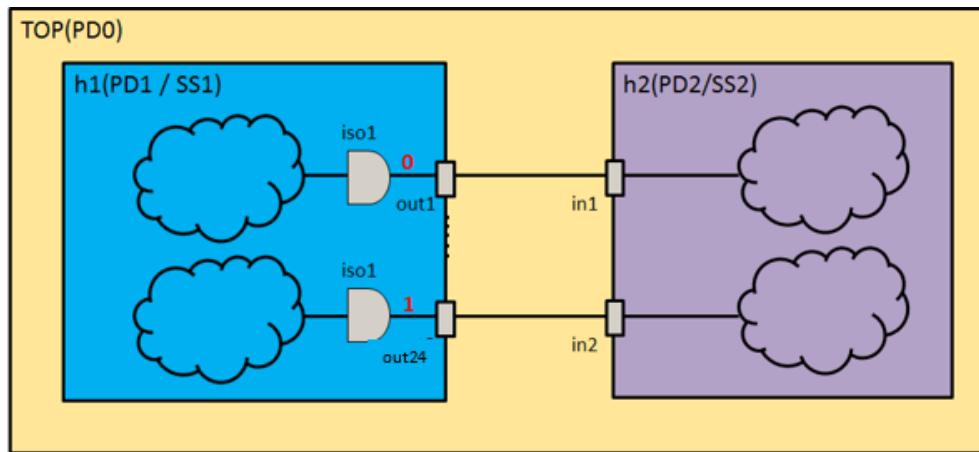
```
set_port_attributes -ports {CORE1/out1} -clamp_value 0 //Clamp Value 0 is defined on the port CORE1/out1
```

```
set_port_attributes -elements {CORE1} -clamp_value 0 -applies_to outputs //Clamp value 0 is defined on output port of element CORE1
```

### 5.3.34.1 Example

Consider the example as shown in [Figure 5-99](#).

**Figure 5-99 clamp\_value in set\_port\_attributes**



**Note**  
Without this feature, you had to define two ISO strategies so as to have different clamp value for ports, using the SPA -clamp\_value option, you need to define only one isolation strategy with different SPA - clamp\_value.

The following UPF defined for the example in [Figure 5-99](#), where different clamp\_value is defined for SPA.

```
set_isolation iso1 -domain PD1 -location self -applies_to output
set_port_attributes -elements {h1} -applies_to outputs -clamp_value 0
set_port_attributes -ports {h1/out24} -clamp_value 1
```

The following is the result of the ISO Strategy:

- ❖ All ports are associated to ISO strategy iso1.
- ❖ port h1/out24 is clamped at 1, while all other output ports are clamped at 0 of cell h1.

### 5.3.34.2 Checks Added

The following new checks are introduced for this support:

- ❖ ISO\_CLAMP\_DEFAULT

If a strategy node has no -clamp\_value defined in the isolation strategy and if SPA is not defined, the default clamp\_value 0 is considered and VC LP reports ISO\_CLAMP\_DEFAULT.

The severity of this violation is *warning* and it is enabled by default. To disable this tag, use the following command:

```
configure_tag -app LP -tag ISO_CLAMP_DEFAULT -disable
```

#### Example of the ISO\_CLAMP\_DEFAULT Violation

```

Tag          : ISO_CLAMP_DEFAULT
Description   : Strategy [Strategy] default clamp value 0 is set for node
[StrategyNode]
  Violation    : LP:1
  Strategy     : PD1/iso1
  StrategyNode : CORE1/out2

```

❖ ISO\_CLAMP\_OVERRIDE

If a isolation strategy node has clamp\_value defined in strategy and if the SPA clamp is defined on same ports, SPA clamp\_value takes precedence and VC LP reports ISO\_CLAMP\_OVERRIDE.

The severity of this violation is *Error* and it is enabled by default. To disable this tag, use the following command:

```
configure_tag -app LP -tag ISO_CLAMP_OVERRIDE -disable
```

#### Example of the ISO\_CLAMP\_OVERRIDE Violation

```

Tag          : ISO_CLAMP_OVERRIDE
Description   : Strategy [Strategy] clamp value is overrided to [UPFClampValue] at
[StrategyNode] by set_port_attribute
  Violation    : LP:1
  Strategy     : PD1/iso1
  UPFClampValue : 1
  StrategyNode : CORE1/out2
  StratgyClampValue : 0

```

#### 5.3.34.3 New debug field added

With this support, the *ClampFromSPA == true* debug field is added to the ISO\_CLAMP\_INVERT and ISO\_STRATCLAMP\_MISMATCH violations. This enables you to debug if the clamp\_value from SPA is considered or not.

#### 5.3.35 Support for -exclude\_elements/-exclude\_ports in set\_port\_attributes

To make VC LP compatible with UPF 3.0, VC LP supports -exclude\_elements and -exclude\_ports options in set\_port\_attributes command.

The ports/elements mentioned in the -exclude\_elements/-exclude\_ports are removed from the current set\_port\_attributes candidate list.

##### Example

In the following example, VC LP honors SPA on all the port list of current scope except in1.

```
set_port_attributes -ports {*} -exclude_ports {in1} -driver_supply SS1
```

If the entire object list of -exclude\_elements/-exclude\_ports is not present in design, then VC LP reports the following error message:

*[Error] UPF\_OBJECTS\_NOT\_FOUND: No valid objects in list*

*Object List specified with command option 'set\_port\_attributes -exclude\_ports' resolved to an empty list.*

If some of the object in -exclude\_elements/-exclude\_ports list are invalid, then VC LP reports the following warning message:

*[Warning] UPF\_OBJECT\_NOT\_FOUND\_WARN: Object not found*

*Object of type 'Port | Pin' specified with command option 'set\_port\_attributes -exclude\_ports' could not be resolved.*

### 5.3.36 Support for -literal\_supply in set\_port\_attributes

VC LP supports the `-literal_supply` option in the `set_port_attributes` command.

The `-literal_supply` option can be used to change the supply of the literal constant. The `-literal_supply` option of `set_port_attributes` command defines the supply to be used to model a literal value (1'b0/1'b1) associated to a pin or port marked with the attribute.

With this support, you can specify (change) the related supply of literal constants (1'b0 and 1'b1). That is, you can use a different supply for literal value instead of parent domain's supply. For example,

```
set_port_attributes -literal_supply SS_A0 -ports {OUT1 OUT2}
```

If a literal constant is associated to a pin/port attributed with literal supply, then the literal constant is assumed to be powered by the supply referenced in the `literal_supply` option, and consequently affects isolation analysis and crossover checks for the literal.

#### Syntax:

```
set_port_attributes
  [-elements elements_list]
  [-exclude_elements elements_exclude_list]
  [-ports port_list]
  [-exclude_ports port_exclude_list]
  [-applies_to inputs | outputs]
  -literal_supply supply_set_reference}
```

Notes:

- ❖ The `-model` option cannot be used with `-literal_supply`.
- ❖ At least one of `-ports` or `-elements` should be specified while using `-literal_supply`.

#### 5.3.36.1 Parser Message Introduced

VC LP reports the parser error and warning messages for the following scenarios.

- ❖ The set of ports to be attributed must not be empty. They must be obtained from processing the explicit references in `-ports` and `-exclude_ports` and the implicit references in `-elements` and `-exclude_elements`.
- ❖ For each port to be attributed, precedence resolution is done based on the precedence rules described in the LRM for the `set_port_attributes` command, but with the following exception:  
Instead of issuing error for two settings with the same precedence but different value set on a given port, the latest setting that is applied is used.
- ❖ If a port that is part of a bus is attributed with `-literal_supply`, all the ports of the bus are attributed with `-literal_supply` with the same value, otherwise an error message is reported.
- ❖ The `-literal_supply` option affects the supplies for source/sink analysis. Once the `-literal_supply` applied on port, the constant resolved PowerMethod is reported as "FROM\_UPF\_LITERAL\_SUPPLY" in violation.

#### 5.3.36.2 Examples

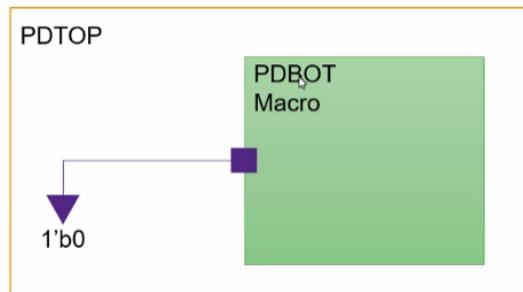
##### Example 1: Modeling Literal Constant Driver on Input Port

Consider the following UPF:

```
create_power_domain PDTOP -elements {.}
create_power_domain PDBOT -elements Macro
```

```
set_port_attributes -elements Macro \
-applies_to inputs \
-literal_supply BOT.primary
set_isolation macro_in_ISO -source TOP.primary \
-sink BOT.primary
```

**Figure 5-100 Modeling Literal Constant Driver on Input Port**



With this enhancement, literal is powered by BOT.primary and isolation strategy macro\_in\_ISO does not apply.

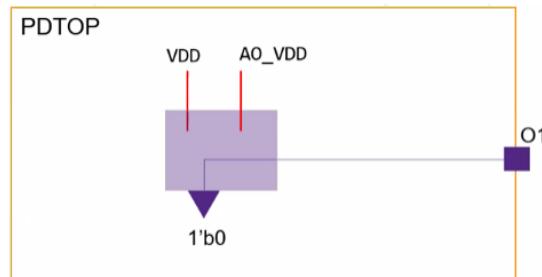
PDTOP.primary	PDBOT.primary	VSS
ON	ON	GND
OFF	ON	GND

### Example 2: Constant Driving Primary Output Port

Consider the following UPF:

```
create_power_domain PDTOP -elements {}
set_port_attributes -ports O1 -reciever_supply SS_A0
set_port_attributes -ports {out1} \
-literal_supply SS_A0
```

**Figure 5-101 Constant Driving Primary Output Port**



With this support, literal constant driver is powered by SS\_A0. Literal is simulated as a Tie-cell powered by SS\_A0, so there is no isolation requirement on this path.

PDTOP.primary	PDBOT.primary	VSS
ON	ON	GND
OFF	ON	GND

### 5.3.36.3 Consistency Checks Introduced for set\_port\_attributes -literal\_supply

The UPF\_LITERAL\_STATE and UPF\_LITERAL\_VOLTAGE consistency checks are introduced for the -literal\_supply attribute at the check\_lp UPF stage. In these checks, the literal supply mentioned in set\_port\_attributes is checked against the actual supply driving the port/pin to verify whether correct supply states are mentioned in PST, and checks whether there is an ISO/LS requirement exists between the literal\_supply and the actual supply driving the pin or port.

The following violations are reported with severity error and are enabled by default.

- ❖ The UPF\_LITERAL\_STATE violation is reported when the PST has a state where the SPA literal\_supply is off, but supply driving the pin/port is on (ISO requirement).
- ❖ The UPF\_LITERAL\_VOLTAGE violation tag is reported when the PST has a state where the SPA literal\_supply, and the supply driving pin/port has a voltage mismatch (LS requirement).

The following are example snippets of the violations:

```

Tag : UPF_LITERAL_STATE
Description : SPA literal_supply off, but supply driving the constraint node
[ConstraintNode] is on
  ConstraintNode : blck1/in1
  ConstraintInfo
    PowerNet
      NetName : VDD_PD1
      NetType : UPF
    PowerMethod : FROM_UPF_POWER_DOMAIN
    GroundNet
      NetName : VSS
      NetType : UPF
    GroundMethod : FROM_UPF_POWER_DOMAIN
  LiteralInfo
    PowerNet
      NetName : VDD_LIT
      NetType : UPF
    PowerMethod : FROM_UPF_LITERAL_SUPPLY
    GroundNet
      NetName : VSS
      NetType : UPF
    GroundMethod : FROM_UPF_LITERAL_SUPPLY
  States
    State : pst1/s2
  UPFCommand
    Command : set_port_attributes
    FileName : top.upf

```

### 5.3.36.4 Specifying Literal Constant Using lp\_literal\_constant Command

You can also specify literal constants using the lp\_literal\_constant command. When the lp\_literal\_constant command is present along with the literal\_supply attribute, the attribute is honored over the command.

#### Syntax

```

vc_static_shell> lp_literal_constant -help
Usage: lp_literal_constant      # enable literal constant behavior in VC LP
       [-sink]                  (Treat literal constant as sink supply)
       [-sink_all]                (Treat literal constant as sink supply for all.)

```

```

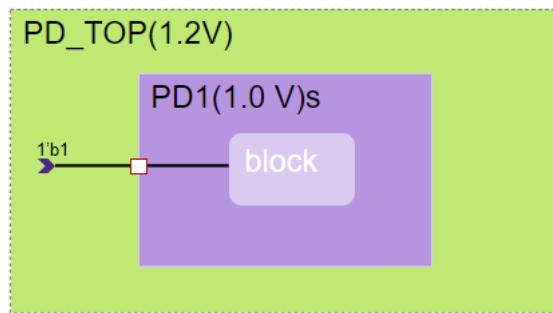
[-sink_internal]          (Treat literal constant as sink supply when sink has
internal supply)
[-flag_redundant]        (Flag Strategy Redundant on literal constant)
[-non_macro_non_pad]     (Enables Macro Or Pad for Literal Constant)
[-macro_driven]          (Enables reporting violations for literal constant driven
macro)

```

For more information on the `lp_literal_constant` command, see the man page.

### Example

The supply of the design constant needs to be considered the default supply of the power domain in which the literal constant is located.



When you set the `lp_literal_constant -sink` command, the sink supply is considered as the driver supply of literal constant, and the LS/ISO related violations are not reported, as there is no requirement of the ISO/LS cell in the path of the literal constant crossover.

### 5.3.36.5 Querying literal\_supply in set\_port\_attribute

The `-literal` option is available in the `get_upf_connection` Tcl command to check the `literal_supply` set by using the `set_port_attribute` command.

### Example

Consider the following UPF

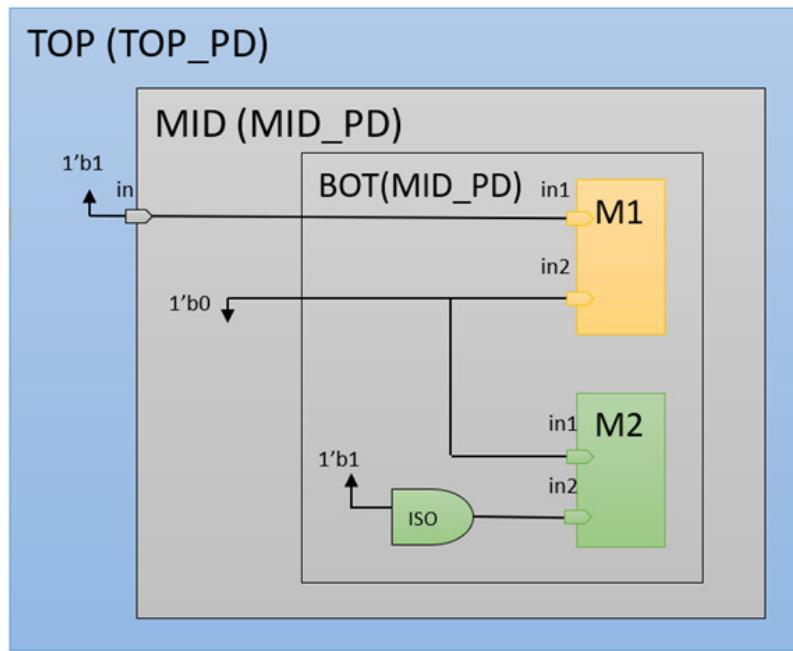
```
set_port_attributes -literal_supply {LIT_SS} -ports {blck1/in1}
```

The following Tcl command reports the `literal_supply` set by the `set_port_attribute`:

```
get_upf_connection -literal blck1/in1 -type spa
{VDD_LIT VSS}
```

### Example

Consider the following design:

**Figure 5-102 literal\_supply in the set\_port\_attribute****Example 1**

```
set_port_attributes -literal_supply {MID_PD.primary} -ports {MID/BOT/M2/in2}
```

Literal supply is considered as *MID\_PD.primary*.

**Example 2**

```
set_port_attributes -literal_supply {MID_PD.primary} -ports {MID/in}
```

```
set_port_attributes -literal_supply {TOP_PD.primary} -ports {MID/BOT/M1/in1}
```

Here, the *MID\_PD.primary* is considered as literal supply, as *Mid/in* is the nearest attributed node to the literal constant.

**5.3.37 Support for set\_port\_attributes -attribute is\_pad**

VC LP supports the *is\_pad* attribute for *set\_port\_attributes* (SPA) command. This attribute is supported only for the top scope and black box scopes.

**Example**

```
set_port_attributes -ports {in* out*} -attribute is_pad true -model { . }
```

The following new violations are introduced for the consistency check of the *is\_pad* attribute

## ❖ UPF\_SPAPAD\_INCORRECT

If the top level port has *set\_port\_attributes -is\_pad*, and is not directly connected to pad pins, the UPF\_SPAPAD\_INCORRECT violation is reported for the top level port.

## ❖ UPF\_SPAPAD\_MISSING

If the top level port does not have *set\_port\_attributes -is\_pad*, and is connected to pad pins or connected to a port of a black box with *set\_port\_attributes -is\_pad*, the UPF\_SPAPAD\_MISSING is reported for the top level port.

**Note**

If the top level port has `set_port_attributes -is_pad`, and is directly connected to pad pins, no violation is reported.

### 5.3.38 Enhancements to `set_port_attributes` on inout ports

For inout ports, if only the `-driver_supply` is specified, then VC LP derives the `-receiver_supply` according to the supply precedence. Similarly, when only the `-receiver_supply` is specified, then VC LP derives the `-driver_supply` according to the supply precedence.

- ❖ From SRSN
- ❖ From Domain's primary supply

#### 5.3.38.1 New Violations Introduced

The following tags are introduced to report violations when the `-driver_supply` and the `-receiver_supply` of a PST state are different.

These violations belong to UPF Stage, with severity *warning* and belong to the Family *UpfConsistency*. These violations are enabled by default.

##### 5.3.38.1.1 UPF\_SPAINOUT\_STATE

If there is a PST state where `-driver_supply` is OFF and `-receiver_supply` is ON or vice versa, then VC LP reports the `UPF_SPAINOUT_STATE` violation.

```

Tag : UPF_SPAINOUT_STATE
Description : Isolation needed between SPA driver supply and receiver supply for SPA
constraint on [ConstraintNode]
ConstraintNode : inout1[2]
DriverInfo
  PowerNet
    NetName : VDD_SPA_DRIVER
    NetType : UPF
  PowerMethod : FROM_UPF_DRIVER_SUPPLY
  GroundNet
    NetName : VSS
    NetType : UPF
  GroundMethod : FROM_UPF_DRIVER_SUPPLY
ReceiverInfo
  PowerNet
    NetName : VDD
    NetType : UPF
  PowerMethod : FROM_UPF_POWER_DOMAIN
  GroundNet
    NetName : VSS
    NetType : UPF
  GroundMethod : FROM_UPF_POWER_DOMAIN
UPFCommand
  Command : set_port_attributes
  FileName : spa_driver_on_inout_vector_port.upf
  LineNumber : 35

```

### 5.3.38.1.2 UPF\_SPAINOUT\_VOLTAGE

If there is a PST state where `-driver_supply` and `-receiver_supply` are working on different voltages, then VC LP reports the UPF\_SPAINOUT\_VOLTAGE violation.

```

Tag : UPF_SPAINOUT_VOLTAGE
Description : Voltage difference between SPA driver supply and receiver supply for
SPA constraint on [ConstraintNode]
Violation : LP:2
ConstraintNode : inout1[2]
DriverInfo
  PowerNet
    NetName : VDD_SPA_DRIVER
    NetType : UPF
  PowerMethod : FROM_UPF_DRIVER_SUPPLY
  GroundNet
    NetName : VSS
    NetType : UPF
  GroundMethod : FROM_UPF_DRIVER_SUPPLY
ReceiverInfo
  PowerNet
    NetName : VDD
    NetType : UPF
  PowerMethod : FROM_UPF_POWER_DOMAIN
  GroundNet
    NetName : VSS
    NetType : UPF
  GroundMethod : FROM_UPF_POWER_DOMAIN
UPFCommand
  Command : set_port_attributes
  FileName : spa_driver_on_inout_vector_port.upf
  LineNumber : 35

```

### 5.3.39 Support for `set_port_attributes -attribute antenna_diode_*`

VC LP supports the following syntax of `set_port_attributes -attribute antenna_diode_*`:

```

set_port_attributes -model { . } -ports {IN1_A} -attribute
antenna_diode_related_power_pins {VDD1 VDD2}
set_port_attributes -model { . } -ports {IN1_A} -attribute
antenna_diode_related_ground_pins {VDD1 VDD2}

```

When the `antenna_diode_related_power_pins/ground_pins` attributes or `is_analog` is set on the top scope ports through `set_port_attributes`, these settings are considered for UPF\_SPA\* and ANALOG\_NET\_\* violations for the ports which are shorted with them.

#### Example

```

assign in2 = in1 ;
set_port_attributes -ports
{ in2 }
-is_analog -model

```

VC LP reports the UPF\_SPAANALOG\_INCORRECT violation for `in2` and UPF\_SPAANALOG\_MISSING for `in1`, since it considers the `is_analog` attribute written for the shorted port `in2` as `in1` is an analog load.

The following violations are enhanced to report additional fields:

- ❖ UPF\_SPA\_NODIODE

*CoveringPortSupplies:* The set of 'port diode supplies which cover this design diode supply even though it isn't listed as an explicit port attribute

*PortAncestorSupplies:* The set of SPA ADRPP supplies which the design diode supply mentioned in this violation is 'derived' from. (A derived supply is a switched or renamed version of the 'ancestor' supply)

- ❖ UPF\_SPA\_DIODE

*CoveredDesignDiodeSupplies:* The field contains a list design diode supplies which are equal/more on than (but not same name as) this top port diode supply.

The field when not-empty indicates that the top-level SPA attribute is using conservative modeling for some supplies inside the design and thus we may want to keep the top-level SPA attribute anyway to cover diode inside the design which are on other supplies.

*CoveringPortSupplies:* lists out if other top-level SPA diode supplies that effectively cover this one from \*\_DIODE\_INSUFFICIENT violations.

### 5.3.40 Support for set\_retention\_elements UPF Command

VC LP supports `set_retention_elements` UPF command. Using this command, you can create a list of critical registers. The list created using `set_retention_elements` can be used directly in the retention strategy.

VC LP performs predefined checks on the UPF strategy based on predefined rules (as described in section [5.3.40.1](#)), and if any of the rules is violated, then parsing error messages are reported.

#### Syntax

```
set_retention_elements <list_name> -elements <retention_element_list>
```

- ❖ `list_name` must be a simple (non-hierarchical) name.
- ❖ `-elements` can be instances, sequential registers, or signals.

#### 5.3.40.1 Parser Error Messages Introduced for this Support

The elements of a `retention_element_list` should abide by the following rules:

1. It shall be an error if an element belonging to `retention_element_list` is not retained when,
  - a. any other element in the same `retention_element_list` is retained (RET\_ELEM\_MIXED).  
(OR)
  - b. any other element in the same power domain extent is retained  
(RET\_ELEM\_DOMAIN\_EXTENT).
2. It shall be an error when a `-no_retention` strategy is applied to any element which belongs to a `retention_element_list` (RET\_ELEM\_NORETAIN).

#### Note

This does not mean all the elements are retained by just specifying them in the `retention_element_list` of `set_retention_elements` command. You must write the retention strategy in order to retain it.

### 5.3.40.1.1 The RET\_ELEM\_MIXED Error

Figure 5-103 depicts an example scenario where rule 1 (a) (as described in section 5.3.40.1) is satisfied, and hence no error message is reported.

**Figure 5-103 Example when RET\_ELEM\_MIXED is not Reported**

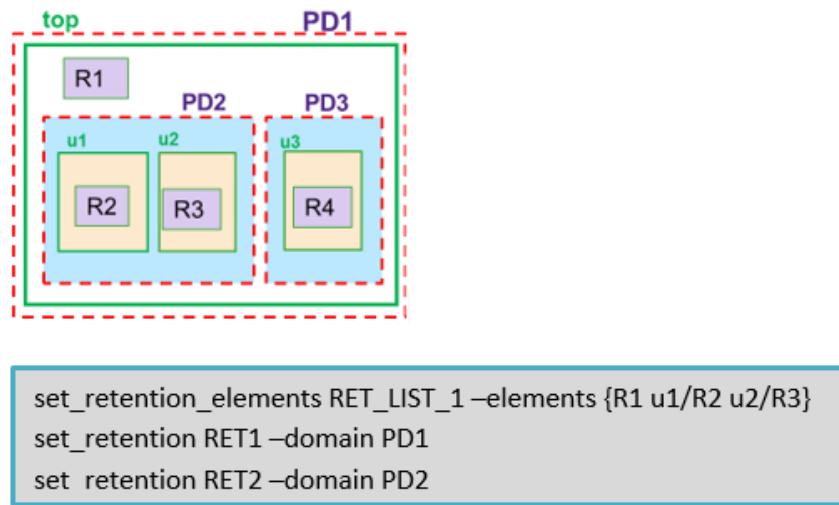


Figure 5-104 depicts an example scenario where rule 1 (a) is violated, and hence the following error message is reported:

[Error] RET\_ELEM\_MIXED: Retetion element list has both retained and non retained elements  
*Retention element list RET\_LIST\_1 has both retained and non retained elements, please refer to dump file for details:./vcst\_rtdb/lpdb/debug\_reports/RET\_ELEM\_MIX.rpt.*

The RET\_ELEM\_MIXED violation dumps details of retained and non-retained elements into a separate file for users to check.

The details in the dump file: set\_retention\_elements list name, all the elements in the list, retained elements with retention strategy info and non-retained elements:

```

=====
set_retention_elements - RET_LIST_1
    Elements - ret_i1, CORE1/ret_i3, CORE2/ret_i5
    Retained Elements - ret_i1(TOP/RET1), CORE1/ret_i3(PD1/RET2)
    Non-Retained Elements - CORE2/ret_i5
=====
```

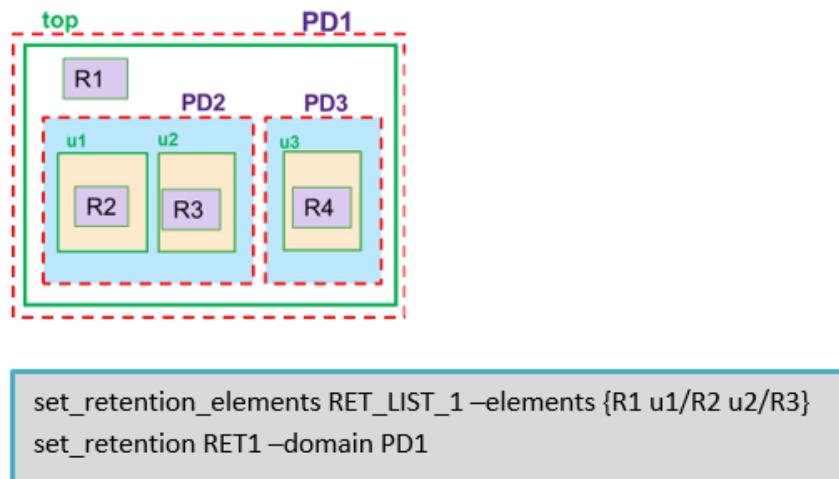
**Figure 5-104 Example when RET\_ELEM\_MIXED is Reported****5.3.40.1.2 The RET\_ELEM\_DOMAIN\_EXTENT Error Message**

Figure 5-105 depicts an example scenario where rule 1 (b) (as described in section 5.3.40.1) is satisfied, and hence no error message is reported.

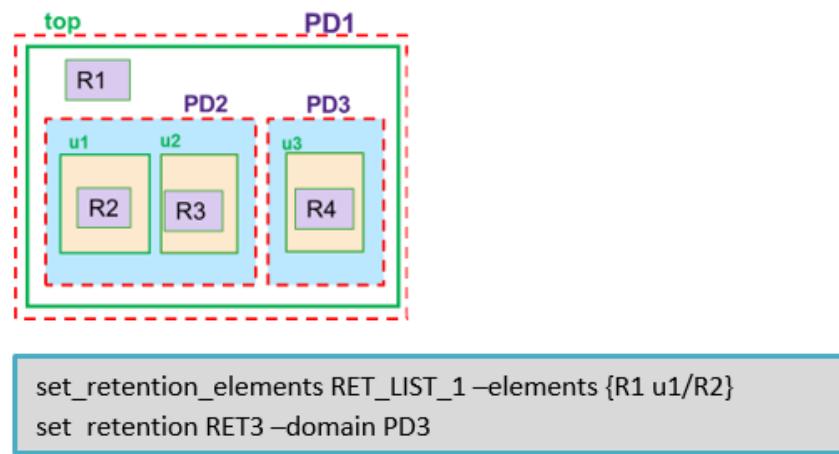
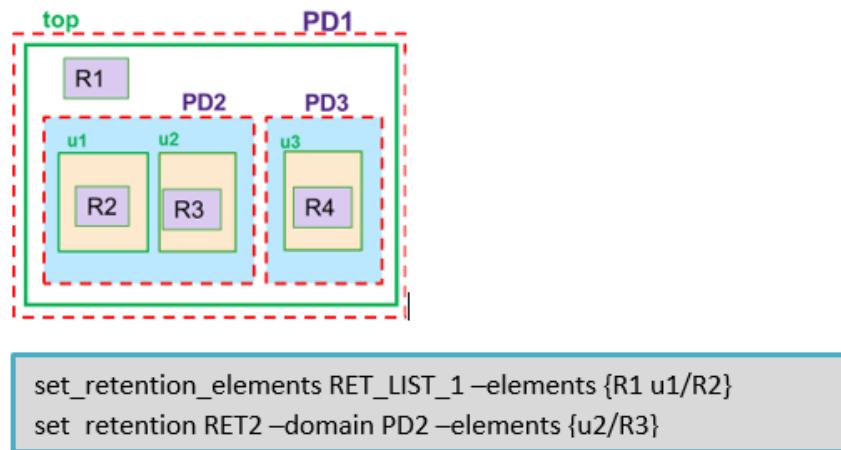
**Figure 5-105 Example when RET\_ELEM\_DOMAIN\_EXTENT is not Reported**

Figure 5-106 depicts an example scenario where rule 1 (b) is violated, and hence the following error message is reported:

*[Error] RET\_ELEM\_DOMAIN\_EXTENT: Elements retained in domain extent*

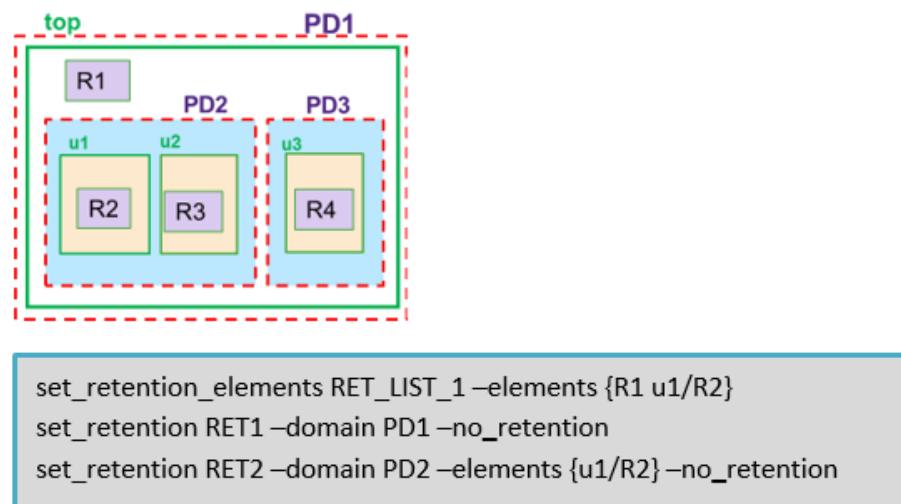
*Object u1/R2 of RET\_LIST\_1 is not retained whereas some objects of domain PD2 are retained*

**Figure 5-106 Example when RET\_ELEM\_DOMAIN\_EXTENT is Reported**

#### 5.3.40.1.3 The RET\_ELEM\_NORETAIN Error

Figure 5-107 depicts an example scenario where rule 2 (as described in section 5.3.40.1) is violated, and hence the following error message is reported

[Error] RET\_ELEM\_NORETAIN: -no\_retention applied  
R1 belongs to RET\_LIST\_1. no\_retention cannot be applied to it  
[Error] RET\_ELEM\_NORETAIN: -no\_retention applied  
u1/R2 belongs to RET\_LIST\_1. no\_retention cannot be applied to it

**Figure 5-107 Example when RET\_ELEM\_NORETAIN is Reported**

### 5.3.41 Support for set\_retention -use\_retention\_as\_primary UPF Command

VC LP supports the `set_retention -use_retention_as_primary` UPF command. Users may need the retention cell outputs they have in their design to be powered with either retention strategy supply or the domain supply. Using the `set_retention -use_retention_as_primary` command, you can specify which one of these to use as the target of a retention strategy. When this command is specified, the retention supply is used; else the domain primary supply is used.

#### Syntax

```
set_retention retention_name
  -domain domain_name
  [-use_retention_as_primary]
```

Where:

- ❖ `-use_retention_as_primary`: The `-use_retention_as_primary` option specifies that the storage element and its output are powered by the retention supply.

### 5.3.42 Support for exclude\_elements with set\_retention Command

The `-exclude_elements` option is supported in the `set_retention` command. The `-exclude_elements` option specifies a list of objects that should not be included in this strategy.

The objects in the exclude list can be retention sequential cell or a pin, port, net driven by sequential operator or retention sequential cell. The element specified in the `-exclude_elements` could also be a hierarchy cell, VC LP searches for valid design objects under the cell for retention association.

The elements list specified in `set_retention_elements` also can be used in `-exclude_elements`.

If the element in `-exclude_elements` list is not in the power domain of the strategy, parser message `UPF_RETENTION_ELEMENT_NOT_IN_EXTENT` is reported.

#### Note

- ❖ For invalid design objects in retention `-exclude_elements`, VC LP does not report any messages for them. This will be introduced in future.
- ❖ For UPF\_EQIVTRETP, the check with `-exclude_elements` is the same as for `-elements`. It does simple UPF command parse check and is not based on effective elements.

The following commands are introduced for this support

- ❖ `get_resolved_elements`  
This debug command is introduced to output the effective elements in the retention strategy after excluding the elements in the exclude element list.
- ❖ `get_retention_strategy_elements`  
This debug command is updated to include information on exclude element list.

### 5.3.43 Support for -supply\_set in create\_power\_switch UPF Command

The `-supply_set` option is supported it the `create_power_switch` UPF Command. When the `-supply_set` is used in the corresponding `create_power_switch` command, the specified supply set is assumed to be the related supply of the control signals and ack ports/nets of the switch.

#### Syntax

```
create_power_switch switch_name
  [-domain domain_name]
```

```
-output_supply_port {port_name [supply_net_name] }  
[-supply_set supply_set_ref]
```

### 5.3.43.1 Checks Added for This Support

#### ❖ PSW\_SUPPLY\_SCOPE

This violation is reported when the indicated power switch strategy is defined in the scope, but the supply\_set defined for the strategy is not defined in the same scope.

**Figure 5-108 Example for PSW\_SUPPLY\_SCOPE Violation**

```
load_upf ./const1.upf -scope inst1  
  
inside const1.upf --> create_supply_set ss_ctrl -function {power vdd1} -function  
{ground vss}  
.....  
create_power_switch PSW1 -domain TOP -supply_set inst1/ss_ctrl  
.....
```

PSW\_SUPPLY\_SCOPE (1 warning/0 waived)

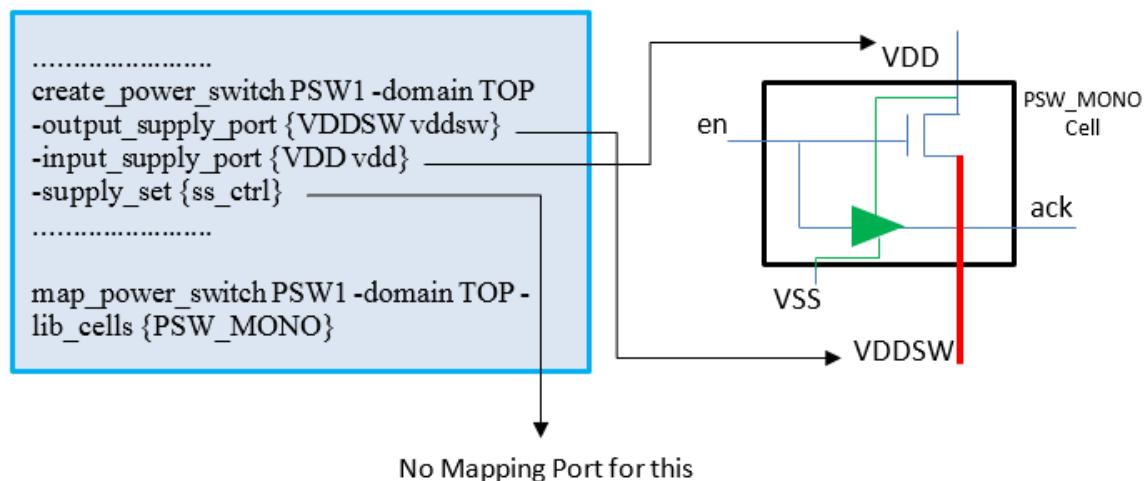
```
Tag      : PSW_SUPPLY_SCOPE  
Description : Supply set [SupplySet] is not defined in the same scope where power switch strategy [Strategy] is  
defined  
Violation : LP:5  
SupplySet : ss_ctrl  
Strategy  : PSW1  
SupplyScope : top/inst1  
StrategyScope : top
```

### 5.3.43.2 Checks Enhanced for This Support

- ❖ PSW\_MAP\_MISMATCH

The debug field `pg_pin_count` is added to PSW\_MAP\_MISMATCH violation. This debug field indicates that the `map_power_switch` does not have the cell that is needed for the `create_power_switch` (for example, `create_power_switch` uses three power supplies but `map_power_switch` can support only two power supplies).

**Figure 5-109 Example for PSW\_MAP\_MISMATCH Violation**




---

PSW\_MAP\_MISMATCH (1 error/0 waived)

---

```

Tag      : PSW_MAP_MISMATCH
Description : Power switch map cells [MappedCells] attribute do not match strategy [Strategy]
Violation   : LP:5
MappedCells
  MappedCell  : PSW_MONO
Strategy     : PSW1
Attribute    : pg_pin_count

```

❖ PG\_STRATEGY\_CONN

Currently, the PG\_STRATEGY\_CONN violation is reported for isolation and retention strategies when there is a mismatch in the supply specified in the strategy and the supply connected to primary/backup pg pins in design.

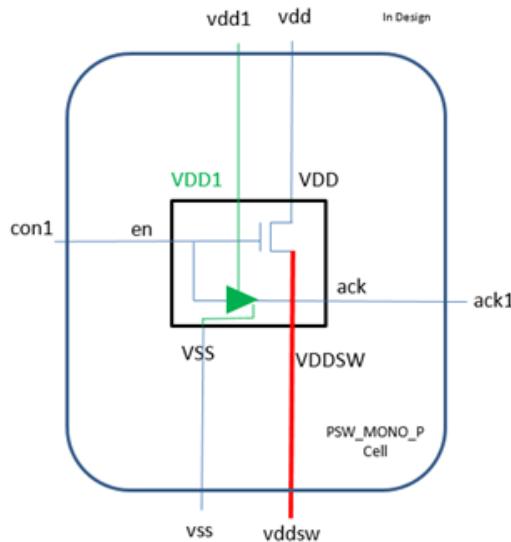
The PG\_STRATEGY\_CONN is also reported when the supply referred in `-supply_set` option of `create_power_switch` command does not match with the supply connected to pg pins in the design.

**Figure 5-110 Example for PG\_STRATEGY\_CONN Violation**

```
create_supply_set ss_ctrl -function {power vdd} -function {ground vss}
create_power_switch PSW1 -domain TOP -supply_set ss_ctrl
```

In UPF

```
PSW_MONO_P psw1 (.en(con1),.ack(ack1),.VDD(vdd),VDD1(vdd1),.VDDSW(vddsw),.VSS(vss));
```



PG\_STRATEGY\_CONN (1 error/0 waived)

```

Tag      : PG_STRATEGY_CONN
Description : UPF strategy supply net [UPFSupply] does not match design supply net [DesignNet] on pin [CellPin] of
CellType instance [Instance] ([Cell])
Violation   : LP:16
UPFSupply   : vdd
DesignNet
  NetName    : vdd1
  NetType     : Design/UPF
  CellPin    : VDD1
  CellType    : PowerSwitch
  Instance    : psw1
  Cell        : PSW_MONO_P
  Strategy    : PSW1
  PinPGType   : PrimaryPower

```

### 5.3.44 Support for add\_power\_state

VC LP supports a larger subset of the `add_power_state` command in UPF. You can use this when you have large, repetitive power state tables, usually as a top level constraining table. You can use the `add_power_state` command at the top level to create a much more compact and readable top level constraint. This also enables you to create new RTL designs which avoids the use of `create_pst` altogether as per the UPF IEEE committee recommendation.

The following sections provide the syntax of the `add_power_state` that is supported, and the changes introduced in VC LP as part of this support.

Prior to L-2016.06, the limited form of the `add_power_state` command was supported, which is exactly equivalent to one port state. The following two separate statements had the same effect.

```
add_port_state V1 -state {P1 1.2}
```

```
add_power_state SS1 -state P1 {-supply_expr {(power == `{FULL_ON, 1.2})}}
```

The first command creates a portstate P1 for the supply net V1, and the second command creates a portstate P1 for the supply set function SS1.power. For VC LP, the two commands behave the same way.

The following enhancements are done to the `add_power_state` command:

- ❖ Support for the `-supply` option in the `add_power_state` command
- ❖ Support for UNDETERMINED in the supply expression in the `add_power_state` command. This string is treated as "off".
- ❖ Support for `add_power_state -domain` without `-logic_expression`
- ❖ Support for multiple terms in `supply_expr`
- ❖ Support for group syntax and for creating PST using `add_power_state -group`.

VC LP automatically detects the version (UPF 2.0 or UPF 2.1) of the `add_power_state` command and supports the syntax. You can also use both UPF 2.0 and UPF 2.1 format of the `add_power_state` command. The following is an example of the UPF2.1 syntax for `add_power_state` command.

#### Examples

- ❖ `add_power_state -supply SS_PD1 -state { VDD1_ON -supply_expr {power == '{FULL_ON, 1.0} } }`
- ❖ `add_power_state -state { VDD1_ON2 -supply_expr {power == '{FULL_ON, 0.8} } }`  
`SS_PD1`
- ❖ `add_power_state SS_PD2 -state { VDD1_ON -supply_expr {power == '{FULL_ON, 1.0} } }`

#### Example 1

```
add_power_state SS1 -state {ON -supply_expr {power =={FULL_ON, 1.1}}}
add_power_state SS2 -state ON {-supply_expr {power =={FULL_ON, 1.0}}}
create_power_state_group Group
add_power_state -group Group -state S1 {-logic_expr {(SS1 == ON1) && (SS2 == ON)}}
```

#### Example 2

```
add_power_state SS1 -state S1 {-supply_expr {power == {OFF}}}
add_power_state SS1 -state {S2 -supply_expr {ground == {OFF}}}
```

Result:

*name.upf:<line\_number>: [Error] UPF\_STATE\_SYNTAX\_NOT\_ALLOWED: add\_power\_state -state syntax not allowed*

*Power state for 'SS1' was previously defined using UPF '2.0' syntax for -state option. UPF style cannot be changed for -state option for add\_power\_state of 'SS1'.*

*Please change '-state' syntax.*

### Limitation

- ❖ For a given supply set, only one of UPF 2.0 or UPF 2.1 is supported.

## The enable\_top\_only\_pst Application Variable

When the `enable_top_only_pst` application variable is enabled, the non-top scope system states are ignored and the top scope system states are considered. The `enable_top_only_pst` application variable is supported for UPF 3.0 constructs of `add_power_state`.

### Example

#### *Top.upf*

```
add_power_state -supply SS_top -state {ON -supply_expr {(power == {FULL_ON 1.4}) && (ground == {FULL_ON 0})}}
add_power_state -supply SS_top -state {OFF -supply_expr {(power == OFF) && (ground == {FULL_ON 0})} }

load_upf SUB.upf -scope uSUB1

connect_supply_net VDD -ports {uSUB1/VDD1}
connect_supply_net VSS -ports {uSUB1/VSS}

create_power_state_group PSG_top
add_power_state -group PSG_top -state {ALL_ON -logic_expr {(SS_top == ON)}}
add_power_state -group PSG_top -state {SWITCH_OFF -logic_expr {(SS_top == OFF)}} - update
```

#### *SUB.upf*

```
add_power_state -supply SS_sub -state {ON -supply_expr {(power == {FULL_ON 1.5}) && (ground == {FULL_ON 0})}}
add_power_state -supply SS_sub -state {OFF -supply_expr {(power == OFF) && (ground == {FULL_ON 0})} }

create_power_state_group PSG_sub
add_power_state -group PSG_sub -state {ALL_ON -logic_expr {(SS_sub == ON)}}
add_power_state -group PSG_sub -state {SWITCH_OFF -logic_expr {(SS_sub == OFF)}} - update
```

```
System pst default
VDD
OFF
```

```
System pst with enabling app var
VDD
```

OFF  
1.4

A warning message is reported when a child scope power group state, pst state, or supply set state is referred from the top level when `enable_top_only_pst` is true

For example

`power_group_upf/INST.upf:12: [Warning] POWER_STATE_IGNORED: Supply set state 'OFF' for supply set 'inst1/SS_sub' is ignored since it is not created in the top scope.`

*Please create this state in top level or set application variable 'enable\_top\_only\_pst' to false.*

`power_group_upf/INST.upf:18: [Warning] INVALID_GROUPSTATE_REFERENCED: Group state 'inst1/PSG_sub/first' for Group 'inst1/PSG_sub' is invalid.`

*Please create this group state in top level or set application variable 'enable\_top\_only\_pst' to false.*

`INST.upf:15: [Warning] PST_STATE_IGNORED: State 'first' for PST 'inst1/pst1' is ignored since it is not created in the top scope.`

*Please create this state in top level or set application variable 'enable\_top\_only\_pst' to false.*

#### 5.3.44.1 Support for Scientific Expression in the add\_power\_state Command

The scientific notation (also referred to as standard form or standard index form) is a way of expressing numbers that are too big or too small in a decimal form. The scientific format displays a number in exponential notation, replacing part of the number with E+ n, where E (which stands for Exponent) multiplies the preceding number by 10 to the nth power.

VC LP supports the scientific notation and converts voltage into a double value.

This converted value is used in the subsequent LP checks.

**Example**

```
add_power_state SS1 -state VDD_12 {-supply_expr {power == `{FULL_ON, 0.12e+1 }`}}
```

You can use either 'E' or 'e' for representing the exponent.

For example, voltage value 1.2 can be represented in various forms: 1.2e+0 or 0.12e+1 1.2E+0 or 0.12E+1.

#### 5.3.44.2 Support for Specifying State Name Outside the Braces

By default, VC LP supports the state name of the `add_power_state` inside the braces (as supported in UPF 2.1).

**Example**

```
add_power_state SS2 -state {VDD2_12 -supply_expr {power == `{FULL_ON, 1.2}`}}
add_power_state SS2 -state {VDD2_OFF -supply_expr {power == `{OFF}`}}
```

#### 5.3.44.3 Support of -supply in the add\_power\_state Command

The `-supply` option is added and made default in `add_power_state` command. This support is backward compatible, that is, you can write `add_power_state` without `-supply` option as well. Therefore, the following two commands are equivalent.

```
add_power_state SS1 -state P1 {-supply_expr {(power == `{FULL_ON, 1.2}`)}}
```

```
add_power_state -supply SS1 -state P1 {-supply_expr {(power == `{FULL_ON, 1.2}`)}}
```

#### 5.3.44.4 Support for Multiple Terms in supply\_expr

You can use complex supply expression, with multiple functions of the same supply connected with AND in the add\_power\_state command.

##### Example

```
add_power_state -supply SS2 -state P2 \
    {-supply_expr { (power == `{FULL_ON, 1.2}) && \
        (ground == `{FULL_ON, 0.0})}}
```

```
add_power_state -supply SS2 -state P3 \
    {-supply_expr { (power == `{FULL_ON, 0.8}) && \
        (ground == `{FULL_ON, 0.0})}}
```

```
add_power_state -supply SS2 -state P4 \
    {-supply_expr { (power == `{FULL_ON, 0.8}) && \
        (nwell == `{FULL_ON, 1.2})}}
```

All of the terms in the supply expression refer to functions of the same supply set. Different expressions may refer to different functions; for example, P4 refers to the nwell function which is not defined for P2 or P3. The missing nwell function in P2 and P3 can be taken as equivalent to the "\*" or don't-care character in a power state table.

#### 5.3.44.5 VC LP Parser Behavior for the add\_power\_state Command

UPF 2.0 has syntax with the tick and comma characters. The tick and comma are required in UPF 2.0:

```
add_power_state SS2 -state P3 \
    {-supply_expr { (power == `{FULL_ON, 0.8}) && \
        (ground == `{FULL_ON, 0.0})}}
```

The tick and comma are not required:

```
add_power_state SS2 -state P3 \
    {-supply_expr { (power == {FULL_ON 0.8}) && \
        (ground == {FULL_ON 0.0})}}
```

The VC LP parser is flexible and accepts either combination of syntax, even within the same line.

#### 5.3.44.6 Support for Group Syntax

Prior to L-2016.06, creating a PST using the create\_pst command was a must. You can create a PST from the concept of a group. The create\_power\_state\_group UPF command is used to create a named group. It takes only single argument, that is the name of the group. The following sections shows how PST can be created using groups.

##### 5.3.44.6.1 Supply Terms in logic\_expr

You can use the logic expression in the add\_port\_state command. Each logic expression can contain several terms. The terms can be connected with the AND operator. Each term may be one of several supply set state names.

##### Example

```
create-power_state_group G2
add_power_state -group G2 -state S5 {-logic_expr { (SS1 == P1) && (SS2 == P3) }}
add_power_state -group G2 -state S6 {-logic_expr { (SS1 == P1) && (SS2 == P4) }} -update
```

Each command of this type can be understood as one line of a power state table, where the columns are built as needed from the supply set members.

VC LP also supports UPFs with more complex expressions including the OR (||) operator. There can be four scenarios where the OR operator may appear in an expression. This represents a cross product of (power states versus logic signals) appearing in (supply versus domain/group).

- ❖ OR operator in power states in `add_power_state -supply -supply_expr`
- ❖ OR operator in logic signals in `add_power_state -supply -logic_expr`
- ❖ OR operator in power states in `add_power_state -domain/group -logic_expr`

### Note

The OR operator in logic signals in `add_power_state -domain/group -logic_expr` command is currently ignored.

### Examples

```
add_power_state TOP.primary -state HV {supply_expr {power == {FULL_ON 1.2} || {ground == {FULL_ON 0.0}}}}
add_power_state TOP.primary -state LV {-supply_expr {power == {FULL_ON 1.0} && {ground == {FULL_ON 0.0}}}}
add_power_state TOP.primary -state OFF {-supply_expr {power == {OFF} && {ground == {FULL_ON 0.0}}}}
create_power_state_group TOP_G
add_power_state TOP_G -state ALL_HV {-logic_expr {TOP.primary == HV && PD1.primary == HV && PD2.primary == HV}}
```

### Example of the PST Generated

```
add_power_state TOP.primary -state STANDBY { -supply_expr {(power == {FULL_ON 1.0}) || (nwell == {FULL_ON -0.6}) && (ground == {FULL_ON 0})}}
add_power_state TOP.primary -state NORMAL {-supply_expr {(power == {FULL_ON 0.8}) && (ground == {FULL_ON 0}) && (nwell == {FULL_ON 0})}}
add_power_state TOP.primary -state TURBO {-supply_expr {(power == {FULL_ON 1.1}) && (ground == {FULL_ON 0}) && (nwell == {FULL_ON 0})}}
add_power_state TOP.primary -state SLEEP {-supply_expr {(power == OFF) || (ground == OFF)}}
add_power_state PD1.primary -state PD1_NORMAL {-supply_expr {(power == {FULL_ON 1.2}) && (ground == {FULL_ON 0})}}
create_power_state_group TOP_G
add_power_state TOP_G -state ALL_HV {-logic_expr {TOP.primary == STANDBY && PD1.primary == PD1_NORMAL }}
```

The PST for TOP.primary should be similar to the following figure. Here the PST entries are replaced with voltage values.



PST State	Sub PST State	power	ground	nwell
STANDBY	STANDBY_state1	1.0	0	*
	STANDBY_state2	*	0	-0.6
NORMAL	NORMAL	0.8	0	0
TURBO	TURBO	1.1	0	0
SLEEP	SLEEP_state1	OFF	*	*
	SLEEP_state2	*	OFF	*

report_system_pst				
TOP.primary.power	TOP.primary.ground	TOP.primary.nwell	PD1.primary.power	PD1.primary.ground
1.0	0	-0.6	1.2	0
1.0	0	0	1.2	0
0.8	0	-0.6	1.2	0
1.1	0	-0.6	1.2	0
OFF	0	-0.6	1.2	0

### 5.3.44.6.2 Group/PST States in -logic\_expr

The advantage of the add\_power\_state command is that the logic expression for a state can refer to other states. Any add\_power\_state logic expression can refer to either a supply set state, or another group state name.

#### Example

```
create_power_state_group G3
add_power_state -group G3 -state S9 \
    {-logic_expr { (SS1 == P1) && (SS2 == P2) } }

create_power_state_group G4
add_power_state -group G4 -state S2 \
    {-logic_expr { (G3 == S9) && (SS3 == P3) } }

create_power_state_group G5
add_power_state -group G5 -state S1 \
    {-logic_expr { (pst1 == s1) && (G1 == s1)&& (SS4 == P4) } }
```

Note the term G3 == S9. This means that group G4 is in power state S2 when group G3 is in power state S9, plus some other conditions. This expression enables you to make top level constraining power states. Normally, a constraining top level power state table must repeat all the values of the individual supply nets. However, now the top level power state group can refer to the state names of the lower level tables instead, which is more compact and readable.

### 5.3.44.7 Adding Voltage Later Using -update Option

VC LP supports -update flag in the add\_power\_state command.

IP providers would like to distribute UPF which does not refer to voltage values. Then, you can add new UPF code with -update that fills in the voltage values. For example, the IP provider would give this type of add\_power\_state; notice there are no voltage values.

```
add_power_state -supply ss1 -state st1 \
    {-supply_expr {power == FULL_ON}}
```

```
add_power_state -supply ss2 -state st2 \
    {-supply_expr {power == FULL_ON}}
```

Then, you can add new UPF code to fill in the voltage values without editing the existing lines:

```
add_power_state -update -supply ss1 -state st1 \
    {-supply_expr {power == {FULL_ON 1.2}}}
```

```
add_power_state -update -supply ss2 -state st2 \
    {-supply_expr {power == {FULL_ON 1.2}}}
```

VC LP issues an error after processing all the UPF, if any state remains with no voltage.

#### 5.3.44.8 Support for -domain Option in add\_power\_state

Previously, VC LP supported the `add_power_state -group` option to create complex PSTs. You had to use the `create_power_state_group` command to create a group.

With UPF 3.0, the `add_power_state -domain` option adds the power state to the specified domain. To make VC LP compatible with UPF 3.0, VC LP supports the `-domain` option in `add_power_state`.

##### Syntax

```
create_power_domain TOP_Domain
add_power_state -domain TOP_Domain \
    -state HV {-logic_expr {TOP.primary == OFF} && {TOP.default.isolation == ON}}
```

The PST state is added to the specified domain.

#### 5.3.44.9 Defining Supply Set Name With Shorthand

There are different methods of defining the same power state. The following two methods are exactly same.

- ❖ The following is the default method to define power state:

```
add_power_state -domain TOP_Domain ... {-logic_expr {TOP.primary == OFF}}
```

- ❖ The following is shorthand method to defined power state.

```
add_power_state -domain TOP_Domain ... {-logic_expr {primary == OFF}}
```

The shorthand method takes the default supply set of that domain. However, using shorthand syntax can sometimes create ambiguity.

Consider the following example:

```
create_supply_set primary ...
add_power_state -domain TOP_Domain ... -logic_expr {primary==s11}
```

In the second line, there is an ambiguity of whether the reference is to the standalone supply set with the name "primary", or to the shorthand reference to the primary supply set of `TOP_Domain`. To avoid this ambiguity, use the dot slash notation.

```
add_power_state -domain TOP_Domain... -logic_expr {./primary==s11}
```

Therefore, the example without `'.'` refers primary supply set of `TOP_Domain`, and the example with the `./` notation refers standalone supply set.

### 5.3.44.10 Support for Predefined Power States on Supply Set Objects

VC LP supports predefined power states on supply set objects. There are two predefined power states *ON* and *OFF* for each supply set. VC LP allows the usage of these *ON* and *OFF* states before definition.

Consider the following UPF:

```
create_power_domain PDSYS -elements { . } -supply {primary} -supply {aon}
create_power_domain PDCPU -elements {v_cpu}-supply {primary} -supply {aon}
add_power_state -domain PDCPU \
    -state {ON -logic_expr {PDCPU.aon == ON && PDCPU.primary == ON}} \
    -state {OFF -logic_expr {PDCPU.aon == ON && PDCPU.primary == OFF}}
```

The above `add_power_state` command does not error out and be accepted and will use the predefined *ON* and *OFF* states defined for the `add_power_state` command in the group.

#### Example

In the following example, "ON" state for supply sets (TOP.primary and TOP.aon) are referred in the group before definition.

```
create_power_domain TOP -supply {primary} -supply {aon}
create_power_domain PD1 -elements {CORE1} -supply {primary} -supply {aon}

add_power_state -domain TOP -state { TOP_ON -logic_expr { {TOP.primary == ON} && {TOP.aon == ON} } }
add_power_state -domain PD1 -state { PD1_ON -logic_expr { {PD1.primary == ON} && {PD1.aon == ON} } }

create_power_state_group Group
add_power_state -group Group -state S1 {-logic_expr {(TOP == TOP_ON) && (PD1 == PD1_ON)}}

add_power_state TOP.primary -state { ON -supply_expr {power=={FULL_ON 1.0}} && ground == {FULL_ON 0.0}} -update
add_power_state TOP.aon -state { ON -supply_expr {power=={FULL_ON 1.0}} && ground == {FULL_ON 0.0}} -update
add_power_state PD1.primary -state { ON -supply_expr {power=={FULL_ON 1.2}} && ground == {FULL_ON 0.0}} -update
add_power_state PD1.aon -state { ON -supply_expr {power=={FULL_ON 1.2}} && ground == {FULL_ON 0.0}} -update
```

Previously, VC LP reported following parser error since as predefined states was not supported:

*top.upf:4: [Error] UPF\_STATE\_NOT\_DEFINED\_ERROR: State not defined on objectState 'ON' not defined on object 'TOP.primary' specified in 'add\_power\_state -logic\_expr'. Please add the state on the specified object or provide a correct name of state.*

Starting with this release, the above example is a valid UPF and no error message is reported.

#### Notes

- ❖ VC LP requires the predefined *ON* and *OFF* states to be fully defined before any action commands are performed.
- ❖ VC LP issues an error at `read_upf` stage when predefined *ON* and *OFF* states are not fully defined using supply expressions in the UPF.
- ❖ If you redefine these predefined states after referring (as in the above example), you should use `-update` option with the `add_power_state` command. Else, an error message is reported.

- ❖ If you redefine these predefined states before referring, the `-update` option in the `add_power_state` command is optional. VC LP supports both ways, with `-update` or without `-update`.
- ❖ These predefined state names *ON* and *OFF* are case sensitive.

### 5.3.44.11 Switches `-domain`, `-group` and `-supply` in `add_power_state` is made optional

Previously, the `add_power_state` command has one mandatory field, which is the object name. VC LP makes all three switches (`-supply`, `-group` and `-domain`) optional.

If the `-group` option is given, it refers to the existing group name. If the `-supply` option is given, it refers to the existing supply set name.

If none of the switches are given, then search is made on all the three types of object and the object that is found first is used.

```
create_supply_set SS1
add_power_state -supply SS1 -state SS1_ON \
    {-supply_expr {power == {FULL_ON 1.2} && {ground == {FULL_ON 0.0}}}}
add_power_state SS1 -state HV \
    {-logic_expr {SS1 == SS1_ON}}
```

```
create_power_domain PD1
add_power_state -domain PD1 \
    -state HV \
    {-logic_expr {PD1.primary == ON && default.retention == ON}}
```

```
create_power_state_group TOP_G
add_power_state TOP_G -group -state ALL_ON \
    {-logic_expr {top_ss == ON && core1_ss == ON}}
```



#### Note

You cannot create any two supply sets, domains or groups with the same name. For example, if a supply set named X exists, and the `add_power_state -domain X` is executed, the command fails because there is no domain with that name.

### 5.3.44.12 Example

This section contains three complete, small implementations using the same reference design. The first implementation shows the old way using power state tables. The second implementation is compliant with the 3.0 LRM, including all the supply set definitions.

#### 5.3.44.12.1 Reference design

Here is a high level, syntax free description of a system with two blocks. Each block has three supplies and three valid block power states.

**Figure 5-111 Reference Design for the Example****PST for PD\_A:**

State	VA	VA1	VA2
RUN	ON	ON	ON
IDLE	ON	OFF	ON
DOZE	ON	OFF	OFF

**PST for PD\_B:**

State	VB	VB1	VB2
RUN	ON	ON	ON
IDLE	ON	ON	OFF
DOZE	ON	OFF	OFF

The top level constraining power state information gives the valid system states:

- ❖ If PD\_A is in RUN, PD\_B can only be in RUN or IDLE
- ❖ If PD\_A is in IDLE, PD\_B can only be in IDLE
- ❖ If PD\_A is in DOZE, PD\_B can only be in DOZE

#### 5.3.44.12.2 Current Style with Power State Tables

Here are the power state tables for the two blocks.

```

add_port_state VA -state {ON 1.2} {OFF off}
add_port_state VA1 -state {ON 1.2} {OFF off}
add_port_state VA2 -state {ON 1.2} {OFF off}
add_port_state VB -state {ON 1.2} {OFF off}
add_port_state VB1 -state {ON 1.2} {OFF off}
add_port_state VB2 -state {ON 1.2} {OFF off}

create_pst_table TA      -supplies {VA  VA1  VA2}
add_pst_state run -pst TA -states {ON  ON  ON}
add_pst_state idle -pst TA -states {ON  OFF  ON}
add_pst_state doze -pst TA -states {ON  OFF  OFF}

create_pst_table TB      -supplies {VB  VB1  VB2}
add_pst_state run -pst TB -states {ON  ON  ON}
add_pst_state idle -pst TB -states {ON  ON  OFF}
add_pst_state doze -pst TB -states {ON  OFF  OFF}

```

To create the top level constraining power state table, you must repeat all of the low level supply combinations. For larger designs, creating it can be tedious and error-prone, and validating it can be very difficult.

```

create_pst_table TTOP      -supplies {VA  VA1  VA2  VB  VB1  VB2}
add_pst_state run_run -pst TTOP -states {ON  ON  ON  ON  ON  ON}
add_pst_state run_run -pst TTOP -states {ON  ON  ON  ON  OFF  ON}
add_pst_state idle_idle -pst TTOP -states {ON  OFF  ON  ON  ON  OFF}
add_pst_state doze_doze -pst TTOP -states {ON  OFF  OFF  ON  OFF  OFF}

```

#### 5.3.44.12.3 Supply set method

First, you must define the valid supply set combinations; then you define the lower level groups; and finally define the top level constraining group.

```
# Supply set states
```

```

add_power_state -supply SA \
    -state ON {-supply_expr {(power == {FULL_ON 1.2}) && \
                           (ground == {FULL_ON 0.0})}} \
    -state OFF {-supply_expr {(power == OFF) && (ground == OFF)}}

add_power_state -supply SA1 \
    -state ON {-supply_expr {(power == {FULL_ON 1.2}) && \
                           (ground == {FULL_ON 0.0})}} \
    -state OFF {-supply_expr {(power == OFF) && (ground == OFF)}}

add_power_state -supply SA2 \
    -state ON {-supply_expr {(power == {FULL_ON 1.2}) && \
                           (ground == {FULL_ON 0.0})}} \
    -state OFF {-supply_expr {(power == OFF) && (ground == OFF)}}

add_power_state -supply SB \
    -state ON {-supply_expr {(power == {FULL_ON 1.2}) && \
                           (ground == {FULL_ON 0.0})}} \
    -state OFF {-supply_expr {(power == OFF) && (ground == OFF)}}

add_power_state -supply SB1 \
    -state ON {-supply_expr {(power == {FULL_ON 1.2}) && \
                           (ground == {FULL_ON 0.0})}} \
    -state OFF {-supply_expr {(power == OFF) && (ground == OFF)}}

add_power_state -supply SB2 \
    -state ON {-supply_expr {(power == {FULL_ON 1.2}) && \
                           (ground == {FULL_ON 0.0})}} \
    -state OFF {-supply_expr {(power == OFF) && (ground == OFF)}}

# Lower level power state information
create_power_state_group GA
add_power_state -group GA \
    -state run {-logic_expr {SA==ON && SA1==ON && SA2==ON}} \
    -state idle {-logic_expr {SA==ON && SA1==OFF && SA2==ON}} \
    -state doze {-logic_expr {SA==ON && SA1==OFF && SA2==OFF}}
create_power_state_group GB
add_power_state -group GB \
    -state run {-logic_expr {SB==ON && SB1==ON && SB2==ON}} \
    -state idle {-logic_expr {SB==ON && SB1==OFF && SB2==ON}} \
    -state doze {-logic_expr {SB==ON && SB1==OFF && SB2==OFF}}

# Top level power state information
create_power_state_group GTOP
add_power_state -group GTOP \
    -state run_run {-logic_expr {GA==run && GB==run}} \
    -state run_idle {-logic_expr {GA==run && GB==idle}} \
    -state idle_idle {-logic_expr {GA==idle && GB==idle}} \
    -state doze_doze {-logic_expr {GA==doze && GB==doze}}

```

### 5.3.44.13 Notes

- ❖ Creating a group using `create_power_state_group` is a must before using it in the `add_power_state` command.
- ❖ There are two ways of adding a state in a group, `supply_set`, `supply_set_handle`:
  - ◆ Either specify all the states in one `add_power_state` command:
    - ❖ `create_power_state_group top_G`
    - ❖ `add_power_state -group top_G -state ON_12 {-logic_expr {....}} \ -state ON_10 {-logic_expr {....}} -state OFF {-logic_expr {....}}`
  - ◆ Or specify different states in different `add_power_state` command. In this case, `-update` will be required for each successive `add_power_state` command, that is:
    - ❖ `create_power_state_group top_G`
    - ❖ `add_power_state -group top_G -state ON_12 {-logic_expr {....}}`
    - ❖ `add_power_state -group top_G -state ON_10 {-logic_expr {....}} -update`
    - ❖ `add_power_state -group top_G -state OFF {-logic_expr {....}} -update`

### 5.3.44.14 Limitations

- ❖ Existing PST checks like `PST_STATE_MULTIPLE`, `PST_STATE_INVALID`, `PST_VOLATEGE_DROPPED`, `PST_VOLTAGE_UNUSED`, `PST_BIAS_PATH`, `PST_SUPPLY_MULTIPLE` and so on are not reported for this enhanced `add_power_state` support.
- ❖ Query command `analyze_invalid_power_state` is not supported for this feature.

### 5.3.44.15 New Query Commands Introduced

The following query commands are added in VC LP.

- ❖ `get_supply_set_states`
- ❖ `report_supply_state`
- ❖ `get_power_groups`
- ❖ `report_power_group`
- ❖ `get_group_states`
- ❖ `report_group_state`

#### 5.3.44.15.1 The `get_supply_set_states` Command

Use the `get_supply_set_states` command to get a collection of the power states of the supply sets.

##### Use Model

```
vc_static_shell> get_supply_set_states -help
Usage: get_supply_set_states      # Returns a collection of power states from supply sets
       [-of_objects <supply_sets>]
                                         (Specifies supply set name patterns or collections)
       [-quiet]                      (Suppresses all messages. Syntax error messages are not
                                         suppressed)
       [<patterns>]                  (List of supply state name patterns)
```

##### Example

```
vc_static_shell> get_supply_set_states
{"core1_ss/core1_pd_12", "core1_ss/core1_pd_off", "core2_ss/core2_pd_12",
"core2_ss/core2_pd_off", "top_ss/top_pd_12"}
```

```
vc_static_shell> get_supply_set_states core1*
 {"core1_ss/core1_pd_12", "core1_ss/core1_pd_off"}
```

### 5.3.44.15.2 The report\_supply\_state Command

Use the `report_supply_state` Command to get a report of specific supply power states.

#### Use Model

```
vc_static_shell> report_supply_state -help
Usage: report_supply_state      # Reports specific supply power states
       [-only_invalid]          (Include only invalid PST states)
       [-only_valid]            (Include only valid PST states)
       [<pst_state>]           (List of supply power state name patterns or collections)
```

#### Example

```
vc_static_shell> report_supply_state -only_valid core2*
Supply State      : core2_pd_12
Full Name        : core2_ss/core2_pd_12
Supply Expression : ((power == `{ FULL_ON 1.2 }) && (ground == `{ FULL_ON 0 }))
Logic Expression  : (none)
-----
Supply State      : core2_pd_off
Full Name        : core2_ss/core2_pd_off
Supply Expression : ((power == `{ OFF }) && (ground == `{ FULL_ON 0 }))
Logic Expression  : (none)
```

### 5.3.44.15.3 The get\_power\_groups Command

Use the `get_power_groups` Command to get a collection of the power group objects.

#### Use Model

```
vc_static_shell> get_power_groups -help
Usage: get_power_groups      # Returns a collection of Power Group objects
       [-quiet]                (Suppresses all messages. Syntax error messages are not
                                suppressed)
       [<patterns>]            (List of power group name patterns)
```

#### Example

```
vc_static_shell> get_power_groups
 {"core1_core2_G", "top_G"}
```

### 5.3.44.15.4 The report\_power\_group Command

Use the `report_power_group` to get a report of specific power groups.

#### Use Model

```
vc_static_shell> report_power_group -help
Usage: report_power_group      # Reports specific power groups
       [<group>]             (List of power group name patterns or collections)
```

## Example

```
vc_static_shell> report_power_group

Group Name      : core1_core2_G
Full Name       : core1_core2_G
State:          Logic Expression|
core1_core2_G/all_12: (core1_ss==core1_pd_12) && (core2_ss==core2_pd_12) |
core1_core2_G/core1_off: (core1_ss==core1_pd_off) && (core2_ss==core2_pd_12) |
core1_core2_G/core2_off: (core1_ss==core1_pd_12) && (core2_ss==core2_pd_off) |
-----
Group Name      : top_G
Full Name       : top_G
State:          Logic Expression|
top_G/all_12:   (top_ss==top_pd_12) && (core1_core2_G==all_12) |
top_G/core1_off: (top_ss==top_pd_12) && (core1_core2_G==core1_off) |
top_G/core2_off: (top_ss==top_pd_12) && (core1_core2_G==core2_off) |
-----
```

### 5.3.44.15.5 The get\_group\_states Command

Use the `get_group_states` command to get a collection of the power states from power groups.

#### Use Model

```
vc_static_shell> get_group_states -help
Usage: get_group_states      # Returns a collection of power states from power groups
      [-of_objects <power_groups>]
                           (Specifies power group name patterns or collections)
      [-quiet]           (Suppresses all messages. Syntax error messages are not
                           suppressed)
      [<patterns>]       (List of group state name patterns)
```

## Example

```
vc_static_shell> get_group_states
{"core1_core2_G/all_12", "core1_core2_G/core1_off", "core1_core2_G/core2_off",
 "top_G/all_12", "top_G/core1_off", "top_G/core2_off"}
```

### 5.3.44.15.6 The report\_group\_state Command

Use the `report_group_state` command to get a report of the power states of a specific group.

#### Use Model

```
vc_static_shell> report_group_state -help
Usage: report_group_state      # Reports specific group power states
      [-only_invalid]        (Include only invalid PST states)
      [-only_valid]          (Include only valid PST states)
      [<pst_state>]         (List of group power state name patterns or collections)
```

## Example

```
vc_static_shell> report_group_state -only_valid top_G/all?12

Group State      : all_12
Full Name       : top_G/all_12
Logic Expression: (top_ss==top_pd_12) && (core1_core2_G==all_12)
```

```
Expanded Supplies      : ((top_ss.power==top_ss/top_pd_12) &&
 (top_ss.ground==top_ss/top_pd_12)) && (((core1_ss.power==core1_ss/core1_pd_12) &&
 (core1_ss.ground==core1_ss/core1_pd_12)) && ((core2_ss.power==core2_ss/core2_pd_12) &&
 (core2_ss.ground==core2_ss/core2_pd_12)))
```

### 5.3.45 Enhancement to State Propagation of add\_power\_state

The current support for `add_power_state` (called as old semantics in the rest of this section) has the following definition:

“A state defined on a supply set through the `add_power_state` command gets interpreted as a port state on the functional net that appears in the `supply_expr`. The state name of the supply set is propagated to the functional net that appears in the `supply_expr`. You can refer to this propagated state name through the `create_pst/add_pst_state` commands.”

However, the old semantics of `add_power_state`, where states are considered as port states rather than supply set states, results in issues. These issues can be broadly classified into 2 categories -

- ❖ Propagation of supply set state names
- ❖ Creation of incorrect system PST

Therefore, VC LP also supports the new definition of `add_power_state` (called as new semantics in the rest of this section), which treats the states added on supply sets through the `add_power_state` command as boolean expressions, and not as supply net/port states. The new semantics is enabled by default. To disable this support, set the `enable_add_power_state_semantics_change` application variable to true.

#### Propagation of supply set state names

With the old semantics, VC LP propagates the supply set state name to the functional nets for the `add_power_state` command. By default, the `add_power_state` defines the state name for the supply set alone.

#### Example

Consider the following UPF:

```
# SS1 and SS2 share the power function
create_supply_set SS1 create_supply_set SS2 -function {SS1.power}
add_power_state SS1 -state ON {-supply_expr {power == {FULL_ON 1.0}}}
add_power_state SS2 -state ON {-supply_expr {power == {FULL_ON 2.0}}}
```

#### Old Semantics Behavior

The first `add_power_state` command creates a state by name ON on `SS1.power` with voltage 1.0. The second `add_power_state` command specifies the same state name for the same supply net (as `SS1.power` and `SS2.power` are connected and in the same netgroup), but with a different voltage value. As it is not possible to have two states on a supply net with the same name but different voltage values, VC LP errors out for the second `add_power_state` command (saying that a conflicting power state has been found for state ON).

#### New Semantics Behavior

The state name used in the `add_power_state` command is a property of the supply set, and not that of the functional nets. So, the expected behavior is that the second `add_power_state` command should not result in an error. In other words, state ON should be allowed on `SS2`.

## Referring to the supply net states in a PST

Consider the below UPF.

```
create_supply_set SS1
add_power_state SS1 -state ON1 {-supply_expr {power == {FULL_ON 1.0}}}
create_pst MY_PST -supplies {SS1.power}
add_pst_state STATE -pst MY_PST {ON1}
```

### Old semantics behavior

You can refer to the propagated state names (seen on the functional nets) in `create_pst`/`add_pst_state` commands. State ON1 is created on the supply net SS1.power. So, you can refer to this propagated state name from the `add_pst_state` command.

### New semantics behavior

The state created through the `add_power_state` command is referred from the supply set alone. New semantics will error out at `add_pst_state`.

## Creation of incorrect system PST

The system PST tells the different states that the design can operate on. When states are created on supply sets using the `add_power_state` command, the system PST needs to be built in terms of the supply set states, and not in terms of the functional net states.

### Example

Consider the following UPF -

```
# SS1 and SS2 share the power function
create_supply_set SS1
create_supply_set SS2 -function {SS1.power}

add_power_state SS1 -state ON1 {-supply_expr {power == {FULL_ON 1.0}}}
add_power_state SS1 -state ON2 {-supply_expr {power == {FULL_ON 2.0}}}
add_power_state SS2 -state ON3 {-supply_expr {power == {FULL_ON 2.0}}}
```

### Old semantics behavior

States created using the `add_power_state` are interpreted as port states. In the above UPF, SS1.power and SS2.power are connected and so are in the same netgroup. The system PST will list SS1.power as having two states, one with voltage 1.0, and the other with voltage 2.0.

### New Semantics behavior

The states defined through `add_power_state` must be considered as supply set states. Going with this interpretation, SS1 can operate either in ON1 state or in ON2 state and SS2 can operate in ON3 state alone. But the two supply sets share the power net. For a system state to be valid, the voltages of the shared net must match. This will happen only when -

`SS1 == ON2 and SS2 == ON3`

This is the only valid state in the system PST as per new semantics.

### 5.3.45.1 Resolving Issues of add\_power\_state

By default, VC LP resolves the issues with the `add_power_state` by treating the states added on supply sets through the `add_power_state` command as boolean expressions, and not as supply net/port states with the new semantics.

This section lists how each of the issues described in earlier sections are addressed with the new semantics.

### 5.3.45.1.1 Propagation of Supply Set State Names

The behavior with respect to the state names is updated as follows:

1. The state name specified in `add_power_state` is used to create a state on the supply set alone.
2. The state name is not propagated to the functional nets.
3. An internal name is used while creating the states on the functional nets
4. As internal state names are created on the functional nets, you cannot refer to these states in `create_pst`/`add_pst_state` commands.
5. The supply set states can only be referred in groups.

### 5.3.45.1.2 Creation of System PST

With the new semantics of `add_power_state`, each supply set state created through the `add_power_state` command is considered as a boolean expression in terms of the functional nets. So, all the states created for a supply set can be conceptualized as a mini-PST.

With the new support, the following rules are used to create system PST:

1. States on a supply set forms a mini PST.
2. Such mini-PSTs are created for each supply set.
3. The mini-PSTs are then merged together with the user specified PSTs (`create_pst` OR `create_power_state_group`) to create the final derived PST for the system.

If a system state results in the same netgroup having two different voltages, then that system state is dropped.

#### Example 1: When Supply Sets are Unconnected

Consider the following UPF where the supply sets SS1 and SS2 are unconnected.

```
create_supply_set SS1 create_supply_set SS2
add_power_state SS1 \ -state ON1 {-supply_expr {power == {FULL_ON 1.0} && ground == {OFF}}} add_power_state SS1 \ -state ON2 {-supply_expr {power == {FULL_ON 2.0} && ground == {OFF}}}
add_power_state SS2 \ -state ON3 {-supply_expr {power == {FULL_ON 2.0} && ground == {OFF}}} add_power_state SS2 \ -state ON4 {-supply_expr {power == {FULL_ON 3.0} && ground == {OFF}}}
```

As every state of a supply set is considered as a boolean expression, the mini-PSTs for supply sets SS1 and SS2 are shown below. Here, the state names listed for the functional nets are the internally created state names (the suffix 'INT' stands for 'internal').



#### Note

Actual internal state names may differ than listed below. This is for illustration purpose only.

**Table 5-6 Mini-PST for SS1**

STATE \ FUNCTION	SS1.power	SS1.ground
ON1	SS1_ON1_INT	SS1_ON1_INT
ON2	SS1_ON2_INT	SS1_ON2_INT

**Table 5-7 Mini-PST for SS2**

STATE \ FUNCTION	SS2.power	SS2.ground
ON3	SS2_ON3_INT	SS2_ON3_INT
ON4	SS2_ON4_INT	SS2_ON4_INT

The system PST is created by merging the mini-PSTs. In other words, the system PST has four states (cross-product of SS1's mini-PST having 2 states and SS2's mini-PST having 2 states). As the two supply sets are unconnected, all the four states of the system PST is valid. The system PST is as described [Table 5-8](#).

**Table 5-8 Result after merging the mini-PSTs**

STATE \ FUNCTION	SS1.power	SS1.ground	SS2.power	SS2.ground
SYSTEM_STATE_1	SS1_ON1_INT	SS1_ON1_INT	SS2_ON3_INT	SS2_ON3_INT
SYSTEM_STATE_2	SS1_ON1_INT	SS1_ON1_INT	SS2_ON4_INT	SS2_ON4_INT
SYSTEM_STATE_3	SS1_ON2_INT	SS1_ON2_INT	SS2_ON3_INT	SS2_ON3_INT
SYSTEM_STATE_4	SS1_ON2_INT	SS1_ON2_INT	SS2_ON4_INT	SS2_ON4_INT

The system PST has 4 states as shown in [Table 5-9](#).

**Table 5-9 System PST Simplified to (in terms of supply set states)**

STATE \ FUNCTION	SS1	SS2
SYSTEM_STATE_1	ON1	ON3
SYSTEM_STATE_2	ON1	ON4
SYSTEM_STATE_3	ON2	ON3
SYSTEM_STATE_4	ON2	ON4

### Example 2: When Supply Sets are Partially Connected)

Consider the following UPF where the supply sets SS1 and SS2 are partially connected.

```
create_supply_set SS1
create_supply_set SS2 -function {power SS1.power}

add_power_state SS1 \
    -state ON1 {-supply_expr {power == {FULL_ON 1.0} && ground == {OFF}}}
add_power_state SS1 \
    -state ON2 {-supply_expr {power == {FULL_ON 2.0} && ground == {OFF}}}

add_power_state SS2 \
    -state ON3 {-supply_expr {power == {FULL_ON 2.0} && ground == {OFF}}}
add_power_state SS2 \
    -state ON4 {-supply_expr {power == {FULL_ON 3.0} && ground == {OFF}}}
```

The mini-PSTs created for SS1 and SS2 are the same as listed in example 1. The system PST is created by merging the mini-PSTs. But in this example, the two supply sets share the power net. Given this, [Table 5-10](#)

shows how the PST looks like after merging the mini-PSTs (see the Comments column to understand what happens to each of the system states). The voltages of each of the states are listed within square brackets for easier understanding of the final result.

**Table 5-10 Result after merging the mini-PSTs**

STATE \ FUNCTION	SS1	SS2	Comments
SYSTEM_STATE_1	ON1 [1.0, OFF]	ON3 [2.0, OFF]	Will be dropped because of conflicting voltages for the shared power net
SYSTEM_STATE_2	ON1 [1.0, OFF]	ON4 [3.0, OFF]	Will be dropped because of conflicting voltages for the shared power net
SYSTEM_STATE_3	ON2 [2.0, OFF]	ON3 [2.0, OFF]	Valid state, as the shared power net is at the same voltage 2.0 for both ON2 and ON3
SYSTEM_STATE_4	ON2 [2.0, OFF]	ON4 [3.0, OFF]	Will be dropped because of conflicting voltages for the shared power net

So, the system PST has only one valid state as shown in [Table 5-11](#).

**Table 5-11 System PST**

STATE \ FUNCTION	SS1	SS2
SYSTEM_STATE_1	ON2 [2.0, OFF]	ON3 [2.0, OFF]

### Example 3: Mixture of Single-function add\_power\_state and Multiple-function add\_power\_state)

Consider the following UPF where the supply set SS1 has some states with one function and some states with two functions.

```
create_supply_set SS1

add_power_state SS1 \
    -state ON1 {-supply_expr {power == {FULL_ON 1.0} && ground == {OFF}}}

add_power_state SS1 \
    -state ON2 {-supply_expr {power == {FULL_ON 2.0} \
        && ground == {FULL_ON 0.0}}}

add_power_state SS1 \
    -state ON3 {-supply_expr {power == {FULL_ON 3.0}}}
```

As explained earlier, every state created for a supply set through the `add_power_state` command is considered as a boolean expression in terms of the functional nets. States ON1 and ON2 are already created in terms of both power and ground functions. But, state ON3 is created only in terms of the power function. So, state ON3, when considered as a boolean expression in terms of the functional nets, looks like:

```
power == 3.0 && ground == *
```

"\*\*" implies that the 'ground' function can take all possible states present/seen on the netgroup. Now, the ground net SS1.ground has the below 2 states created on it by virtue of states ON1 and ON2 -

1. {OFF}
2. {FULL\_ON 0.0}

So, the “\*\*” for ground will expand to the above 2 states.

So, the mini-PST for SS1 (and the system PST, as there are no other supply sets) is as shown in [Table 5-12](#).

**Table 5-12 -System PST**

STATE \ FUNCTION	SS1.power	SS1.ground
SYSTEM_STATE_1	1.0	OFF
SYSTEM_STATE_2	2.0	0.0
SYSTEM_STATE_3	3.0	OFF
SYSTEM_STATE_4	3.0	0.0

### 5.3.45.2 Distinguishing between Old Semantics and New Semantics

You can control if VC LP should interpret the block's UPF using the old semantics and new semantics using the `enable_state_propagation_in_add_power_state` design attribute in block's UPF.

The `enable_state_propagation_in_add_power_state` design attribute is a scope-level boolean attribute that defines whether the block's UPF has to be interpreted taking the old semantics into consideration or the new semantics.

#### 5.3.45.2.1 Using the `enable_state_propagation_in_add_power_state` design attribute

The following example shows how the `enable_state_propagation_in_add_power_state` design attribute can be set in

```
# Setting the design attribute at the top-most scope
set_design_attributes -elements {.} \
    -attribute enable_state_propagation_in_add_power_state FALSE

# Setting the design attribute on a cell named 'block'
set_design_attributes -elements {block} \
    -attribute enable_state_propagation_in_add_power_state TRUE
```

If the value of the design attribute on a block is TRUE, then it means that the block has to be interpreted in the old semantics style. If the value is FALSE, then it means that the block has to be interpreted in the new semantics style. If both the semantics are mixed in a design, then a block written in the old semantics style must be integrated within a design written in the new semantics style. This is the only valid configuration.

Upon trying reverse configuration, that is value of FALSE within TRUE, VC LP reports the following error:

*my\_file.upf:6: [Error] UPF\_ATTR\_VALUE\_UNSUPPORTED: Attribute value conflicting with parent*

*Attribute 'enable\_state\_propagation\_in\_add\_power\_state' has been specified in an ancestor hierarchy as 'true'. Cannot specify attribute as 'false' in scope 'top/block' as new add\_power\_state semantics within old semantics is unsupported.*

When power group tries to refer to supply set state with old semantics from a scope with `enable_state_propagation_in_add_power_state true`, the SUPPLY\_SET\_SEMANTICS error message is reported.

Attribute `enable_state_propagation_in_add_power_state` value for a scope is allowed to be changed only prior to adding states on supply sets in the scope. When trying to set the attribute after setting the supply states through add power state, `UPF_SET DESIGN_ATTRIBUTE_ILLEGAL` error is reported.

You can also create power state groups in hierarchies having state propagation enabled, that is, you can create power state groups in parent scope where state propagation enabled, and then use these groups to refer to power states of the child scope where state propagation is disabled.

### Example

*BLOCK.upf*

```
state propagation = FALSE
set_design_attributes -elements { . } -attribute
enable_state_propagation_in_add_power_state FALSE
create_pst PST
add_pst_state S1 -pst PST
create_supply_set SS
add_power_state SS -state S2 {power == 1.0 && ground == 0.0}
create_power_state_group GROUP
add_power_state GROUP -state S3
```

*# TOP.upf*

```
#state propagation = TRUE
set_design_attributes -elements { . }
-attribute enable_state_propagation_in_add_power_state TRUE
create_pst TOP_PST
add_pst_state S0 -pst TOP_PST
create_power_state_group TOP_GROUP
add_power_state TOP_GROUP -state GS1
```

```
{TOP_PST == S0 && BLOCK/PST == S1 && BLOCK/SS == S2 && BLOCK/GROUP == S3}
```

The `SUPPLY_SET_SEMANTICS` error message is reported when power state group tries to refer to supply set state written in a scope with the state propagation disabled from a scope with state propagation enabled/disabled.

#### 5.3.45.2.2 Using the `upf_enable_state_propagation_in_add_power_state` Application Variable

You can also globally control if the whole design must use the old semantics or new semantics of `add_power_state`. If you want VC LP to use the old semantics of `add_power_state`, then set the `upf_enable_state_propagation_in_add_power_state` application variable to true.

If this application variable is set to false, VC LP uses the new semantics for `add_power_state`.



#### Note

The TCL variable, if set to true, will have the highest precedence over design attribute "enable\_state\_propagation\_in\_add\_power\_state" setting(s) in the design. Default value is false.



#### Note

If the `upf_enable_state_propagation_in_add_power_state` application variable is set to false (by default), then new semantics is considered for entire design as long as it does not have `enable_state_propagation_in_add_power_state` design attribute. If the design attribute is also set, then VC LP gives priority to it.

### 5.3.45.3 Behavioral Differences with the Design Attribute

[Table 5-13](#) provides the behavioral differences when the `enable_state_propagation_in_add_power_state` design attribute is set to TRUE or FALSE.

**Table 5-13 Behavioral Differences with the Design Attribute**

Features \ Semantics	<code>enable_state_propagation_in_a_dd_power_state = TRUE (old semantics)</code>	<code>enable_state_propagation_in_add_power_state = FALSE (new semantics)</code>
Boolean operators (&&)	Are not allowed in supply_expr of add_power_state.	Are allowed in supply_expr of add_power_state.
State name propagation	Supply set state names will be propagated to the supply nets.	Supply set state names will not be propagated to the functional nets. Internal state names will be created for the functional nets.
Interpreting power states	Mini-PSTs are not created. Supply set states are considered as port states.	Mini-PSTs are created for the supply set states.
Referring to the states	The propagated state names can be used in <code>create_pst/add_pst_state</code> commands.	The supply set states can be referred only in GROUPs.
Using <code>create_pst</code>	Is allowed. The propagated supply set state names can be referred in the PST.	Is allowed. However, the states that can be referred in the PST are the ones added through the <code>add_port_state</code> command.
Using <code>create_power_stat e_group</code>	Is not allowed.	Is allowed.

### 5.3.46 Support for add\_supply\_state Command

VC LP supports the `add_supply_state` command to capture the PST behavior of the reference only supply net and for correct insertion of always ON and level shifters. The reference only supply nets cannot be connected to any supply ports in the scope.

The `add_supply_state` command defines a named supply state for a supply object. If a voltage value is specified, the supply net state is FULL\_ON, and the voltage value is the specified value. Otherwise, if off is specified, the supply set state is OFF.

This command behaves similar to the `add_port_state`, but on supply nets. VC LP has already supporting supply state definitions for supply nets under an application variable which is not LRM compatible. The `add_supply_state` support is aligned with LRM.

#### Syntax

```
add_supply_state object_name {state {name<nom|off>}}
```

Where:

- ❖ `object_name`: The name of the supply port, supply net, or supply set function. Hierarchical names are allowed.
- ❖ `-state {name<nom|off>}`: The name and value for a state of the supply object. The value can be a nominal voltage or off.

### 5.3.47 Support for set\_equivalent Command

VC LP supports the `set_equivalent` UPF command. The `set_equivalent` directive, introduced in IEEE 1801-2013 allows you to declare two supply nets (or supply sets) as electrically or functionally equivalent. The `set_equivalent` command provides the support for supply set association and supply connection.

The IEEE 1801 standard defines two types of equivalence:

- ❖ Electrical equivalence
- ❖ Functional equivalence

#### 5.3.47.1 Electrical Equivalence

Two supplies are electrically equivalent if they are connected (whether the connections are evident or not in the design) without any intervening switches, and therefore guaranteed to have the same value at all times from the perspective of any load.

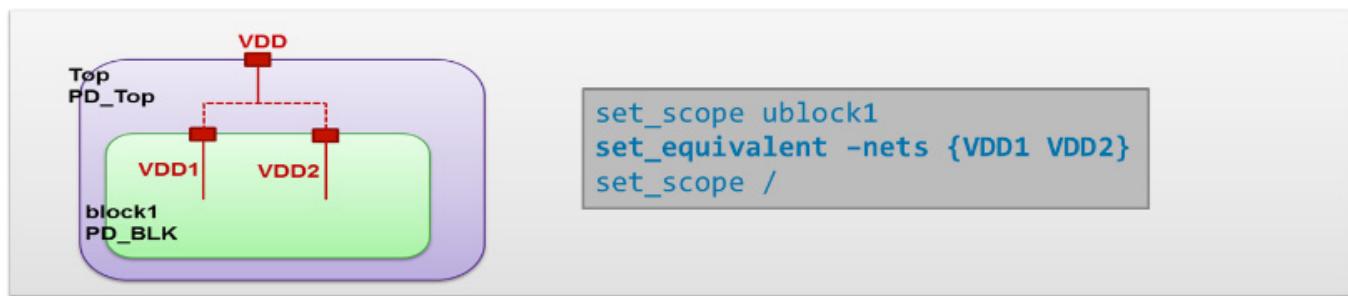
##### Syntax

```
set_equivalent -nets <supply_nets/ports>|-sets <supply_sets>
```

##### Use Model

- ❖ You can declare electrical equivalence to model a direct connection between two supply nets that exists outside the design scope.
- ❖ You can declare electrical equivalence only on supplies which originate, and are connected, outside of the design. Scenarios such as
  - ◆ Supplies driven by a single top-level input port, that is, connection exists outside the design.
  - ◆ Output PG pins of a macro, black box design (that is, connection exists inside the cell), or pad cell.

**Figure 5-112 Electrically Equivalent Example**



#### 5.3.47.2 Functionally Equivalent

Two supplies are functionally equivalent if they function identically from the perspective of any load, either as a result of being electrically equivalent, or due to independent but parallel circuitry. All electrically equivalent supplies are also functionally equivalent.

##### Syntax

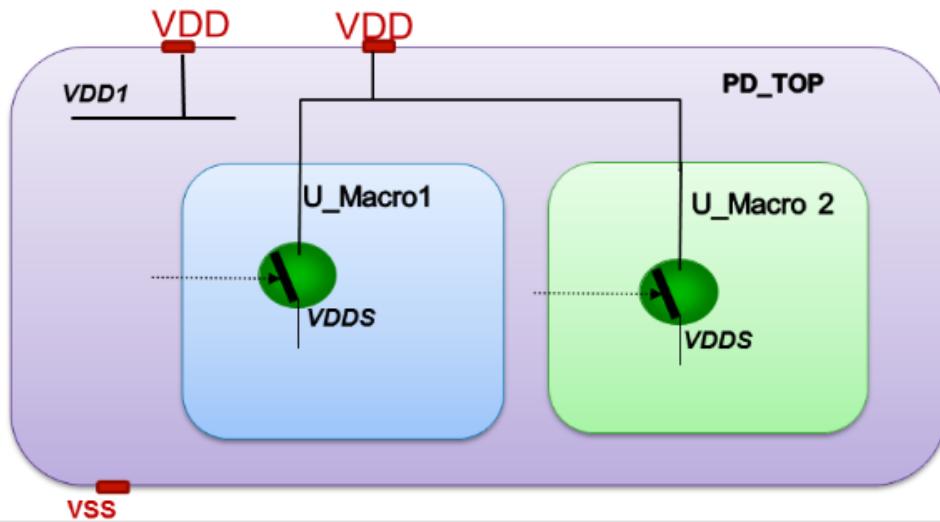
```
set_equivalent -nets <supply_nets/ports>|-sets <supply_sets> -function_only
```

##### Use Model

- ❖ Functional equivalence can be declared between any supplies in the design

- ❖ Declared supplies should not have contradictory power states

**Figure 5-113 Functionally Equivalent Examples**



```
set_equivalent -function_only \
-nets {U_Macro1/VDDS U_Macro2/VDDS}
```

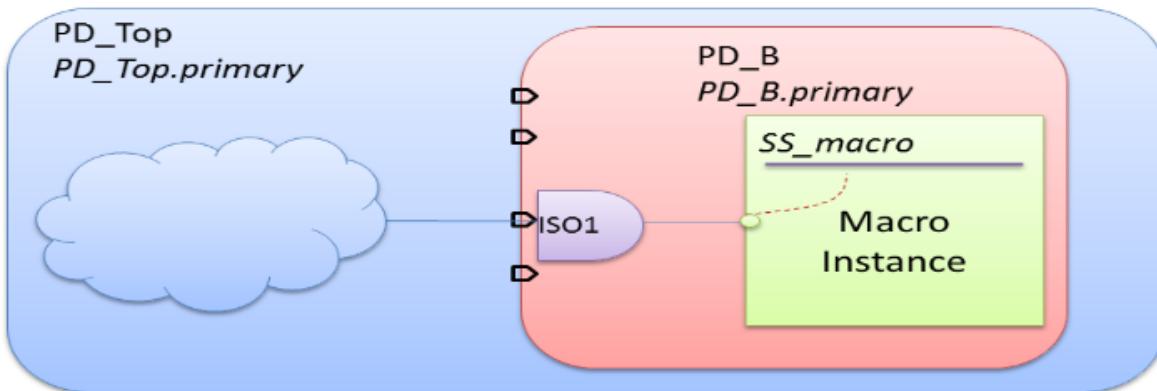
### 5.3.47.3 Example - Use Case

In the example in [Figure 5-114](#), the isolation strategy ISO\_SRAT1 associates with isolation cell ISO1 as PD\_B.primary and u\_macro/SS\_macro are equivalent supplies.

The following is the UPF for the example in [Figure 5-114](#).

```
set_isolation ISO_SRAT1 -domain PD_B -applies_to input \
-source PD_Top.primary -sink PD_B.primary
set_equivalent -function_only -sets {PD_B.primary u_macro/SS_macro}
```

**Figure 5-114 Set Equivalent Example**



### 5.3.47.4 Support for set\_equivalent Transitivity

VC LP supports the transitive law for the `set_equivalent` command.

- ❖ For electrical equivalence, this applies both to declared equivalence (with `set_equivalent`) and direct connection between the supplies (with `connect_supply_net`). So, if supply A is directly connected to supply B, and supply B is declared equivalent to supply C, then supply A is also electrically equivalent to supply C.
- ❖ Functional-only equivalence is also transitive. If supply A is function-only equivalent to supply B, and supply B is function-only equivalent to supply C, then supply A is also function-only equivalent to supply C.
- ❖ Finally, functional-only equivalence is transitive across electrically equivalent supplies as well. That is, if supply A is electrically equivalent to supply B, and supply B is function-only equivalent to supply C, then supply A is also function-only equivalent to supply C.

### 5.3.47.5 Parser Checks for `set_equivalent` Support

The following parser checks are supported for the `set_equivalent` command:

- ❖ The following error message is reported if the design entities mentioned in -nets/-sets are not supply nets/ports or supply sets.  
*[Error] UPF\_OBJECT\_NOT\_FOUND\_ERROR: Object not found*
- ❖ The following error message is reported if you declare a domain-dependent supply net to be equivalent to another supply.  
*[Error] UPF\_SET\_EQUIVALENT\_SUPPLY\_DOMAIN\_DEPENDENT: Domain dependent supply declared equivalent.*
- ❖ The following error message is reported if the supplies of type power and ground are declared as equivalent.  
*[Error] UPF\_SET\_EQUIVALENT\_SUPPLY\_TYPE\_MISMATCH: Equivalent supply type not matching*
- ❖ The following error message is reported if two supply sets declared as equivalent does not have same set of functions.  
*[Error] UPF\_SET\_EQUIVALENT\_SUPPLY\_FUNC\_MISMATCH: Equivalent supply set function mismatch*
- ❖ The following error message is reported for equivalent supplies that have different port states with the same name.  
*[Error] UPF\_EQUIVALENT\_FUNC\_ONLY\_STATES\_CONFLICT: Different port states with the same name*
- ❖ The following warning message is reported if only one net/set is used in `set_equivalent` command.  
*[Warning] UPF\_SET\_EQUIVALENT\_SINGLE\_SUPPLY: Single supply in set\_equivalent command*
- ❖ The following error message is reported when nets and sets are simultaneously used in the `set_equivalent` command.  
*[Error] TCL\_OPT\_ATMOST\_ONE\_OF: Incorrect command options*
- ❖ The following error message is reported when an empty list is used in the `set_equivalent` command.  
*[Error] TCL\_OPT\_EMPTY\_LIST: Option value list specified is empty*
- ❖ The following error message is reported if electrical equivalence is declared on a set of supplies unless one of these circumstances applies.

[Error] UPF\_SET\_EQUIVALENT\_INVALID\_DRIVER: Invalid driver for equivalent supplies

- ◆ Each supply is driven only by a top-level supply port;
- ◆ Each supply is driven only by an output or internal PG pin on the same macro cell;
- ◆ Each supply is driven only by an output supply port on the same blackbox cell;
- ◆ Each supply is driven only by an output of a pad cell;
- ◆ Each supply is already physically connected to the others.

### Examples

- ✧ Case 1: Electrical equivalence between output or internal pg pin of the same macro instance

```
set_equivalent -nets {MACRO/out MACRO/internal}
```

VC LP allows to define as electrical equivalence in this case.

- ✧ Case 2: Electrical equivalence between output or internal pg pin of different macro instances instantiated using same/different Macro DB

```
set_equivalent -nets {MACRO1/internal MACRO2/internal}
```

VC LP reports the UPF\_SET\_EQUIVALENT\_INVALID\_DRIVER error in this case.

- ✧ Case 3: Electrical equivalence between output supply port of same black box instance

```
set_equivalent -nets {BBOX/out1 BBOX/out2}
```

VC LP allows to define as electrical equivalence in this case.

- ✧ Case 4: Electrical equivalence between output supply port of different black box instance

```
set_equivalent -nets {BBOX1/out BBOX2/out}
```

VC LP reports the UPF\_SET\_EQUIVALENT\_INVALID\_DRIVER error in this case.

- ✧ Case 5: Electrical equivalence between output pg pin of pad cell

```
set_equivalent -nets {PAD1/out PAD2/out}
```

VC LP allows to define as electrical equivalence in this case.

- ✧ Case 6: Electrical equivalence between internal pg pin of pad cell

```
set_equivalent -nets {PAD1/int PAD2/int}
```

VC LP reports the UPF\_SET\_EQUIVALENT\_INVALID\_DRIVER error in this case.

## 5.3.47.6 VC LP Tcl Commands for set\_equivalent

### 5.3.47.6.1 The get\_equivalent\_nets Command

#### Syntax

```
%vc_static_shell> get_equivalent_nets -help
Usage: get_equivalent_nets      # Returns a collection of supply nets/ports equivalent to
given supply
      [-function_only]      (Returns functionally equivalent nets/ports)
      <supply_net | supply_port>
                           (SupplyNet/Port Name)
```

#### Example

```
set_equivalent -nets {VDD1 VDD2}
set_equivalent -nets {VDD3 VDD4} -function_only

%vc_static_shell> get_equivalent_nets VDD1
```

```
{ "VDD2" }

%vc_static_shell> get_equivalent_nets VDD3 -function_only
{ "VDD4" }
```

### 5.3.47.6.2 Syntax for get\_equivalent\_sets

#### Syntax

```
%vc_static_shell> get_equivalent_sets -help
Usage: get_equivalent_sets      # Returns a collection of equivalent supply sets
       [-function_only]        (Returns functionally equivalent supply sets)
       <supply_set>           (SupplySet Name)
```

#### Example

```
set_equivalent -sets {SS1 SS2}
set_equivalent -sets {SS3 SS4} -function_only
```

```
vc_static_shell> get_equivalent_sets SS1
{ "SS2" }
```

```
vc_static_shell> get_equivalent_sets SS3 -function_only
{ "SS4" }
```

### 5.3.48 Support for Power Down Function

The power\_down\_function attribute specifies the boolean condition when the cell's output pin is switched off by the power and ground pins (when the cell is in off mode due to the external power pin states). You specify the power\_down\_function attribute for combinational and sequential cells. For simple or complex sequential cells, the power\_down\_function attribute also determines the condition of the cell's internal state.

#### Example

```
cell ("retention_dff") {
  pg_pin(VDD) {
    voltage_name : VDD;
    pg_type : primary_power;
  }
  pg_pin(VSS) {
    voltage_name : VSS;
    pg_type : primary_ground;
  }
  pin ("D") {
    direction : "input";
  }
  pin ("CP") {
    direction : "input";
  }
  ff(IQ,IQN) {
    next_state : "D" ;
    clocked_on : "CP" ;
    power_down_function : "!VDD + VSS" ;
  }
  pin ("Q") {
    function : " IQ ";
  }
}
```

```

direction : "output";
power_down_function : "!VDD + VSS";
}
...
}

```

The power\_down\_function for an output pin may contain additional PG pins, other than the ones specified as RPP and RGP.

### Example

```

cell(Buffer_Type_LH_Level_shifter) {
is_level_shifter : true;
level_shifter_type : LH ;
pg_pin(VDD1) {
voltage_name : VDD1;
pg_type : primary_power;
std_cell_main_rail : true;
}
pg_pin(VDD2) {
voltage_name : VDD2;
pg_type : primary_power;
}
pg_pin(VSS) {
voltage_name : VSS;
pg_type : primary_ground;
}
pin(A) {
direction : input;
related_power_pin : VDD1;
related_ground_pin : VSS;
input_voltage_range ( 0.7 , 0.9 );
}
pin(Z) {
direction : output;
related_power_pin : VDD2;
related_ground_pin : VSS;
function : "A";
power_down_function : "!VDD1 + !VDD2 + VSS";
output_voltage_range (1.1 , 1.3 );
}
}

```

If the value of the power\_down\_function is 1, it means that the pin output is powered down.

To calculate the value of the power\_down\_function expression, power pin with value OFF is assumed to be 0 and ground pin that is OFF is assumed to be 1.

#### 5.3.48.1 Adding power\_down\_function using Tcl Commands

You can add a power\_down\_function attribute using the set\_attribute or set\_pin\_model Tcl commands. For Example

```
set_pin_model {INTOBUF} -pins {Z} -power_down_function "!VDDAON + !VDD + VSS"
```

#### 5.3.48.2 VC LP Checks for power\_down\_function

- ❖ CORR\_POWERDOWN\_STATE:

This violation is reported for cases where the power\_down\_function of an output pin becomes 1 when the immediate load supply is ON.

```
CORR_POWERDOWN_STATE (1 error/0 waived)
-----
```

```
Tag : CORR_POWERDOWN_STATE
Description : Signal from [Instance] to [ViolationSink] corrupted due to power
down function [PowerDownFunction].
Violation : LP:4
Instance : buf
PowerDownFunction : !VDD + VSS + !VDDAON
Cell : INTBUF
CellPin : Z
SourceInfo
PowerNet
NetName : VDD1
NetType : UPF
PowerMethod : FROM_UPF_POWER_DOMAIN
GroundNet
NetName : VSS
NetType : UPF
GroundMethod : FROM_UPF_POWER_DOMAIN
SinkInfo
PowerNet
NetName : VDD3
NetType : UPF
PowerMethod : FROM_UPF_POWER_DOMAIN
GroundNet
NetName : VSS
NetType : UPF
GroundMethod : FROM_UPF_POWER_DOMAIN
States
State : pst1/R4
ViolationSink : SUB_A/buf2/I
IsIsolated : true
```

#### ❖ CORR\_POWERDOWN\_FUNC

VC LP analyzes the complete LogicSource'LogicSink path for the ON - ON quadrant, and flag the driver of an ISO cell, which might be a buf/inv/LS/ISO or LogicSource itself, gets PDF=1. This violation is reported when an instance which is driving an isolation cell is becoming OFF because of the power down function.

A sample violation will look like:-

```
Tag : CORR_POWERDOWN_FUNC
Description : Instance [Instance] Pin [CellPin] which is driving an isolation
cell is becoming OFF due to power down function [PowerDownFunction]
Violation : LP:19
Instance : u_pad_ana
CellPin : C
PowerDownFunction : !VDD+!VDDPST+VSS+VSSPST
LogicSink : i_aon_snk0/reg_pad_reg/D
Cell : PDDW04DGZ_H_G
LogicSource
  PinName : u_pad_ana/C
DomainSource : u_pad_ana/C
DomainSink : i_aon_snk0/pad_in
SourceInfo
```

```

PowerNet
  NetName      : VDD_ANA
  NetType       : UPF
  PowerMethod   : FROM_UPF_CSN
GroundNet
  NetName      : VSS
  NetType       : UPF
  GroundMethod : FROM_UPF_CSN
SinkInfo
  PowerNet
    NetName      : VDD_AON
    NetType       : UPF
    PowerMethod   : FROM_UPF_POWER_DOMAIN
  GroundNet
    NetName      : VSS
    NetType       : UPF
    GroundMethod : FROM_UPF_POWER_DOMAIN
States
  State        : TOLEVEL_PST/AON_on

```

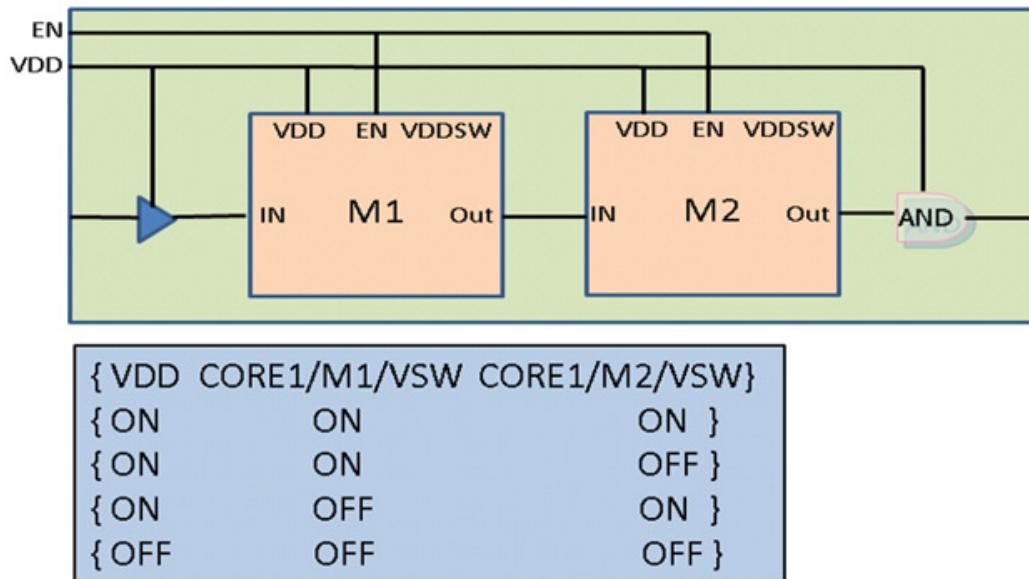
### 5.3.49 Support for -pg\_function Attribute

Prior to this release, when a macro had an internal switch, you had to manually apply the port states and power states.

If a design had 50 cells and 1000 instances, a separate UPF file had to be created for each instance and each UPF file had to be loaded separately. If instances are added or removed from the design, then all the UPF files had to be carefully updated which was a time consuming and error prone process.

For the example shown in [Figure 5-115](#), there is inconsistency between design and UPF. In the design the macro pins EN and VDD is connected to the same EN and supply source VDD. While in the UPF, ON and OFF states are defined for the same signal at same time for different macro cells.

**Figure 5-115 Example for Inconsistencies in Design and UPF**



To remove this inconsistency, you had to create dummy net in the UPF as shown in the figure below which removes the inconsistency.

```
create_supply_net dummy_net  
connect_supply_net dummnny_net -ports { CORE1/M1/VSW CORE1/M2/VSW }
```

{ VDD1 CORE1/M1/VSW CORE1/M2/VSW}		
{ON	ON	ON }
{ON	ON	OFF }
{ON	OFF	ON }
{OFF	OFF	OFF }

VC LP automatically generates internal power state tables based on the pg\_function and switch\_function which eliminates the painful process of manually applying the port states and power states.

To enable this support, set the enable\_pst\_from\_pg\_function application variable to true.

When you set the enable\_pst\_from\_pg\_function application variable to true, VC LP automatically generates internal power state tables based on the pg\_function.

The following is example of cell description with the important portions of the cell's liberty model.

```
pg_pin(VSW) {  
    pg_type :internal_power;  
    direction :internal;  
    switch_function : "E";  
    pg_function : "VDD";  
}  
pin (E) {  
    related_power_pin :VDD;  
}  
pin (I) {  
    related_power_pin :VSW;  
}  
pin (O) {  
    related_power_pin :VSW;  
}
```

VC LP generates following PST based on the pg\_function for the internal switch.

**UPF :**

```
add_port_state VDD -state {ON 1.0} -state {OFF off}
```

```
create_pst table0 -supplies {VDD M1/VDDSW M2/VDDSW}
add_pst_state state0 -pst table0 -state {ON ON ON}
add_pst_state state1 -pst table0 -state {ON OFF OFF}
add_pst_state state2 -pst table0 -state {OFF OFF OFF}
```



### Note

If you explicitly define the PST and set\_app\_var enable\_pst\_from\_pg\_function true, VC LP considers the user-defined PST and checks happens based on user-defined PST, and not the automatically created PST which VC LP generates.

#### 5.3.49.1 The analyze\_pg\_function -upf Command

You can review the new power state table generated by VC LP using analyze\_pg\_function command. This command reports the results in a UPF or text format. By default, the result is reported in text format.

##### Syntax

```
%vc_static_shell> analyze_pg_function [-upf]
```

When the -upf switch is used, the PST is reported in the UPF format which can be reused in the user-defined UPF. This helps when you want to reuse the PST in other tools which do not support the pg\_function.

##### Use Models

- ❖ %vc\_static\_shell> analyze\_pg\_function [-upf]

The following is an example of the UPF format reported:

```
create_supply_port CORE1/M1/VSW
create_supply_port CORE1/M2/VSW
create_pst ps1 -supplies {VDD CORE1/M1/VSW CORE1/M2/VSW}
add_pst_state s1 -pst ps1 -state {ON ON OFF}
add_pst_state s2 -pst ps1 -state {ON OFF OFF}
add_pst_state s3 -pst ps1 -state {OFF OFF OFF}
```

- ❖ %vc\_static\_shell> analyze\_pg\_function

The following is an example of the text format reported:

Table 1

UpfNet	:	VDD
State	:	ON
State	:	OFF (off)
Enable	:	I_0_N_2
Constant	:	0
PowerSwitches		
InternalPin	:	psw_1/TVDD (generated)

```
InternalPin : psw_2/TVDD (generated)
```

### 5.3.49.2 New Tags Introduced

The UPF\_MACRO\_NOSTATE and UPF\_PIN\_NOSTATE tags are deprecated and the following new tags are added:

- ❖ UPF\_PORTSTATE\_NOSTATE
- ❖ UPF\_PORTSTATE\_MISSING

#### 5.3.49.2.1 The UPF\_PORTSTATE\_NOSTATE Violation

The UPF\_PORTSTATE\_NOSTATE violation is reported if the liberty of the cell contains a pg\_function for the pin, and the enable\_pst\_from\_pg\_function application variable is set to false.

For more details on this violation, type `view_help UPF_PORTSTATE_NOSTATE` on the %vc\_static\_shell.

The following is example of the UPF\_PORTSTATE\_NOSTATE violation:

Tag	:	UPF_PORTSTATE_NOSTATE
Description	:	No state defined for PG pin [CellPin] of instance [Instance] ([Cell])
Violation	:	LP:4
CellPin	:	VDDSW
Instance	:	psw_1
Cell	:	MACRO
CellType	:	Macro

#### 5.3.49.2.2 The UPF\_PORTSTATE\_MISSING Violation

This violation is reported when the enable\_pst\_from\_pg\_function is set to true and the liberty has -pg\_function on the internal pin, and VC LP automatically generates the power state tables.

For more details on this violation, type `view_help UPF_PORTSTATE_MISSING` on the vc\_static\_shell.

The following is example of the UPF\_PORTSTATE\_MISSING violation:

Tag	:	UPF_PORTSTATE_MISSING
Description	:	No state defined for internal PG pin [CellPin] of instance [Instance] ([Cell]) so state constructed from library data
Violation	:	LP:1
CellPin	:	TVDD
Instance	:	CORE1/psw_1
Cell	:	MACRO_SW_1
CellType	:	Macro
PowerStateTable	:	_SNPS_GENERATED_PST_PGFUNC_0

### 5.3.50 Support for create\_power\_domain -available\_supplies

VC LP supports the -available\_supplies {supply\_set\_list} in the `create_power_domain -available_supplies {supply_set_list}` command. Its functionality is similar to the `create_power_domain -supply {extra_supply# supply_set}`.

The following rules apply for the -available\_supplies option.

- ❖ If -available\_supplies does not appear, all supply sets and supply set handles defined at the scope of the power domain are available for the power domain.
- ❖ If -available\_supplies appears with an empty string argument, only the locally available supplies are available for the power domain.
- ❖ If -available\_supplies appears with a non-empty string, the string is a list of the names of additional supply sets or supply set handles defined at the scope of the power domain that are also available for use by tools to power cells inserted into the power domain, in addition to the locally available supplies.

### Syntax Checks

- ❖ If the `create_power_domain` command has `-available_supplies` and `-supply {extra_supplies[_# xx]}` together, (either both in a single `create_power_domain` command or else one in base command and another with `-update` for same power domain), an error message is reported.

**Example**

```
create_power_domain PD1 -elements {CORE1} -available_supplies {SS_TOP} \
                     -supply {extra_supplies_0 "SS3"}
```

OR

```
create_power_domain PD1 -elements {CORE1} -available_supplies {SS_TOP}
create_power_domain PD1 -supply {extra_supplies_0 "SS3"} -update
```

*[Error] TCL\_OPT\_NOT\_TOGETHER: Incorrect command options*

*Options '-available\_supplies and extra\_supplies' must not be specified together in command 'create\_power\_domain'.*

- ❖ If `-available_supplies` with empty braces is used with the `-update` option, an error is reported.

**Example**

```
create_power_domain PD1 -include_scope -supply {primary SS_TOP}
create_power_domain PD1 -available_supplies {} -update
```

OR

```
create_power_domain PD1 -include_scope -supply {primary SS_TOP} -available_supplies
{SS1}
create_power_domain PD1 -available_supplies {} -update
```

*[Error] TCL\_OPT\_ATMOST\_ONE\_OF: Incorrect command options*

*It is not legal to specify more than one of '(-update, ATLEAST\_ONE\_OF(simulation\_only, -exclude\_elements, -include\_scope, -scope, -define\_func\_type, -available\_supplies))' in command 'create\_power\_domain'.*

- ❖ If the original `create_power_domain` command has no `-available_supplies` and later on if you update it with `-update`, then the supply is already available and hence parsing warning message is reported.

**Example**

```
create_power_domain PD1 -elements {CORE1}
create_power_domain PD1 -elements {CORE1} -available_supplies {SS_TOP} -update
```

*[Warning] UPF\_PD\_SUPPLY\_ALREADY\_AVAILABLE: Supply already available in Power Domain*

*The supply 'SS\_TOP' already available in Power Domain 'PD1' because no -available\_supplies in base command.*

- ❖ If originally the power domain has supply set SS1 available and if you update it with another supply set SS2, then both SS1 and SS2 is available for that domain.

```
create_power_domain PD1 -elements {CORE1} -available_supplies {SS1}
create_power_domain PD1 -available_supplies {SS2} -update
```

Both SS1 and SS2 are available for power domain PD1.

### 5.3.50.1 Tags Affected by This Option

The following tags are introduced at the UPF level checks for this support:

### ❖ LS\_SUPPLY\_UNAVAIL

This violation is reported when there is a level shifter strategy and for a particular strategy node, the source or sink supply is not available at -location specified in the strategy.

Consider the following UPF is defined for the example shown [Figure 5-116](#), where the UPF supply VDD is not defined for the power domain element CORE1.

```
create_supply_set SS_TOP -function {power VDD} -function {ground VSS}
create_supply_set SS1 -function {power VDD1} -function {ground VSS}

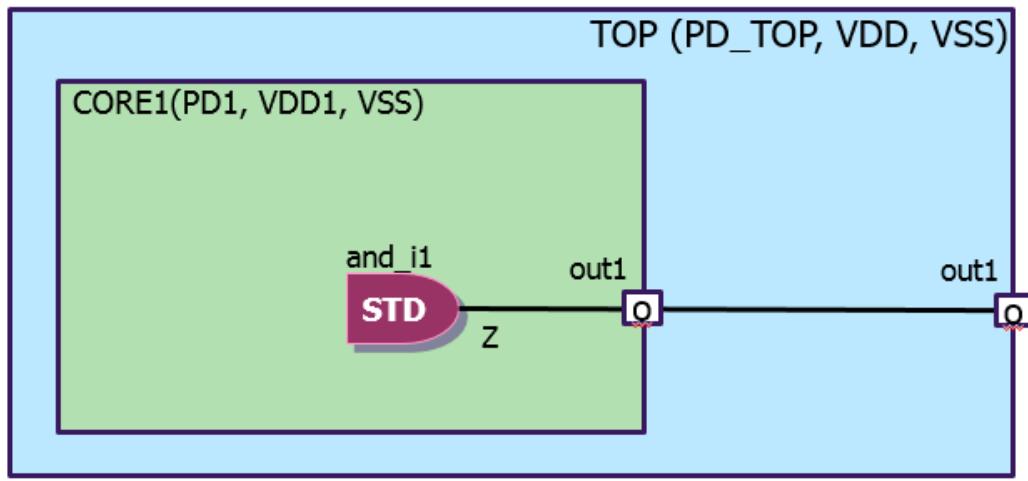
create_power_domain TOP -include_scope -supply {primary SS_TOP} -available_supplies
{SS1}
create_power_domain PD1 -elements {CORE1} -supply {primary SS1} -available_supplies {}

set_level_shifter PD1_outputs -domain PD1 -applies_to outputs -rule low_to_high -
location self
```

As shown in [Figure 5-116](#), the level shifter cell will require supplies VDD and VDD1. Out of which only VDD1 is available in PD1 (that is, -location self). Therefore, VC LP reports violation LS\_SUPPLY\_UNAVAIL for supply VDD:

Tag	:	LS_SUPPLY_UNAVAIL
Description	:	Level shifter supply net [UPFSupply] for strategy [Strategy] is domain dependent but not available in domain [Domain]
UPFSupply	:	VDD
Strategy	:	PD1/ls_PD1_outputs
Domain	:	PD1
Source	:	
PinName	:	CORE1/and_i1/Z
Sink	:	out1
LogicSource	:	
PinName	:	CORE1/and_i1/Z
LogicSink	:	out1

**Figure 5-116 LS\_SUPPLY\_UNAVAIL**



	VDD	VDD1	VSS
s1	1.2V	1.2V	0.0V
s2	1.2V	1.0V	0.0V

### ❖ UPF\_REPEATERS\_UNAVAIL

The `-repeater_supply` attribute is a domain dependent supply, therefore for the domain port on which the `-repeater_supply` is provided, if the `-repeater_supply` attribute is not available, then `UPF_REPEATERS_UNAVAIL` is reported.

Consider the following UPF is defined for the example shown in [Figure 5-117](#), where the repeater supply VDD is not defined for the power domain element CORE1.

```
create_supply_set SS_TOP -function {power VDD} -function {ground VSS}
create_supply_set SS1 -function {power VDD1} -function {ground VSS}
Create_supply_set SS_dummy

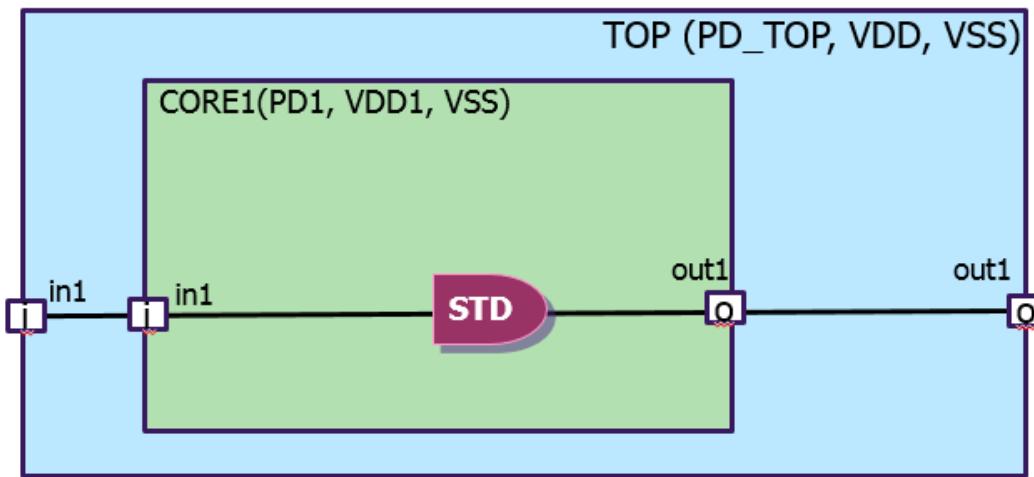
create_power_domain TOP -include_scope -supply {primary SS_TOP}
create_power_domain PD1 -elements {CORE1} -supply {primary SS1} -available_supplies
{SS_dummy}

set_port_attributes -ports {CORE1/in1 CORE1/out1} -repeater_supply SS_TOP
```

As shown in [Figure 5-117](#), the repeater for CORE1/out which is to be placed inside CORE1 requires VDD supply, however it is not available in the UPF for PD1 power domain. Therefore, the following violation is reported for this scenario:

```
Tag      : UPF_REPEATERS_UNAVAIL
Description : Supply net [UPFSupply] in repeater_supply [CellPin] of instance
[Instance] ([Cell]) is domain dependent but not available in domain [Domain]
    UPFSupply   : VDD
    CellPin     : out1
    Instance    : CORE1
    Cell        : core1
    Domain      : PD1
```

**Figure 5-117 UPF\_REPEATERS\_UNAVAIL**



#### 5.3.50.2 Limitation

VC LP should report an error parser message when hierarchical `supply_sets/supply_set_handles` are used in the `-available_supplies` option of a `create_power_domain` command. However, VC LP does not report any parser error message, and continues to compile the UPF.

### 5.3.51 Support for `create_power_domain -elements { . }`

Previously, to include the top instance of the current scope in a power domain, you had to use the option `-include_scope` in the `create_power_domain` command. However, `-include_scope` is deprecated in LRM. To make VC LP compatible with UPF 3.0, you can use the `create_power_domain -elements { . }` instead of the `-include_scope` option to include the top instance of the current scope in the power domain.

#### Example

The following two commands create power domain on the top level module of the design.

```
create_power_domain PD -include_scope
create_power_domain PD -elements { . }
```



**Note**  
VC LP does not report a deprecated Warning for `-include_scope`. VC LP also supports `-include_scope` along with `-elements { . }`.

### 5.3.52 Support for `-resolve` Function in `create_supply_net`

VC LP supports the following resolution methods for the `create_supply_net` UPF command:

- ❖ `unresolved`
- ❖ `parallel`
- ❖ `one_hot`

A resolution method determines the state and voltage of the supply net when the net has multiple supply sources.

To enable compatibility with UPF3.0, VC LP supports custom resolution functions. You can write custom functions in the `create_supply_net -resolve` command.

```
create_supply_net net_name
[-domain domain_name] [-reuse]
[-resolve <unresolved | one_hot | parallel | parallel_one_hot |resolution_
function_name >]
```

#### For example

```
create_supply_net VDD -resolve xyz
```

#### 5.3.52.1 Parser Messages Introduced

Consider a supply net is specified with one resolution function and if it is updated with the `-reuse` option, then VC LP reports the `[Error] UPF_SNET_REUSE_RESOLUTION` parsing message.



If you proceed despite the parser error, with the `all_golden_upf_error_proceed` application variable, then the `UPF_SUPPLY_RESOLUTION` violation is reported.

For Example:

```
create_supply_net VDD -resolve strong
create_supply_net VDD -resolve parallel -reuse
```

#### 5.3.52.2 Violations Tags Introduced

The following violation tags are introduced for this support:

- ❖ `UPF_SUPPLY_RESOLUTION`

This violation is reported when the same net has different resolution functions which are in different scope, and they both are connected using connect\_supply\_net.

```
-----  
UPF_SUPPLY_RESOLUTION (1 error/0 waived)  
-----  
Tag : UPF_SUPPLY_RESOLUTION  
Description : Supply net [UPFSupply] resolution inconsistent related to its  
drivers/connected nets with resolutions  
Violation : LP:1  
UPFSupply : VDDA_switched  
DriverList  
PortName : psw/VDDV  
ResolutionList  
Resolution : parallel
```

The UPF\_SUPPLY\_RESOLUTION violation is reported when multiple supply drivers are driving the same supply net without resolution. You can provide an extended list of valid supply drivers using the lp\_macro\_list application variable. This application variable can take the following values:

*none, Cell name, \*, false*

- ❖ When the application variable is set to false, VC LP follows the default behavior.
- ❖ When the application variable is set to value: "\*" (asterix), all the macro cells are considered as valid drivers.
- ❖ When the application variable is set to "none", the following pins/ports are considered as valid drivers.

Cell Type	Port Type/PG Type	Direction
Power switch cell	primary_power/ground backup_power/ground internal_power/ground	inout,output inout,output inout,output
Power switch strategy		output supply
UPF supply ports		input,inout
Design Primary ports		input,inout

- ❖ When the application variable is set to *Cell name* ("DMMACRO1 DMMACRO2"), in addition to the "none" behavior, the following macro cell pins are considered.

User Macro cell	primary_power/ground	inout,output
	backup_power/ground	inout,output
	internal_power/ground	inout,output

Wild cards are supported when the variable is set to cell name. You can use the command in the following methods.

```
set_app_var lp_macro_list {LNMACRO*}  
set macro_list [get_lib_cells */LNMACRO]  
append_to_collection macro_list [get_lib_cells */NLNMACRO_?]  
set_app_var lp_macro_list $macro_list
```

### 5.3.53 Support for enable\_bias Design Attribute

VC LP has been performing bias related checks irrespective of the `enable_bias` design attribute value.

The other SNPS tools require this design attribute set to be true/ derived. Therefore, the UPF with bias nets but without this attribute (or attribute set to false) reports syntax errors in other Synopsys tools while it works fine in VC LP.

The following is an example definition in the UPF with the `enable_bias` design attribute:

```
set_design_attributes -elements . -attribute enable_bias true
```

To indicate this incompatibility, VC LP has introduced the `COMPAT_BIASATTR_MISSING` violation. This violation is disabled by default, and it is reported during the `check_lp -stage UPF`.

This violation is reported when the UPF has bias nets defined, but do not use the `enable_bias` attribute or the `enable_bias` attribute is explicitly set to false. If the `enable_bias` attribute is set to true/derived, VC LP does not report the `COMPAT_BIASATTR_MISSING` violation.

Note that irrespective of the `enable_bias` attribute set to true/false/derived, VC LP performs all the bias related checks. The only change is in flagging this new violation.

#### UPF Snippet

```
set_design_attributes -elements { . } -attribute enable_bias false
...
...
create_supply_set SS_TOP -function {ground VSS} -function {power VDD} -function { nwell
VDD1
```

`COMPAT_BIASATTR_MISSING (1 warning/0 waived)`

```
-----  
Tag : COMPAT_BIASATTR_MISSING  
Description : Design uses bias nets but attribute enable_bias not set  
Violation : LP:1
```



#### Note

VC LP allows only `".` as the value of `-elements` while setting the design attribute `"enable_bias"`. VC LP does not consider setting of this attribute hierarchically. So, it is recommended to set this attribute in the top level UPF only.

### 5.3.54 Support for -elements Option in the `set_design_attribute -attribute correlated_supply_group` Command

Previously, VC LP was allowing `".` in the `-element` option of the `set_design_attribute -attribute correlated_supply_group` command. If an instance name was given in the `-elements` option of `set_design_attribute -attribute correlated_supply_group` command, then VC LP was reporting the following parser message:

`[Warning] UPF_ATTR_INVALID_ELEMENT_VALUE.`

The other Synopsys tools allows instance name in the `-element` option of the `set_design_attribute -attribute correlated_supply_group` command. VC LP supports instance name in the `-element` option of the `set_design_attribute -attribute correlated_supply_group` command and honors it.

#### Example

Consider the following UPF snippet:

```
set_design_attributes -elements {CORE1} -attribute correlated_supply_group "*"
```

Previously, VC LP was reporting the following warning message:

*Project\_6000021386\_sc1.upf:43: [Warning] UPF\_ATTR\_INVALID\_ELEMENT\_VALUE: Invalid element value for attribute*

*Element 'top/CORE1' specified with '-elements' of 'set\_design\_attributes' cannot be specified for attribute 'correlated\_supply\_group'. The only valid value is '!'.*

VC LP accepts the instance name in the `-elements` option of the `set_design_attribute -attribute correlated_supply_group` command, and does not report any error or warning message when loading the UPF.

Notes:

- ❖ Currently if a flat UPF instance is used, then VC LP does not report any error/warning.
- ❖ If the block scope level supplies have `correlated_supply_group` attribute in its own scope, and this attribute for the same supplies is missing at the top scope where block level supplies are connected, then VC LP should report an error, but currently VC LP does not report an error for this case.

### 5.3.55 Support for ignore\_voltage\_difference User Defined Attribute

The retention cells with internal level shifters can operate with different voltages at primary and backup supplies. In such cases, the RET\_INST\_VOLTAGE violations reported are considered as noise for these retention cells.

The `ignore_voltage_difference` boolean user attribute is introduced, which can be defined at cell scope (default false). When the retention cells are defined with the `ignore_voltage_difference` attribute, the RET\_INST\_VOLTAGE check is not performed on such cells.

The `ignore_voltage_difference` attribute can be set to a cell using the following Tcl command, and as custom attribute in liberty

#### Examples

- ❖ Setting up attribute using TCL command
 

```
set_attribute [get_lib_cells tiny/RETFF] ignore_voltage_difference true
```
- ❖ Setting the attribute as a custom liberty attribute

```
library (library) {
  define(ignore_voltage_difference,cell,boolean);
  cell (RETFF) {
    ignore_voltage_difference true
  }
}
```



#### Note

The `ignore_voltage_difference` attribute is not supported in Library Compiler.

### 5.3.56 Support for sim\_corruption\_control Command

VC LP supports the `sim_corruption_control` UPF 3.1 command.

#### Syntax

```
sim_corruption_control
  -type <real | integer | seq | comb | port | process >
```

```
[-elements element_list ]
[-exclude_elements exclude_list]
[-model model_name]
[-domain domain_name]
[-transitive [<TRUE | FALSE>]]
```

Where:

- ❖ The default value of `-transitive` is TRUE.
- ❖ The `-type` option is a mandatory option, and can take one of real, integer, seq, comb, port, process arguments.
- ❖ The `-elements`, `-exclude_elements`, `-model` and `-domain` options accept any value.

VC LP parses and ignores the `sim_corruption_control` commands with the following info message:

*[Info] UPF\_COMMAND\_IGNORED: Upf command ignored  
The command 'sim\_corruption\_control' will be ignored.*

VC LP performs the following syntax checks on the `sim_corruption_control` command.

- ❖ If the command does not have valid set of options, VC LP reports the `TCL_OPTION_UNKNOWN` error message.

#### Example

```
sim_corruption_control -direction in -type port
```

*FileName:LineNumber: [Error] TCL\_OPTION\_UNKNOWN: Option not known 'Unknown option '-direction'*

*FileName:LineNumber: [Error] TCL\_OPTION\_EXTRA: Extra positional option specified 'Extra positional option 'in''*

- ❖ The `-type` option is mandatory, if the `-type` option is not specified, the following error message is reported:

*FileName:LineNumber: [Error] TCL\_OPTION\_MANDATORY: Mandatory option not found 'Required argument '-type' was not found'*

- ❖ If the command options do not have valid arguments, VC LP reports the `TCL_OPTVAL_INVALIDONEOF` error message.

#### Example

- ◆ `sim_corruption_control -type user_given`

*FileName:LineNumber: [Error] TCL\_OPTVAL\_INVALIDONEOF: Invalid one-of value specified 'Value 'user\_given' for option '-type' is not valid. Specify one of: real, integer, seq, comb, port, process'*

- ◆ `sim_corruption_control -transitive user_given -type port`

*FileName:LineNumber: [Error] TCL\_OPTVAL\_INVALIDONEOF: Invalid one-of value specified 'Value 'user\_given' for option '-transitive' is not valid. Specify one of: TRUE, FALSE'*

### 5.3.57 Support for `sim_assertion_control` Command

VC LP supports the `sim_assertion_control` UPF 3.1 command.

#### Syntax

```
sim_assertion_control
[-elements element_list ]
```

```

[-exclude_elements exclude_list]
[-domain domain_name]
[-model model_name]
[-controlling_domain domain | -control_expr boolean_expression]
[-type <reset | suspend | kill>]
[-transitive [<TRUE | FALSE>]]

```

Where:

- ❖ The default value of -transitive is TRUE.
- ❖ All options are optional options.
- ❖ The -elements, -exclude\_elements, -model, -domain, -controlling\_domain and -control\_expr options accept any value.

VC LP parses and ignores the sim\_assertion\_control commands with following info message:

*\* Ignored with an info message*

*[Info] UPF\_COMMAND\_IGNORED: Upf command ignored*

*The command 'sim\_assertion\_control' will be ignored.*

VC LP performs the following syntax checks on the sim\_assertion\_control command.

- ❖ If the command does not have valid set of options, VC LP reports the TCL\_OPTION\_UNKNOWN error message.

### Example

```
sim_assertion_control -direction in
```

*FileName:LineNumber: [Error] TCL\_OPTION\_UNKNOWN: Option not known 'Unknown option '-direction''*

*FileName:LineNumber: [Error] TCL\_OPTION\_EXTRA: Extra positional option specified 'Extra positional option 'in''*

- ❖ If the command options do not have valid arguments, VC LP reports the TCL\_OPTVAL\_INVALIDONEOF error message.

### Example

- ◆ sim\_assertion\_control -type unknown

*FileName:LineNumber: [Error] TCL\_OPTVAL\_INVALIDONEOF: Invalid one-of value specified 'Value 'user\_given' for option '-type' is not valid. Specify one of: reset, suspend, kill'*

- ◆ sim\_assertion\_control -transitive user\_given

*FileName:LineNumber: [Error] TCL\_OPTVAL\_INVALIDONEOF: Invalid one-of value specified 'Value 'user\_given' for option '-transitive' is not valid. Specify one of: TRUE, FALSE'*

- ❖ If you specify both the -controlling\_domain and -control\_expr option, the TCL\_OPT\_ATMOST\_ONE\_OF error message is reported.

```
sim_assertion_control -model modA -controlling_domain PD1 -control_expr {
(interval(clk_dyn posedge negedge) >= 100ns) }
```

*FileName:LineNumber: [Error] TCL\_OPT\_ATMOST\_ONE\_OF: Incorrect command options  
It is not legal to specify more than one of '(-controlling\_domain, -control\_expr)' in command  
'sim\_assertion\_control'.*

### 5.3.58 Support for sim\_replay\_control Command

VC LP supports the `sim_replay_control` UPF 3.1 command.

#### Syntax

```
sim_replay_control
[-elements element_list ]
[-exclude_elements exclude_list]
[-domain domain_name]
[-model model_name]
[-controlling_domain domain]
[-transitive [<TRUE | FALSE>]]
```

Where:

- ❖ The default value of `-transitive` is TRUE.
- ❖ All options are optional options.
- ❖ The `-elements`, `-exclude_elements`, `-model`, `-domain`, `-controlling_domain` options accept any value.

VC LP parses and ignore the `sim_replay_control` commands with following info message:

*\* Ignored with an info message*

*[Info] UPF\_COMMAND\_IGNORED: Upf command ignored*

*The command 'sim\_replay\_control' will be ignored.*

VC LP performs the following syntax checks on the `sim_replay_control` command.

- ❖ If the command does not have valid set of options, VC LP reports the `TCL_OPTION_UNKNOWN` error message.

#### Example

```
sim_replay_control -direction in
```

*FileName:LineNumber: [Error] TCL\_OPTION\_UNKNOWN: Option not known 'Unknown option '-direction'*

*FileName:LineNumber: [Error] TCL\_OPTION\_EXTRA: Extra positional option specified 'Extra positional option 'in''*

- ❖ If the command options do not have valid arguments, VC LP reports the `TCL_OPTVAL_INVALIDONEOF` error message.

#### Example

```
♦ sim_replay_control -transitive user_given
```

*FileName:LineNumber: [Error] TCL\_OPTVAL\_INVALIDONEOF: Invalid one-of value specified 'Value 'user\_given' for option '-transitive' is not valid. Specify one of: TRUE, FALSE'*

### 5.3.59 Support for Compatibility with Other Synopsys Implementation Tools

The following violations are supported in VC LP.

- ❖ `COMPAT_SUPPLYNET_SCOPE`
- ❖ `COMPAT_SUPPLYSET_SCOPE`

These checks are added to make VC LP compatible with the Synopsys implementation tools. In the Synopsys implementation tools, error messages are reported when the scoped level supply net/supply set is not available in the Top scope UPF, but they are used in the UPF commands.

In Synopsys implementation tools, the supply set must be available in the same scope or upper scope in which the UPF command domain is created.

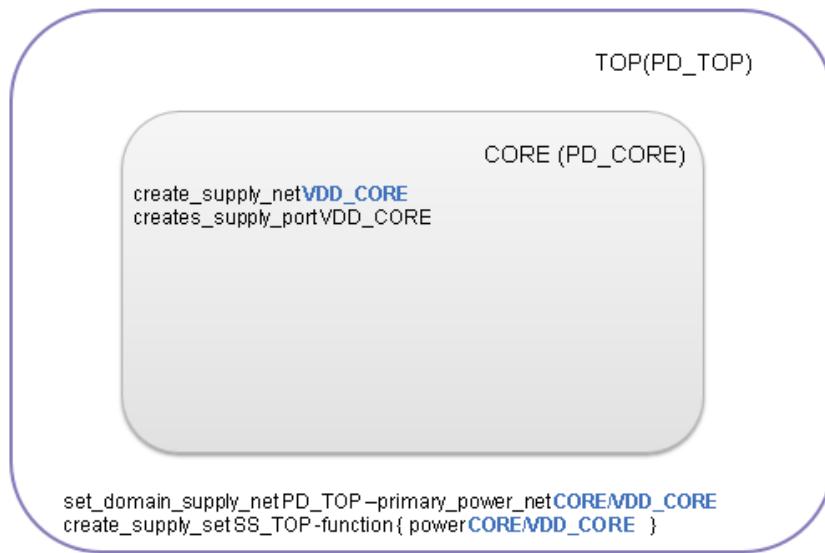
For designs where Synopsys implementation tools are not used, this message may be safely ignored. When this message appears, the tool accepts and uses the supply set definition, but you need to change the UPF before running Synopsys implementation tools.

### 5.3.59.1 COMPAT\_SUPPLYNET\_SCOPE

The COMPAT\_SUPPLYNET\_SCOPE violation is reported when the supply net is not available in the same scope or upper scope in which the UPF power domain is created.

As shown [Figure 5-118](#), the supply net VDD\_CORE is created/available in the scope CORE. This Supply net is not available in TOP scope. However, the supply net is used in TOP scope (set\_domain\_supply\_net /create\_supply\_set). Therefore, VC LP reports the COMPAT\_SUPPLYNET\_SCOPE violation.

**Figure 5-118 COMPAT\_SUPPLYNET\_SCOPE**



### 5.3.59.2 COMPAT\_SUPPLYSET\_SCOPE

The COMPAT\_SUPPLYSET\_SCOPE violation is reported when the supply set is not available in the same scope or upper scope in which supply set or the UPF power domain is created.

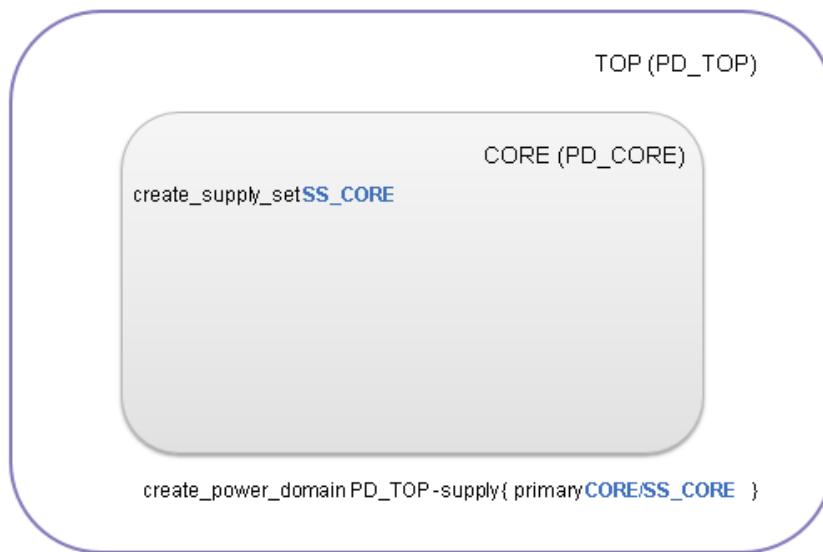
If the scope level supply set is used in the following commands and supply set is not available, VC LP reports the COMPAT\_SUPPLYSET\_SCOPE violation.

- ❖ create\_power\_domain - supply, available\_supplies
- ❖ set\_retention - retention\_supply\_set
- ❖ set\_isolation - isolation\_supply\_set
- ❖ set\_level\_shifter - input\_supply\_set, output\_supply\_set

As shown [Figure 5-119](#), the supply set SS\_CORE is created/available in the scope CORE. This supply set is not available in TOP (PD\_TOP) scope. However, this supply set is used in TOP scope command (for

example, `create_power_domain -supply`). Therefore, VC LP reports the COMPAT\_SUPPLYSET\_SCOPE violation.

**Figure 5-119 COMPAT\_SUPPLYSET\_SCOPE**



### 5.3.60 Support for Syntax and Semantics Checks for VCT commands

When a supply net in UPF is connected to a HDL port which is not of supply\_net\_type\* (SV struct or VHDL record), a value conversion must be performed. When the HDL port drives the supply net, the HDL data type value must be converted to UPF supply net type. When the supply net drives the HDL port, the value of the UPF Supply Net must be converted to a value of the HDL data type of the HDL port. VCT is a value conversion table that converts a value in HDL (verilog or VHDL) to a value in UPF or a value in UPF to a value in HDL.

The following two commands can be used to create the conversion table

- ❖ `create_upf2hdl_vct`

This command defines a VCT for the supply\_net\_type.state value when that value is propagated from a UPF supply net into a logic port defined in an HDL. It provides a 1:1 conversion for each possible combination of the partially on and on/off states.

**Syntax**

```
create_upf2hdl_vct vct_name -hdl_type {<vhdl | sv> [typename]} -table {{from_value
to_value}*}
```

- ❖ `create_hdl2upf_vct`

This command defines a VCT from an HDL logic type to the state type of the supply net value (see Annex B) when that value is propagated from HDL port to a UPF supply net. It provides a conversion for each possible logic value that the HDL port can have.

**Syntax**

```
create_hdl2upf_vct vct_name -hdl_type {<vhdl | sv> [typename]} -table {{from_value
to_value}*}
```

Prior to this release, VC LP was just parsing and ignoring the above two VCT commands. VC LP performs syntax checks, and reports the following parsing messages:

- ❖ UPF\_VCT\_NAME\_NOT\_UNIQUE
- ❖ UPF\_MISMATCH\_VCT\_HDL\_TYPE
- ❖ UPF\_INVALID\_VCT\_HDL\_TYPE
- ❖ TCL\_OPT\_INVALID\_LIST
- ❖ UPF\_INVALID\_VCT\_SUPPLY\_STATE
- ❖ UPF\_INVALID\_VCT\_HDL\_VALUE
- ❖ UPF\_INVALID\_VCT\_TABLE
- ❖ UPF\_INVALID\_ID

## Examples

- ❖ `create_upf2hdl_vct UPF_GNDZERO2SV_LOGIC -hdl_type {sv reg} -table {{OFF 0} {FULL_ON 1} {PARTIAL_ON Z}}`

### Parser Message

*test.upf:47: [Error] UPF\_VCT\_NAME\_NOT\_UNIQUE: VCT name is not unique  
 'UPF\_GNDZERO2SV\_LOGIC' is already used as vct name in UPF or a default vct.  
 Vct names must be unique in the global design scope.*

- ❖ `create_hdl2upf_vct stdlogic2upf_vss1 -hdl_type {sv Logic} -table {{0 FULL_ON} {H OFF} {Z PARTIAL_ON}}`

### Parser Message

*test.upf.upf:48: [Error] UPF\_MISMATCH\_VCT\_HDL\_TYPE: Hdl type is mismatch  
 'sv' is used with '-hdl\_type' option of create\_hdl2upf\_vct, but the table value is H.  
 The table is defined for vhdl supply states but -hdl\_type is sv.*

- ❖ `create_hdl2upf_vct stdlogic2upf_vss2 -hdl_type {svv Logic} -table {{1 FULL_ON} {0 OFF} {Z PARTIAL_ON}}`

### Parser Message

*test.upf.upf:49: [Error] UPF\_INVALID\_VCT\_HDL\_TYPE: Incorrect Hdl type is specified  
 'svv' is used with '-hdl\_type' option of create\_hdl2upf\_vct.  
 Valid hdl types with -hdl\_type option are 'sv' and 'vhdl'.*

- ❖ `create_hdl2upf_vct stdlogic2upf_vss3 -hdl_type {sv Logic} -table {{FULL_ON} {0 OFF} {Z PARTIAL_ON}}`

### Parser Message

*test.upf.upf:50: [Error] TCL\_OPT\_INVALID\_LIST: Invalid string list value specified  
 Invalid string list '{FULL\_ON}' specified with command option 'create\_hdl2upf\_vct -table'.  
 Please specify a valid string value.*

- ❖ `create_hdl2upf_vct stdlogic2upf_vss4 -hdl_type {sv Logic} -table {{1 FULL_ONN} {0 OFF} {Z PARTIAL_ON}}`

### Parser Message

*test.upf:51: [Error] UPF\_INVALID\_VCT\_SUPPLY\_STATE: Invalid upf Supply State is specified  
 FULL\_0NN is not a correct upf supply state.*

*The upf Supply State used with '-table' option of create\_hdl2upf\_vct is not valid.*

- ❖ `create_hdl2upf_vct stdlogic2upf_vss5 -hdl_type {sv Logic} -table {{0 FULL_ON} {1 OFF} {P PARTIAL_ON}}`

#### Parser Message

*test.upf:52: [Error] UPF\_INVALID\_VCT\_HDL\_VALUE: Invalid Hdl value is given*

*The hdl value P used with '-table' option of create\_hdl2upf\_vct is not valid.*

*Please provide a valid hdl value.*

- ❖ `create_hdl2upf_vct stdlogic2upf_vss6 -hdl_type {sv Logic} -table {{0 FULL_ON} {0 OFF} {Z PARTIAL_ON}}`

#### Parser Message

*test.upf:53: [Error] UPF\_INVALID\_VCT\_TABLE: Incorrect VCT Table has been defined*

*Multiple entries found for value '0' in '-table' option of create\_hdl2upf\_vct.*

*Please provide a correct mapping.*

- ❖ `create_hdl2upf_vct 18 -hdl_type {sv logic} -table {{0 OFF} {1 FULL_ON} {X PARTIAL_ON} {Z UNDETERMINED}}`

#### Parser Message

*test.upf:65: [Error] UPF\_INVALID\_ID: Invalid UPF Identifier*

*Identifier '18' in command 'create\_hdl2upf\_vct' is not properly named. As per UPF, the first character of an identifier must be an alphabet. All other characters of a name shall be alphanumeric or the underscore character (\_).*

*Please specify a valid UPF Identifier.*

### 5.3.60.1 Limitations:

- ❖ VC LP is not considering type name mentioned in -hdl\_type switch for parsing checks.
  - ◆ `create_upf2hdl_vct vct_name -hdl_type {<vhdl | sv> [typename]} -table {{from_value to_value}*}`
  - ◆ VC LP is not reporting errors for the following commands:
    - ◆ `create_hdl2upf_vct vct_1 -hdl_type {vhdl bit} -table {{0 OFF} {1 FULL_ON} {X PARTIAL_ON} {Z UNDETERMINED}}`
    - ◆ `create_hdl2upf_vct vct_12 -hdl_type {vhdl bt} -table {{0 OFF} {1 FULL_ON} {X PARTIAL_ON} {Z UNDETERMINED}}`
- ❖ VC LP is not checking hdl\_type values.
  - ◆ For example, bit data type is a two state value, but here X and Z hdl values are given, and VC LP does not report error message for these hdl values. (Only 0 and 1 hdl values are supported)
 

```
create_hdl2upf_vct vct_12 -hdl_type {vhdl Bit} -table {{0 OFF} {1 FULL_ON} {X PARTIAL_ON} {Z UNDETERMINED}}
```
- ❖ Design attribute feature does not work with vct UPF commands.
- ❖ PG pin port direction interface does not work when it drives a supply net or is driven by a supply net.
- ❖ The connect-supply\_net -vct UPF command is not supported.

### 5.3.61 Support for Name Space Conflict between UPF and Design Logic Net/Port

The `create_logic_port`/`create_logic_net` commands may introduce conflicts if the design objects with the same names are already present in the netlist.

Before synthesis, the logic port/net mentioned in the UPF does not exist in the RTL design. During synthesis, the tool instruments the logic ports/nets as per these commands. As these names exist in the instrumented gate level netlist, using the original UPF on the gate level netlist must not result in any error, and the run must proceed with analysis.

VCLP reports a warning message in such cases and enable reapplication of the same UPF after synthesis.

VC LP reports the `UPF_OBJECT_FOUND_WARN` warning message:

- ❖ When `create_logic_port` is used with an existing design port
- ❖ When `create_logic_net` is used with an existing design net

VC LP silently accepts and does not report any warning message:

- ❖ When `create_supply_port` is used with existing design port
- ❖ When `create_supply_net` is used with existing design net

#### 5.3.61.1 Example

Consider the following design:

```
module top (in1, in2, iso_en, clk, reset, out1, out2);
  input in1, in2, iso_en, clk, reset;
  output out1, out2;
endmodule
```

And The following UPF snippet:

```
create_logic_port iso_en
create_logic_net iso_en
```

The following parser warning messages are reported for this example:

- ❖ *case1.upf:13: [Warning] UPF\_OBJECT\_FOUND\_WARN: Object already exists  
Object 'iso\_en' of type 'Port' is already present at case1.v,1. Cannot create UpfLogicPort with the same name  
in command 'create\_logic\_port'.  
Please specify a name which is not already created.*
- ❖ *case1.upf:13: [Warning] UPF\_CMD\_IGNORED: Ignoring erroneous command  
Ignoring command 'create\_logic\_port' since it has errors.  
Please correct the errors by looking at the log.*
- ❖ *case1.upf:14: [Warning] UPF\_OBJECT\_FOUND\_WARN: Object already exists  
Object 'iso\_en' of type 'Net' is already present at case1.v,1. Cannot create UpfLogicNet with the same name in  
command 'create\_logic\_net'.  
Please specify a name which is not already created.*
- ❖ *case1.upf:14: [Warning] UPF\_CMD\_IGNORED: Ignoring erroneous command  
Ignoring command 'create\_logic\_net' since it has errors.  
Please correct the errors by looking at the log.*

### 5.3.62 Support for `is_on_clock_path` Attribute

VC LP introduced a new methodology to determine whether a library cell (in netlist) or an inferred logic (in rtl) is present on a clock path or not. A new attribute `is_on_clock_path` is added to each cell in the clock

path during infer\_source. With attribute `is_on_clock_path`, VC LP detects cells (like MUX & logic gates), blackbox feedthrough cells, macro feedthrough cells, lib cells, protection cells (ISO/LS/ELS) and so on in the clock paths. Using this attribute, you can write Tcl scripts to find all the cells that are on the clock path.

### Example

The following Tcl script lists all the cells on the clock paths:

```
set gc [get_cells -hier *]
foreach i_gc $gc {
    puts [get_attribute $i_gc "is_on_clock_path"]
```

### Impact on existing support

The `report_protection_cells_on_clock_path` application variable provides a similar support but only for low power protection cells. It is recommended to use the new attribute `is_on_clock_path` instead of the `report_protection_cells_on_clock_path` application variable. Hence, the following message is reported when the application variable `report_protection_cells_on_clock_path` is set to true.

*<message>*

*Info: LP\_CMD\_DEPRECATED*

### 5.3.63 Support for `describe_state_transition` and `add_state_transition` UPF Command

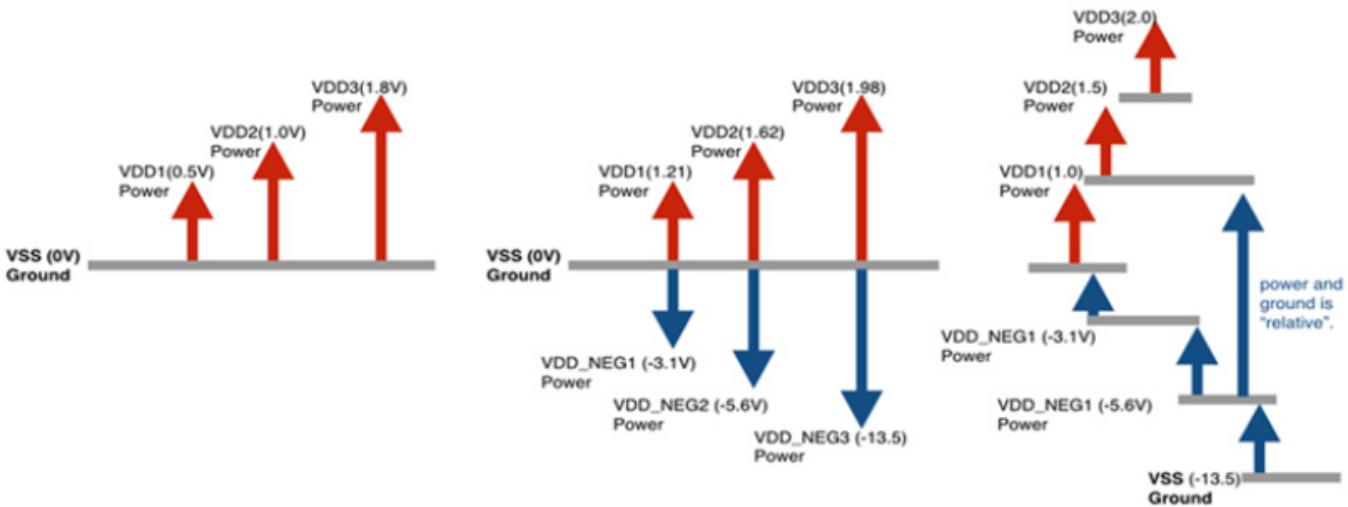
With UPF 3.0, the `describe_state_transition` command is replaced with `add_state_transition` command. VC LP parses and ignores both the `describe_state_transition` and `add_state_transition` UPF commands with the following informational messages:

*[Info] UPF\_COMMAND\_IGNORED: Upf command ignored  
The command 'describe\_state\_transition' will be ignored.*

*[Info] UPF\_COMMAND\_IGNORED: Upf command ignored  
The command 'add\_state\_transition' will be ignored.*

### 5.3.64 Support for Relative Power Ground Supply

VC LP supports a complex power scheme where supply can act as power in one section and act as ground in another section. For example, consider the node in the right-side in [Figure 5-120](#), where power and ground is 'relative'. A supply net can be used both as power and ground according to the voltage different with another supply net. This support is added under the `mv_upf_relative_pg_support` application variable. By default, this application variable is set to false.

**Figure 5-120 Power ground modeling**

The same PG pin can be used as 'related\_power\_pin' for one signal pin, and used as 'related\_ground\_pin' for another signal pin.

Relative PG pin is only in macro cells, from leafcell derived cell level boolean attribute 'use\_same\_pin\_for\_power\_and\_ground' for cell using 'relative' PG pin. VC LP queries the attribute to check if 'relative' PG pin is added in the cell. In UPF, both power supply net and ground supply net can have positive voltage value. However, in a supply set, voltage of power must be higher than voltage of ground.

A net will be marked as relative signal net if at least one pin on the net connected with relative signal pin. The net here is a flat net concept that it can cross hierarchical ports.

Relative supply nets or sets can only be used in the following UPF commands:

- ❖ create\_supply\_net
- ❖ connect\_supply\_net
- ❖ create\_supply\_set
- ❖ create\_power\_domain
- ❖ set\_domain\_supply\_net
- ❖ set\_port\_attributes -driver\_supply/receiver\_supply
- ❖ set\_related\_supply\_net
- ❖ add\_port\_state
- ❖ add\_power\_state
- ❖ associate\_supply\_set

#### 5.3.64.1 Parser Messages Introduced

- ❖ If a net without related\_pg\_pin is used as relative power ground is used as relative power-ground, the UPF\_SUPPLY\_NET\_BOTH\_PG parser error message is reported.
- ❖ If a strategy uses relative supply net or set with following usage, the UPF\_INVALID\_RELATIVE\_SUPPLY\_UASG parser error message is reported:
  - ◆ Isolation strategy uses relative supply as isolation supply, source and sink supply.

- ◆ Level shifter strategy uses relative supply as source and sink supply.
- ◆ Repeater strategy uses relative supply as repeater supply, source and sink supply.
- ◆ `create_power_switch` uses relative supply for `-input_supply_port/-output_supply_port` and `-supply_set`
- ◆ Retention strategy uses relative supply as retention supply.

### 5.3.64.2 Violations Introduced for this Support

The following violation message are introduced for this support:

#### ❖ RELSUPPLY\_NET\_INCORRECT

If some pins on the net are relative signal pin, and some are normal signal pin, VC LP reports the `RELSUPPLY_NET_INCORRECT` warning message, but the net will still be considered as a relative signal net.

#### ❖ RELSUPPLY\_DOMAIN\_INCORRECT

VC LP reports the `RELSUPPLY_DOMAIN_INCORRECT` violation in following cases.

- ◆ Power domain uses relative supply SupplySet as primary, but it contains cells without attribute `using_same_pg_pin_as_related_power_and_ground` as domain root element.
- ◆ Power domain uses relative supply set as primary, and domain is applied for non-leaf cell

#### ❖ RELSUPPLY\_VOLTAGE\_MISMATCH

If the power minimum voltage is higher than ground maximum voltage, VC LP reports the `RELSUPPLY_VOLTAGE_MISMATCH` violation with severity error.

### 5.3.64.3 Example

In the following example, in the supply set `SS__VDD_PHY__VSS`, VSS acts as ground and in the `SS__VDD_NEG__VSS`, VSS acts as power. Therefore, VSS acts as relative power- ground net, since this net is connected to I0/VL port, where the cell has derived `use_same_pin_for_power_and_ground` boolean attribute.

```

create_supply_port VDD_PHY -direction in
create_supply_port VDD_NEG -direction in
create_supply_port VSS -direction in
create_supply_net VDD_PHY -resolve parallel
create_supply_net VDD_NEG -resolve parallel
create_supply_net VSS -resolve parallel

connect_supply_net VDD_PHY -ports {VDD_PHY}
connect_supply_net VDD_NEG -ports {VDD_NEG}
connect_supply_net VSS -ports { VSS }

create_supply_set SS__VDD_PHY__VSS \
    -function { power VDD_PHY } \
create_supply_set SS__VDD_NEG__VSS \
    -function { ground VSS } \
        -function { power VSS } \
        -function { ground VDD_NEG }

create_power_domain PD_TOP \

```

```

-include_scope \
-supply { primary SS__VDD_PHY__VSS }
-supply { extra_supplies_1 SS_VDD_NEG_VSS }

connect_supply_net VDD_PHY -ports { I0/VH }
connect_supply_net VSS -ports { I0/VL VSS}
connect_supply_net VDD_NEG -ports { I1/VL }
connect_supply_net VSS -ports { I1/VH }

set_port_attributes -port OUT1 -driver_supply SS__VDD_PHY__VSS
set_port_attributes -port OUT2 -driver_supply SS__VDD_NEG__VSS

```

#### 5.3.64.4 Support for method\_power and method\_ground Attributes

VC LP has introduced the `method_power` and `method_ground` attributes for cell pins, ports, and crossover nodes. The output of `get_pins`, `get_ports` and `get_crossover_nodes` can be queried for it's supply method.

##### Example

```
get_attribute [get_crossover_nodes out] method_ground
[get_attribute [get_ports inst1/core1/in1] method_power]
```

Supported power/ground methods are "FROM\_PG\_NETLIST, FROM\_UPF\_CSN, FROM\_UPF\_POLICY, FROM\_UPF\_POWER\_DOMAIN, FROM\_UPF\_SRSN, FROM\_UPF\_DRIVER\_SUPPLY, FROM\_UPF\_RECEIVER\_SUPPLY, FROM\_UPF\_ISO\_SOURCE, FROM\_UPF\_ISO\_SINK, FROM\_UPF\_MAIN\_RAIL, FROM\_PG\_PIN, FROM\_DRIVER\_SUPPLY, FROM\_LOAD\_SUPPLY, FROM\_SET\_REPEATERS\_SUPPLY, FROM\_PAD, INVALID\_EXTRACTION\_TYPE"

#### 5.3.65 Support for set\_repeater -source /-sink

The `set_repeater` UPF command is enhanced with the `-source/-sink` options to give the user path level control because path based constraints are more specific and complete.

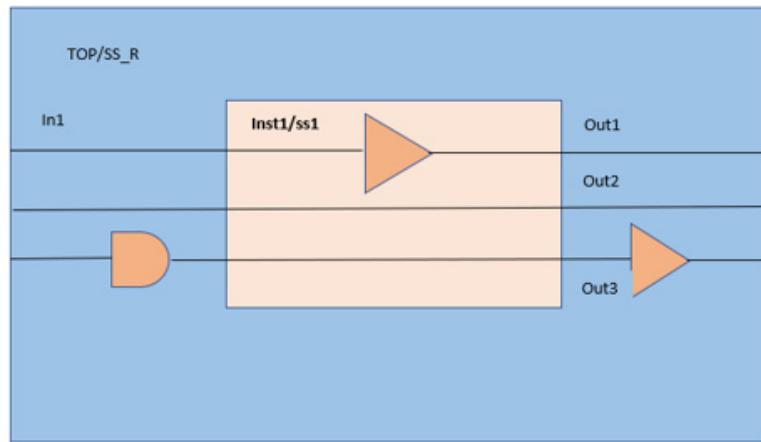
##### Syntax

```
set_repeater strategy_name
-domain domain_name
[-elements element_list]
[-exclude_elements exclude_list]
[-applies_to <inputs | outputs | both> ]
[-repeater_supply supply_set_ref ]
[-name_prefix string] [-name_suffix string]
[-source <source_domain_name | source_supply_ref >]
[-sink <sink_domain_name | sink_supply_ref >]
[-update]
```

- ❖ The `-source` option defines the rooted name of a supply set or power domain
- ❖ The `-sink` option defines the rooted name of a supply set or power domain

### 5.3.65.1 Matching Source/Sink Supplies

**Figure 5-121 Matching Source/Sink**



//Insert repeater only if sink is SS\_R

```
set_repeater R1 \
    -repeater_supply SS_2 \
    -domain TOP \
    -elements {in1} \
        -sink SS_R
```

//Insert repeater only if source add sink are SS\_R

```
set_repeater R2 \
    -repeater_supply SS_2 \
    -domain PD1 \
    -elements {inst1/out3} \
        -source SS_R \
        -sink SS_R
```

//Repeater not inserted as sink is not SS\_1

```
set_repeater R4 \
    -repeater_supply SS_2 \
    -domain PD1 \
    -elements {inst1/out1} \
        -sink SS_1
```

### 5.3.65.2 Precedence Rules for Policy Association

VC LP applies the repeater strategies on a pin based on the precedence rules. A power domain boundary port can have multiple applied repeater strategies that are intended for different paths or source/sink.

1. Pin or port-level strategy matching the -source and -sink options.
2. Pin or port-level strategy matching the -source option on a power domain input pin or matching the -sink option on a power domain output pin.
3. Pin or port level strategy matching the -sink option on a power domain input pin or matching the -source option on a power domain output pin.
4. Pin or ports specified with the -elements option, without specifying the -source or -sink option.

5. Cell-level strategy matching the -source and -sink options.
6. Cell-level strategy matching the -source option on a leaf input pin, or matching the -sink option on a leaf output pin.
7. Cell-level strategy matching the -sink option on a leaf input pin, or matching the -source option on a leaf output pin.
8. Cell-level strategy specified without using the -source or -sink options.
9. Domain-level strategy matching the -source and -sink options.
10. Domain-level strategy matching the -source option on an input pin or matching the -sink option on an output pin
11. Domain-level strategy matching the -sink option on an input pin or matching the -source option on an output pin.
12. Domain-level strategy specified without the -source or -sink options

 **Note**

The UPF\_STRATEGY\_RESOLVE violation with severity error is reported, if the precedence rules identify multiple strategies that apply to the same port.

### 5.3.65.3 Identification of Repeater buffer

Repeater buffers may be present in the design. Repeater will be matched topologically to a strategy at a port if the repeater has the same power supply as specified by the strategy.

Name based matching:

#### Example

```
set_repeater R1 -domain PD1 -repeater_supply SS_R -source ...
```

If repeater buffer name matches with "\*snps\_PD1\_\_R1\_snps\*", then it is a match.

### 5.3.65.4 Crossover Generation and Strategy Resolution

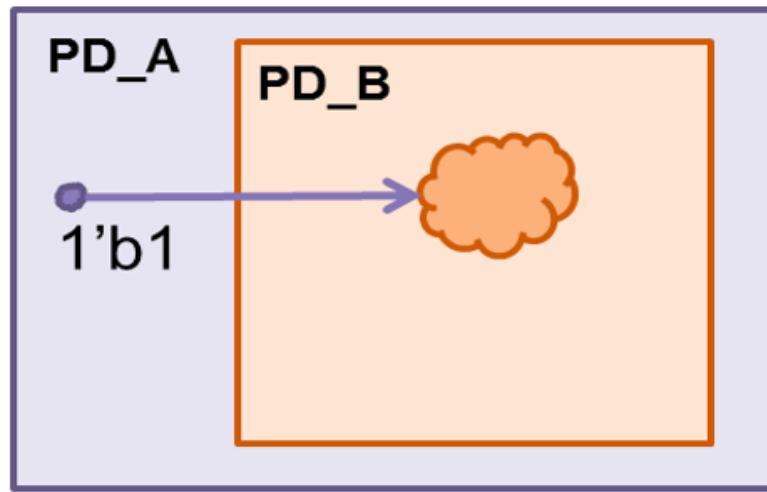
- ❖ Path-Based & Non Path-Based Repeater Strategies will be resolved based on the precedence rules defined above
- ❖ Crossovers break at the set\_repeater buffers virtual or real if present.
- ❖ Apply ISO/LS strategies by considering virtual/matched repeater as the source/sink boundary

### 5.3.65.5 Heterogeneous Sinks

For loads with heterogeneous supplies, the set\_repeater strategy cannot be applied as repeater buffer is always inserted inside the specified domain and it cannot satisfy Repeater needs for both paths. The SREP\_STRATEGY\_RESOLVE violation is reported for such scenarios.

### 5.3.66 Support for relax\_constant\_corruption\_for\_macro\_inputs\_and\_primary\_outputs Design Attribute

Implementation tool optimizes the path between literal constants (1'b0, 1'b1) to sink by moving the constant to sink domain and updating the root supply of constant to its sink supply. As a result, it can avoid redundant isolation or level shifter cell to be inserted in the path. Verification tools are expected to handle this in early stages and relax some checks.

**Figure 5-122 Literal constant crossing domain boundary**

To achieve this, the following design attribute is introduced:

```
relax_constant_corruption_for_macro_inputs_and_primary_outputs
```

When this attribute is set to true, VC LP relaxes the checks in the path from literal constant to sink.

In VC LP, this attribute mimics the behavior of the `lp_literal_constant -sink_all` command.

VC LP will relax the checks, when

- ❖ The crossover does not have isolation requirement
- ❖ Path has isolation requirement but
  - ◆ Isolation policy not provided or cannot be provided in the path
  - ◆ Isolation policy provided and has a clamp value matching the constant value

Use Model

```
set_design_attributes -elements {.} -attribute
relax_constant_corruption_for_macro_inputs_and_primary_outputs true
```

Note:

- ❖ This is by default FALSE attribute.
- ❖ This is an element base attribute, -model is not supported.
- ❖ This attribute must be specified on top scope (global attribute)
- ❖ For block analysis, this can be used in block upfs. When integrating the blocks together with the same block upfs, values should always match its top scope value.

Use the `lp_literal_constant -sink_internal` command to enable that the constant uses sink's related supply when sink is a macro pin with rpp/rgp is internal with direction internal.

Generally, the supply of the design constant is considered as the default supply of the power domain in which the literal constant is located. When this application variable is set to true, the supply of the design constant is considered as the supply of sink when `related_power_pin/related_ground_pin` of sink is `inter-internal_power/internal_ground` with direction internal.

### 5.3.67 Support for set\_level\_shifter input\_supply and output\_supply

Conventionally for VC LP checks, Supply driving the inputs of level shifter cell is derived from its driver supply and supply driving the outputs of the level shifter cell is derived from its load supply. Previously, VC LP had been providing only parser level support and minimal checker support for the set\_level\_shifter input supply/output supply command.

As per UPF LRM, you can provide input supply and output supply for Level shifter strategy. This allows user to have more control on deciding the supplies of the level shifters. Therefore, VC LP provides complete support for the set\_level\_shifter input supply/output supply command.

VC LP supports the following commands:

```
set_level_shifter <STRAT name> ..... -input_supply SS_LS_IN -output_supply SS_LS_OUT
set_level_shifter <STRAT name> ..... -input_supply_set SS_LS_IN -output_supply_set
SS_LS_OUT
```

With this support, there is a change in the supply resolution order for LS checks, and the new resolution order is

FROM\_PG\_NETLIST > FROM\_UPF\_CSN > FROM\_UPF\_POLICY (new for LS) > FROM\_UPF\_DERIVE (based on driver/load) > FROM\_POWER\_DOMAIN

#### 5.3.67.1 Impact of LS strategy supply:

The following table describes the resolving ISO, LS strategy supply for the root supply of LS/ELS cell data pins.

For netlist designs with strategy supply as the high precedence (i.e. No CSN available)

Cell (LS/ELS)	Input pin	Output pin	Enable pin
Dual rail LS	LS strategy input supply	LS strategy output supply	
Dual rail ELS (RPP, RGP of output and enable are same)	LS strategy input supply	ISO strategy supply	ISO strategy supply
Single rail LS (overdrive cell)		LS strategy output supply	
Single rail ELS (overdrive cell with RPP, RGP of output and enable are same)		ISO strategy supply	ISO strategy supply

#### 5.3.67.2 Violation Tags Impacted

The general LS tags with supply resolution information provides a new resolution method as "FROM\_UPF\_POLICY".

- ❖ The LS\_SUPPLY\_MISMATCH is reported with no changes.
- ❖ The LS\_SUPPLY\_UNAVAIL tag is enhanced to consider availability of strategy supply in the domain.

- ❖ The PG\_STRATEGY\_CONN is enhanced to check the PG to strategy supply inconsistency for LS cells
- ❖ LS\_SUPPLY\_STATE: Checks for the PST states with strategy input supply off and strategy output supply on.
- ❖ LS\_SUPPLY\_VOLTAGE: Checks for the PST states of the strategy supply that affect the rule of the strategy
- ❖ UPF\_ELSUPPLY\_MISMATCH: Checks for the mismatch between ISO supply and LS Strategy output supply

### 5.3.67.3 Impact on Application Variable

The handle\_aob\_ls\_no\_csn application variable saves the connect\_supply\_net (CSN) command for LS cell without pre-existing CSN/PG/SPA/SRSN. When this application variable is set:

- ❖ If LS strategy supply is not available, Tool dumps the CSN with derived supplies as per driver and load.
- ❖ If LS strategy supply is available, Tool dumps the CSN with strategy input, output supplies

### 5.3.67.4 Impact on Tcl Query commands

The following LS strategy related commands support strategy supplies

- ❖ report\_level\_shifter\_strategy
- ❖ get\_level\_shifter\_strategies
- ❖ get\_level\_shifter\_strategy\_elements
- ❖ The report\_trace\_paths and get\_trace\_paths commands populate the new resolution method, FROM\_UPF\_POLICY in their outputs

### 5.3.67.5 GUI impact

The change in supply resolution order of LS cell will have the effect on the property table of the violation schematic elements.

### 5.3.67.6 New Parser Messages Introduced

An error message is reported if -location other is applied to an upper boundary (lowconn) port of the design top module, or a lower boundary (highconn) of a leaf cell. However, if a LS is inserted for the command, a warning message is reported.

For regular cases which is not a macro, if -applies\_to\_boundary is both, a warning message is reported. If -applies\_to\_boundary is upper or lower, an error message is reported.

For soft macro/hard macro scenarios, an error message is reported if there is a LS inserted across the macro boundary.

#### Example

UPF	Parser Message
Top.upf: set_level_shifter ls2 -domain TOP -location other	Error - [UPF_STRATEGY_LOCATION_PWR_TOP_PARENT]

UPF	Parser Message
.Top.upf: set_level_shifter ls2 -domain TOP -location other - applies_to_boundary both	Warning - [UPF_STRATEGY_LOCATION_PWR_TOP_PARENT_WARN]
Top.upf: create_power_domain PD -elements {leaf} set_level_shifter ls3 -domain PD -location other - applies_to_boundary lower	Error - [UPF_STRATEGY_LOCATION_LOWER_BOUNDARY_LEAF]
Top.upf: create_power_domain PD -elements {leaf} set_level_shifter ls3 -domain PD -location other - applies_to_boundary both	Warning - [UPF_STRATEGY_LOCATION_LOWER_BOUNDARY_LEAF_WARN]
Macro.upf: set_design_attributes -models . -is_soft_macro TRUE create_power_domain PD_MODEL1 -elements {} set_level_shifter ls -domain PD_MODEL1 -location other -applies_to upper	Error - [UPF_STRATEGY_MACRO_WRONG_LOCATION]
Macro.upf: set_design_attributes -models . -is_soft_macro TRUE create_power_domain PD_MODEL1 -elements {} set_level_shifter ls -domain PD_MODEL1 -location other -applies_to both	Error - [UPF_STRATEGY_MACRO_WRONG_LOCATION]
Top.upf: create_power_domain TOP create_power_domain PD1 -elements {macro_inst} set_design_attributes -models macro -is_soft_macro TRUE set_level_shifter ls -domain TOP -location other - applies_to_boundary lower	Error - [UPF_STRATEGY_MACRO_WRONG_LOCATION]
Top.upf: create_power_domain TOP create_power_domain PD1 -elements {macro_inst} set_design_attributes -models macro -is_soft_macro TRUE set_level_shifter ls -domain TOP -location other - applies_to_boundary both	Error - [UPF_STRATEGY_MACRO_WRONG_LOCATION]

### 5.3.68 Support for -location other in LS strategy in UPF 3.0

Previously, VC LP had been providing support for value *self*, *parent*, and *fanout* in the -location field of `set_level_shifter` command. Starting with this release, VC LP supports value *other* in the -location field.

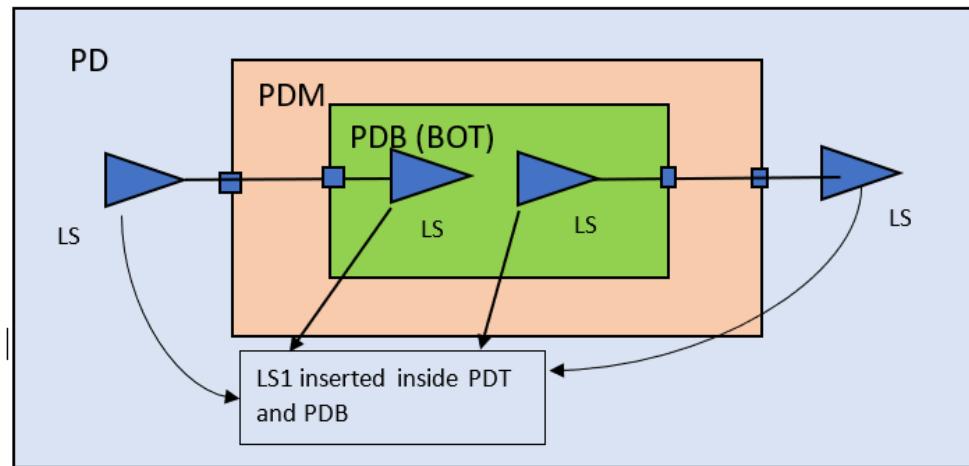
- ❖ The -location *other* in `set_level_shifter` allows users to have flexible specification.
- ❖ Level shifters can be placed in the parent domain for upper boundary port.

- ❖ Level shifters can be placed in the child domain for the lower boundary port.

### Example

```
create_power_domain PDT
create_power_domain PDM -element {MID}
create_power_domain PDB -elemnet {MID/BOT}
set_level_shifter LS1 -domain PDM -applies_to_boundary both -applies_to both -location other
```

Strategy LS1 would be inserted as follows:



The following parser checks are introduced for this support:

Error Message	Description	Example
<i>Error - [UPF_STRATEGY_LOCATION_N_PWR_TOP_PARENT]</i>	Location other defined for top level domain	<i>Top.upf:</i> set_level_shifter ls2 -domain TOP -location other
<i>Warning - [UPF_STRATEGY_LOCATION_N_PWR_TOP_PARENT_WARN]</i>	Location other applies to both boundaries for top domain gives a warning as LS insertion for lower domain boundary is possible.	<i>Top.upf:</i> set_level_shifter ls2 -domain TOP -location other - applies_to_boundary both
<i>Error - [UPF_STRATEGY_LOCATION_LOWER_BOUNDARY_LEAVE]</i>	Gives ERROR when location other defined for lower domain boundary of leaf level cell as LS insertion is not possible	<i>Top.upf:</i> create_power_domain PD -elements{leaf} set_level_shifter ls3 -domain PD -location other - applies_to_boundary lower

Error Message	Description	Example
<i>Warning -</i> [UPF_STRATEGY_LOCATION_LOWER_BOUNDARY_LEAVE_WARN]	Location other applies to both boundaries for leaf level cell gives a warning as LS insertion for upper domain boundary is possible.	<i>Top.upf:</i> create_power_domain PD -elements {leaf} set_level_shifter ls3 -domain PD -location other - applies_to_boundary both
<i>Error -</i> [UPF_STRATEGY_MACRO_WRONG_LOCATION]	Error is reported when LS inserted across macro boundary.	<i>Macro.upf:</i> set_design_attributes -models . - is_soft_macro TRUE create_power_domain PD_MODEL1 -elements { . } set_level_shifter ls -domain PD_MODEL1 -location other - applies_to upper
<i>Error -</i> [UPF_STRATEGY_MACRO_WRONG_LOCATION]	Error is reported when LS inserted across macro boundary.	<i>Macro.upf:</i> set_design_attributes -models . - is_soft_macro TRUE create_power_domain PD_MODEL1 -elements { . } set_level_shifter ls -domain PD_MODEL1 -location other - applies_to both
<i>Error -</i> [UPF_STRATEGY_MACRO_WRONG_LOCATION]	Error is reported when LS inserted across macro boundary.	<i>Top.upf:</i> create_power_domain TOP create_power_domain PD1 -elements {macro_inst} set_design_attributes -models macro - is_soft_macro TRUE set_level_shifter ls -domain TOP -location other - applies_to_boundary lower

Error Message	Description	Example
<i>Error - [UPF_STRATEGY_MACRO_WRONG_LOCATION]</i>	Error is reported when LS inserted across macro boundary.	<i>Top.upf:</i> create_power_domain TOP create_power_domain PD1 -elements {macro_inst} set_design_attributes -models macro - is_soft_macro TRUE set_level_shifter ls - domain TOP -location other - applies_to_boundary both

### 5.3.69 Support to Compare Power and Ground voltages of Supply Pairs

VC LP has introduced the following violation tags to compare power and ground voltages of supply pairs. Two new UPF stage violations and two new PG stage violations introduced for supply pair comparison. These tags are disabled by default.

The following are the UPF stage violations:

- ❖ UPF\_SUPPLY\_EQUAL (Warning)

This violation is reported if based on the power states defined in the power intent for the supplies associated with a supply set, there is a power state in which the power supply voltage is same as ground supply voltage. Or, for uncorrelated triplet usage, the power min voltage is same as ground max voltage.

Example UPF extract

Note that Power and ground supply voltage values of supply set SS are equal.

```
add_port_state VDD -state {ON 1.0}
add_port_state VSS -state {ON 1.0}
create_supply_set SS-function {power VDD} -function {ground VSS}
create_pst my_pst -supplies {VDD VSS}
add_pst_state R1 -pst my_pst -state {ON ON}
```

```
Tag : UPF_SUPPLY_EQUAL
Description : Voltage value difference between Power Net and Ground Net listed under supply information [SupplyInfo] is zero.
SupplyInfo
  PowerNet
    NetName: VDD
    NetType : Design/UPF
    PowerMethod : FROM_UPF_POLICY
  GroundNet
    NetName : VSS
    NetType : Design/UPF
    GroundMethod : FROM_UPF_POLICY
  SupplySet : SS
  States
```

State : my\_pst/R1

- ❖ UPF\_SUPPLY\_INSUFFICIENT (Error)

This violation is reported if based on the power states defined in the power intent for the supplies associated with a supply set, there is a power state in which the power supply voltage is lower than the ground supply voltage. Or, for uncorrelated triplet usage, the power min voltage is lower than ground max voltage.

#### Example UPF extract

The ground supply voltage value of supply set SS is higher than power supply voltage

```
add_port_state VDD -state {ON 1.0}
add_port_state VSS -state {ON 1.2}
create_supply_set SS-function {power VDD} -function {ground VSS}
create_pst my_pst -supplies {VDD VSS}
add_pst_state R1 -pst my_pst -state {ON ON}
```

The following is an example of the violation reported:

```
Tag : UPF_SUPPLY_INSUFFICIENT
Description : Voltage value difference between Power Net and Ground Net listed under supply information [SupplyInfo] is negative i.e. voltage value of Power Net is lower than that of Ground Net.

SupplyInfo
PowerNet
  NetName: VDD
  NetType : Design/UPF
  PowerMethod : FROM_UPF_POLICY
GroundNet
  NetName : VSS
  NetType : Design/UPF
  GroundMethod : FROM_UPF_POLICY
SupplySet : SS
States
  State : my_pst/R1
```

The following are PG stage violations introduced:

- ❖ PG\_SUPPLY\_EQUAL (Warning)

This check finds pairs of ground supply nets and power supply nets which are not declared as supply sets in the UPF, but are "discovered" due to the connections on a particular instance. Based on the power states defined in the power intent for these "discovered" supply pairs, there is a power state in which the power supply voltage is same as the ground supply voltage. Or, for uncorrelated triplet usage, the power min voltage is same as ground max voltage.

#### Example UPF extract

Note that VDD and VSS supply voltage values are equal. PG stage violation will be issued if they resolve to power and ground supply of a design cell pin

```
add_port_state VDD -state {ON 1.0}
add_port_state VSS -state {ON 1.0}
create_pst my_pst -supplies {VDD VSS}
add_pst_state R1 -pst my_pst -state {ON ON}
```

The following is example of the violation reported:

```
Tag : PG_SUPPLY_EQUAL
```

```

Description      : Discovered ground net [DesignNet] or Ground Net [InstanceInfo]
voltage value is equal to supply [UPFSupply] voltage value for instance [Instance] pin
[CellPin]
DesignNet
    NetName      : in1
    NetType       : Design
InstanceInfo
    PowerNet
        NetName      : VDD
        NetType       : Design/UPF
        PowerMethod   : FROM_PG_NETLIST
    GroundNet
        NetName      : VSS
        NetType       : Design/UPF
        GroundMethod  : FROM_PG_NETLIST
    UPFSupply      : VDD
    Instance       : leaf_cell_1
    CellPin        : I1
    States
        State       : my_pst/R1

```

❖ PG\_SUPPLY\_INSUFFICIENT (Error)

This check finds pairs of ground supply nets and power supply nets which are not declared as supply sets in the UPF, but are "discovered" due to the connections on a particular instance. Based on the power states defined in the power intent for these "discovered" supply pairs, there is a power state in which the power supply voltage is lower than the ground supply voltage. Or, for uncorrelated triplet usage, the power min voltage is lower than ground max voltage.

Note that VSS supply voltage value is higher than VDD. PG stage violation will be issued if they resolve to power and ground supply of a design cell pin.

```

add_port_state VDD -state {ON 1.0}
add_port_state VSS -state {ON 1.2}
create_pst my_pst -supplies {VDD VSS}
add_pst_state R1 -pst my_pst -state {ON ON}

```

The following is example of the violation:

```

Tag                  : PG_SUPPLY_INSUFFICIENT
Description      : Discovered ground net [DesignNet] or Ground Net in [InstanceInfo]
voltage value is higher than supply [UPFSupply] voltage value, for instance [Instance]
pin [CellPin]
DesignNet
    NetName      : in1
    NetType       : Design
InstanceInfo
    PowerNet
        NetName      : VDD
        NetType       : Design/UPF
        PowerMethod   : FROM_PG_NETLIST
    GroundNet
        NetName      : VSS
        NetType       : Design/UPF
        GroundMethod  : FROM_PG_NETLIST
    UPFSupply      : VDD
    Instance       : leaf_cell_1
    CellPin        : I1
    States
        State       : my_pst/R1

```

Notes:

- ❖ Supply sets explicitly declared in power intent and supply pairs that can be discovered resolving connect\_supply\_net and set\_related\_supply\_net commands using only the power intent are considered as supply pairs for checks in UPF stage.
- ❖ If both UPF checks are disabled, all supply pairs found in PG stage will be considered as supply pairs for PG stage violations instead of only the supply pairs discovered due to connections on instances.

### 5.3.70 Support for Checks on Simstate

Simstates specify the simulation behavior semantics for a power state of supply set and supply set handle. A simstate specifies the level of operational capability supported by a supply set state. The simstate specification provides digital-simulation tools with sufficient information for approximating the power-related behavior of logic connected to the supply set with sufficient accuracy.

All connected supplies with the same supply power state should have same simstate. VC LP performs consistency check on simstate, if connected supplies with same supply power state but with different simstates, the UPF\_SIMSTATE\_INCONSISTENT violation is reported.

#### Example

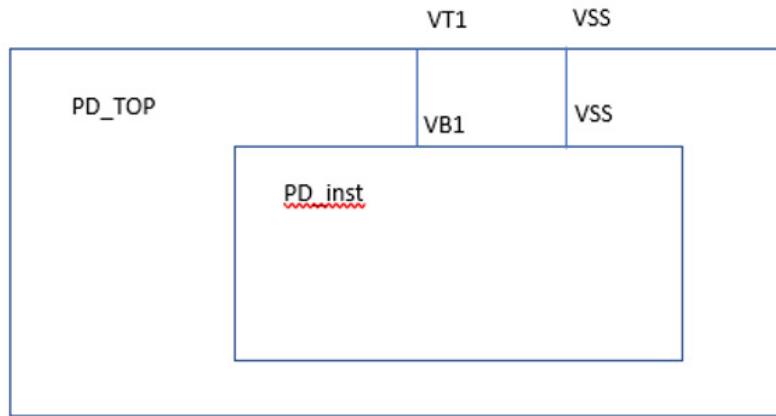
```
create_supply_port VT1
create_supply_port VSS
create_supply_net VT1
create_supply_net VSS
create_supply_set ST1 -function {power VT1} -function {ground VSS}

add_power_state -supply ST1 -state V1 {-supply_expr {power == {FULL_ON 1.1}} \ 
                                         -simstate NORMAL }

set_scope inst1
create_supply_port VB1
create_supply_port VSS
create_supply_net VB1
create_supply_net VSS
create_supply_set SB1 -function {power VB1} -function {ground VSS}
add_power_state -supply SB1 -state V1 {-supply_expr {power == {FULL_ON 1.1}} \ 
                                         -simstate CORRUPT }

set_scope ..
connect_supply_net VT1 -ports {inst1/VB1}
connect_supply_net VSS -ports {inst1/VSS}
```

The following is an example of Simstate inconsistency:



**Violation:**

```

-----  

UPF_SIMSTATE_INCONSISTENT           (1 errors/0 waived)  

-----  

Tag : UPF_SIMSTATE_INCONSISTENT  

Description : Power state simstate of supply [UPFSupply] in set  

[SupplySet] is inconsistent  

Violation : LP:1  

UPFSupply : VT1  

SupplySet : ST1  

PowerStateName : ST1/V1  

Simstate : NORMAL  

PowerStateInfoList  

PowerStateInfo  

UPFSupply : inst1/VB1  

SupplySet : inst1/SB1  

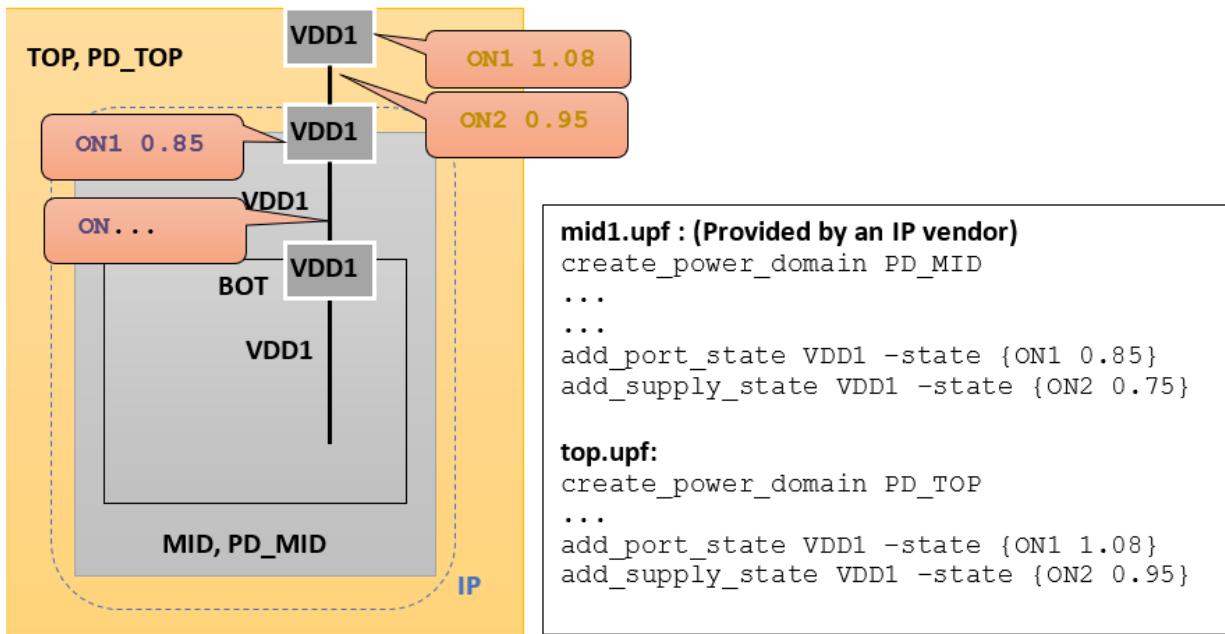
PowerStateName : inst1/SB1/V1  

Simstate : CORRUPT
  
```

### 5.3.71 Support for Duplicate Port State Name with Different Voltages

VC LP allows users to define multiple ON states of the same name and different voltages on connected objects. This feature is necessary when an IP's UPF which has port states defined is the same as the port states defined in a different IP's UPF. Without the feature, the you should edit the IP's UPF to resolve ON state conflicts.

An example of the ON states between the UPF in Top and the UPF in MID (IP) is shown in the figure. VC LP no longer reports an error in such cases.



### 5.3.72 Support for `get_resolved_elements` Command

The `get_resolved_elements` command is introduced to get the collection of resolved objects related to the strategies: isolation, level shifter or retention. The input collection may contain any number of strategies.

#### Syntax

```
get_resolved_elements <strategies> [-quiet]
```

Where:

- ❖ `<strategies>`: Specifies strategy names.
- ❖ `[-quiet]`: Suppresses all messages. Syntax error messages are not suppressed.

#### Example

```
get_resolved_elements [get_retention_strategies]
```

#### Output

```
{"i_core1/i_core11/core11_out1", "i_core2/i_core22/core22_out1[1]",  
"i_core2/i_core22/core22_out1[2]", "ret", "out3[1]"}
```

### 5.3.73 Support for Treating HighConn of Macro as Domain Boundary

HighConn of macros that do not have a power domain of their own are treated as domain boundary as well as boundaries of the root cells of power domains.

To enable this feature, the following two conditions should be satisfied-

- ❖ Lower domain boundary feature should be turned on.
- ❖ UPF design attribute `macro_as_domain_boundary` should be set true.



Lower domain boundary can be enabled in one of the following methods:

1. By setting the `lower_domain_boundary` UPF design attribute to TRUE for the top-level scope or a block-level scope.
- Or
2. By specifying `-applies_to_boundary lower` or `-applies_to_boundary both` in `set_isolation`, `set_level_shifter`, `set_repeater` commands.

The `macro_as_domain_boundary` UPF design attribute is introduced to achieve backward compatibility. This new design attribute is a scope-level attribute to indicate whether macros at or below that scope need to be considered as domain boundary or not.

## Syntax

The design attribute to be supported with both `-elements` and `-models` options. Here are the usage models with both the options.

```
set_design_attributes -elements {scope} \
-attribute macro_as_domain_boundary TRUE|FALSE
```

Here, scope can be ".", a hierarchical instance or a hard macro instance, otherwise, VCLP will issue parser error.

```
Set_design_attributes -models {hard_macro_library_cell} \
-attribute macro_as_domain_boundary TRUE|FALSE
```

Only hard macro library cells are allowed with `-models` option, otherwise, VC LP reports parser error.

Follow these rules while setting the new design attribute:

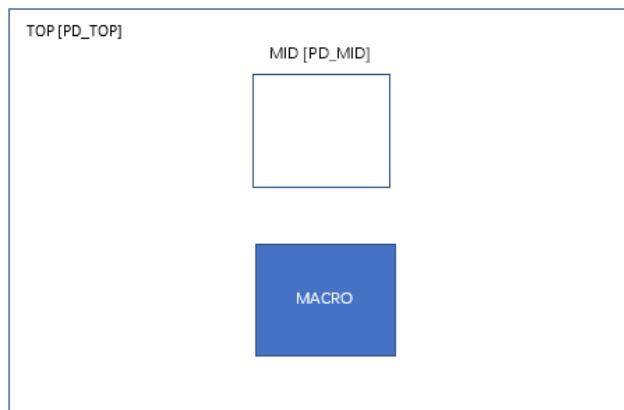
1. If a particular hierarchy/scope does not have an explicit setting, then the hierarchy/scope will inherit the value from the closest ancestor having an explicit setting.
2. If the attribute is set to one value on a model and to the opposite value on the instance of the model, then the value on the instance will take precedence.

## Crossover generation

In crossover generation, one extra boundary node is created for such hard marco pins, similar to the domain boundary node for `lower_domain_boundary`.

### Examples

Consider the following design:



```
set_design_attributes -elements { . } -attribute lower_domain_boundary TRUE
set_design_attributes -elements { . } -attribute macro_as_domain_boundary TRUE
create_power_domain PD_TOP -elements { . }
create_power_domain PD_MID -elements { MID }
set_isolation ISO -domain PD_TOP -applies_to outputs
```

or

```
set_design_attributes -elements { . } -attribute macro_as_domain_boundary TRUE
create_power_domain PD_TOP -elements { . }
create_power_domain PD_MID -elements { MID }
set_isolation ISO -domain PD_TOP -applies_to outputs -applies_to_boundary both
```

The isolation strategy will also be applied to the HighConn of macros. So, for this UPF, isolation policies are considered at:

- ❖ Output ports of TOP (upper domain boundary)
- ❖ Input ports of MID (lower domain boundary)
- ❖ Input ports of MACRO (lower domain boundary as per the design attribute)

### Limitations

Terminal boundary support is not available with the `macro_as_domain_boundary` attribute. False `UPF_ELMNOTINDOMBOUND` parser message is reported for LS strategy written for macros with new SDA attribute.

## 5.3.74 Support for SPA -driver/-receiver and SRSN on Hard Macro

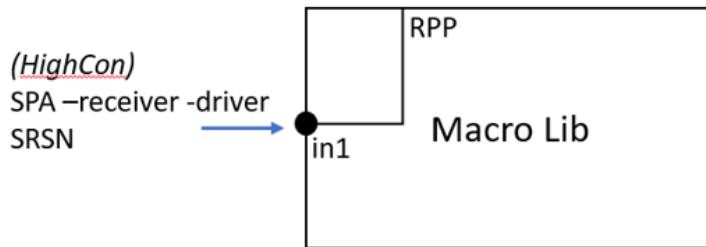
The following behavior is present in VC LP for supporting SPA -driver/-receiver and SRSN on hard macros.



**Note**  
When `-model` is present along with the driver/receiver supply attribute in `set_port_attributes` command, VC LP reports the `UPF_ATTR_INVALID_OBJECT` warning message.

### 5.3.74.1 When only HighCon UPF Constraints are Defined (SPA/SRSN)

VC LP supports `set_port_attributes -receiver_supply/-driver_supply` on hard macro pins when macro pin is not in a power domain boundary.



VC LP honors the following priority order when resolving the root supply for macro input pin:

*(HighCon) SPA -receiver\_supply > (HighCon) set\_related\_supply\_net (SRSN) > Related Power Pin (RPP)*

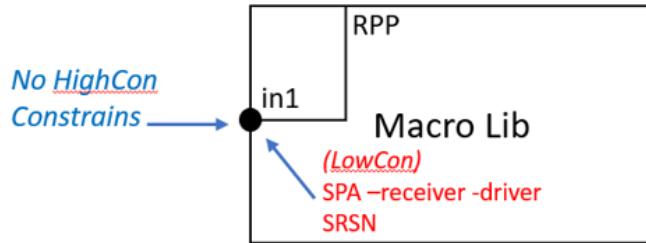
Limitations

[Warning] UPF\_PORT\_NOT\_ON\_BOUNDARY is reported when SPA is defined on Macro pin which is not in a power domain boundary. This warning will be removed in the future releases.

### 5.3.74.2 When Both HighCon and LowCon UPF Constraints are Defined (SPA/SRSN)

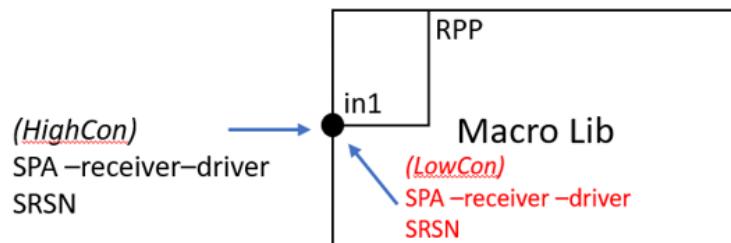
VC LP honors the following priority order when resolving root supply for macro input pin  
*(HighCon) SPA > (HighCon) SRSN > Related Power Pin (RPP)*

#### Example



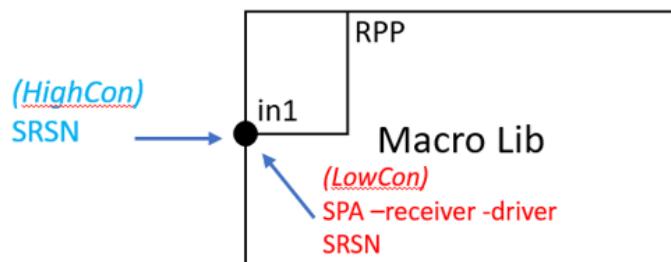
- ❖ Macro/in1 root supply is resolved from the related power/ground supplies (RPP/RGP) of the database.
- ❖ VC LP ignores the LowCon SPA/SRSN for root supply resolution.

#### Example 1



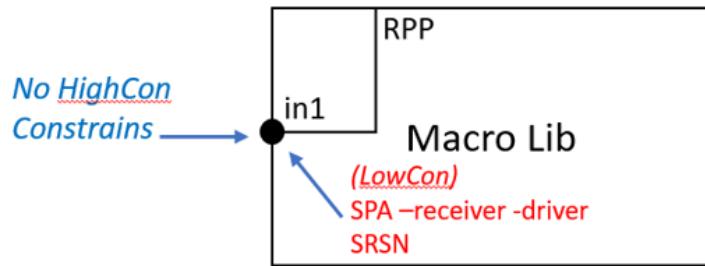
Root supply of Macro/in1 is resolved from (HighCon) SPA -receiver. The UPF\_SPADRIVER\_STATE is reported for HighCon SPA -driver\_supply when PST has state where SPA driver supply is on, but supply of actual driver is off.

#### Example 2



Root supply of Macro/in1 is resolved from (HighCon) SRSN. The UPF\_SPADRIVER\_STATE is reported for LowCon SPA -driver\_supply when PST has a state where SPA driver supply is on, but supply of actual driver is off.

### Example 3:



- ❖ Macro/in1 root supply is resolved from (LowCon) SPA -receiver
- ❖ UPF\_SPADRIVER\_STATE is reported for LowCon SPA driver when PST has a state where SPA driver supply is on, but the supply of actual driver is off.
- ❖ UPF\_ATTR\_MISMATCH\_DRIVER\_RECEIVER\_SUPPLY\_WITH\_DB\_PORT\_RPP is parser level Warning is reported comparing (LowCon) SPA -receiver with its RPP supply.

*[Warning] UPF\_ATTR\_MISMATCH\_DRIVER\_RECEIVER\_SUPPLY\_WITH\_DB\_PORT\_RPP:  
Mismatch between driver/receiver supply and DB power/ground For DB port, receiver supply  
'top/macro1/SS\_MACRO' specified with 'set\_port\_attributes' is ignored due to mismatch with DB supply ''  
when specified at the top-scope of DB 'MRM1'.*

Change driver/receiver supply with 'set\_port\_attributes' to align with DB supply, or specify above the scope of the instantiated DB.

### 5.3.75 Support for Defining Isolation Strategy

You can define an isolation strategy separately from the -location option. The goal of this fix is to support the specification of an isolation strategy location, after defining the isolation strategy using the set\_isolation -location -update.

If the strategy was set with set\_isolation in combination with set\_iso\_control, the -location -update cannot be done.

When the -update option is specified, only the options: strategy\_name, -domain, -elements, -exclude\_elements, and -location options are accepted, any other options causes the command error out with syntax error.

```
set_isolation strategy_name -domain power_domain -update \
[-elements {element_list}] [-exclude_elements {exclude_element}] [-location
{self|parent|fan_out}]
```

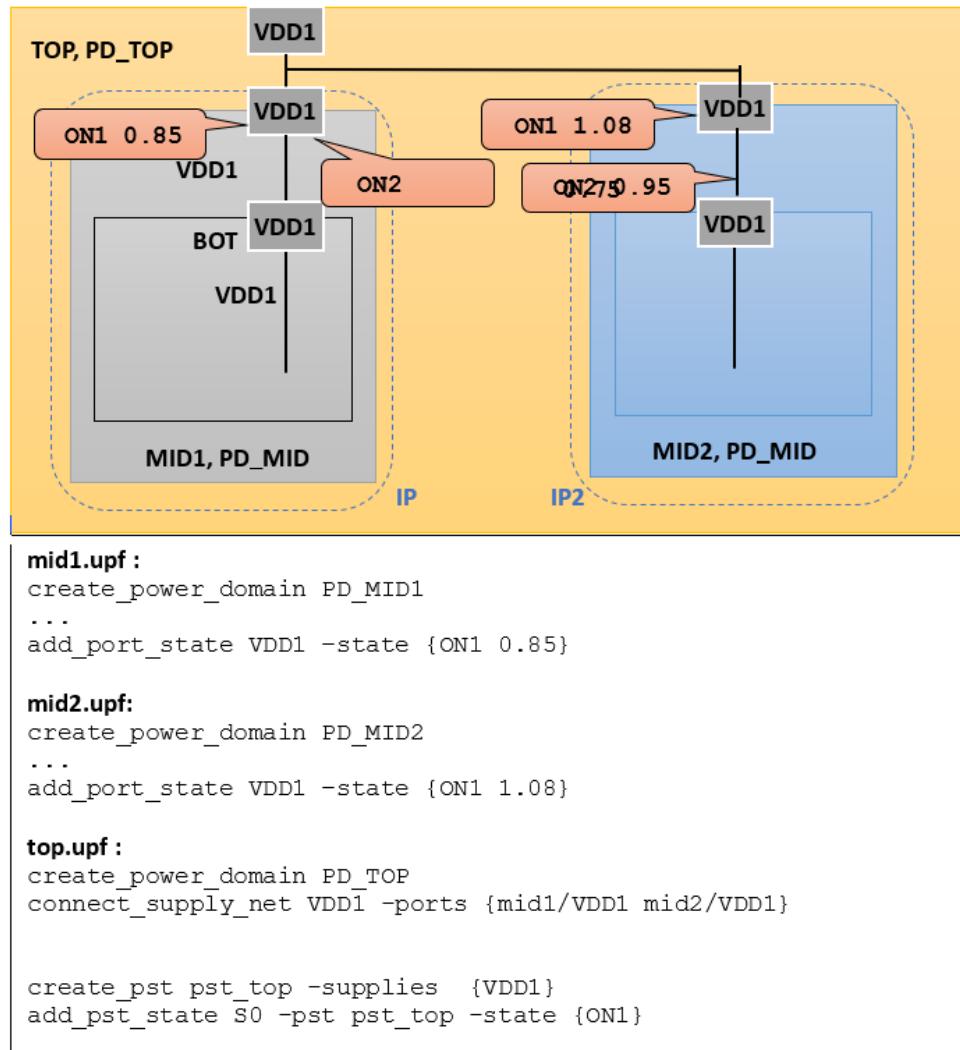
### Example

This is the most common usage. The -location is not specified in the first command and updated with the later command.

```
set_isolation strategy_name -domain PD_name -isolation_signal en {strategy_name}
set_isolation strategy_name -domain PD_name -location parent -update {strategy_name}
```

For more information on the set\_isolation command, see the set\_isolation man page.

A similar conflict would occur on a connected net group between two different IPs. In such cases, you can redefine the port states at top scope for the required voltages on the required supply object. VC LP reports an UPF parser error if there are no any port states defined for such supply objects at top scope.



In this case, VC LP reports the following UPF parser Error.

[Error] *PST\_STATE\_CONFLICT\_RESOLUTION\_FAILED: PST state entry not resolved*  
*PST entry 'ON1' specified for 'TOP/VDD1' in PST state 'S0' of PST 'TOP/pst\_top' resolved to multiple states, but none of states is defined in the scope of 'TOP/VDD1' for tool to choose in priority.*  
*Please define the state in supply object's scope to allow local-first resolution*

### Note

The add\_pst\_state command refers to supply states and port states. Support is available for port states defined by add\_port\_state or add\_supply\_state.

### 5.3.76 Support for isolation\_enable\_condition\_allow\_pg\_pin Attribute

Certain macros have an analog voltage monitor which controls input isolation. If a monitored supply turns off, some inputs of this macro automatically become isolated. The conditions for the isolation are contained in a liberty attribute `isolation_enable_condition_allow_pg_pin`.

VC LP supports the `isolation_enable_condition_allow_pg_pin` liberty attribute with ISO\_VOLTAGE\_ALLOW violation. This violation detects the electrical error which occurs when the macro input is connected to a driver whose supply is less than the monitored supply. In this case, the analog monitor leaves the input isolation transparent, which allows the unknown value from the driver to propagate inside the macro.

#### Example

```
isolation_enable_condition_allow_pg_pin : (!VDD2) | (!VDD3) | (!VDD4);
```



#### Note

VC LP ignores the logic pins mentioned in the attribute value.

The following is an example snippet of the ISO\_VOLTAGE\_ALLOW violation.

```
Tag : ISO_VOLTAGE_ALLOW
Description: Macro input isolation [CellPin] is transparent when enable supplies are
on, but source is off
CellPin: I
Instance: mac1
Cell: SINGLE_PG_MAC
DriverNode: modof/andgate/Z
SourceInfo
...
SinkInfo
...
Supplies
UPFSupply: VDDC2
States
State: pst/s1
```

The ISO\_ENABLE\_PGEPR violation relies on the liberty attribute

`isolation_enable_condition_allow_pg_pin`. In `isolation_enable_condition_allow_pg_pin` liberty attribute value, each supply pin must be negative unate. It must only appear in *active low* form. If these values contain supplies with the active high form then, the ISO\_ENABLE\_PGEPR violation is reported for each supply.

#### Example

```
isolation_enable_condition_allow_pg_pin : !VDD1 & VDD2;
```

In the above example, VC LP reports ISO\_ENABLE\_PGEPR violation for VDD2.

```
Tag: ISO_ENABLE_PGEPR
Description: Active high supply [CellPin] ([Cell]) in isolation condition allowing pg
pin is not permitted
Violation: LP:2
CellPin: VDD2
Cell: SINGLE_PG_MAC
IsolationCondition: !VDD1 & VDD2
```

This also work for complex values such as !(V1 & I) because converted to the sum of products, the result is "!"V1 | !"I".

## Limitations

The liberty attribute equation must have supply pins only in active low form and only joined with OR statements.

### 5.3.77 Support for Automatic Supply Net Generation from `create_supply_port`

Generally, it is possible anticipate the creation of a same name supply net which will be connected to the port at the generation of the supply port it self.

- ❖ User has to create the same name supply nets and connect to the port using `connect_supply_net` command.
- ❖ With this feature, VC LP automatically creates and connects the same name supply nets to supply ports in the `create_supply_port` command.

This feature is disabled by default and the following application variable should be set to enable the feature.

```
set_app_var plato_power_option bcid_2020_12_allow_implicit_supply_net_creation
```

When the application variable is set, the `create_supply_port` command creates a domain independent supply net in addition to the supply port if there is a reference to the supply net in the `create_supply_port` and `create_supply_set` UPF commands.

#### Example

```
create_supply_port VSS
create_supply_set SS1 -function {ground VSS} # supply net 'VSS' is referred here
```

With `set_app_var plato_power_option bcid_2020_12_allow_implicit_supply_net_creation`, this maps to:

```
create_supply_port VSS
create_supply_net VSS
connect_supply_net VSS -ports VSS
create_supply_set SS1 -function {ground VSS}
```

### 5.3.78 Support for ISO\_SIGSUP\_MISSING Violation

VC LP provides the `report_signal_supply -only_unconstrained` Tcl command approach for users to identify missing `lp_signal_supply` (LPSS) constraints between an isolation signal and a source supply. The output of the `report_signal_supply -only_unconstrained` Tcl command cannot be used in the waivers directly.

To address this issue, the `ISO_SIGSUP_MISSING` violation is introduced containing the same information as `report_signal_supply -only_unconstrained`. The `ISO_SIGSUP_MISSING` violation is disabled by default. When `ISO_SIGSUP_MISSING` violation is enabled using the `configure_lp_tag` command, the violation was not reported unless the script contained at least one `lp_signal_supply` command.

To get the `ISO_SIGSUP_MISSING` violation for all the control signals that do not have the `lp_signal_supply` constraints, set the `lp_sigsup_always_report` application variable to true. By default, the `lp_sigsup_always_report` application variable is set to false.

To get the `ISO_SIGSUP_MISSING` violation of only the signals which have no `lp_signal_supply` at all, for any supply, set the `lp_sigsup_only_missing` application variable to true. When the `lp_sigsup_only_missing` application variable is set to true, fewer `ISO_SIGSUP_MISSING` violations are reported, that is, only signals that do not have `lp_signal_supply` at all are reported. You can also use the

`report_signal_supply -only_missing` command to get the report of only the signals which have no `lp_signal_supply` at all, for any supply.

### Example

```
Tag : ISO_SIGSUP_MISSING
Description : Constraint lp_signal_supply missing for control [DesignNet] on isolated
port [StrategyNode]
Violation : LP:9
DesignNet
NetName : isoa
NetType : Design
StrategyNode : i2c/Z
Source
PinName : u2/uc/Q
SourceInfo
PowerNet
NetName : v2
NetType : UPF
PowerMethod : FROM_UPF_POWER_DOMAIN
GroundNet
NetName : vss
NetType : UPF
GroundMethod : FROM_UPF_POWER_DOMAIN
Strategy : d2/s2ac
SenseMismatch
MisSenseDesign : UNKNOWN
MisSenseUpf : HIGH
DesignDriver:
DrivingNode
PathLocation:
ConstValue:
CellType:
Goal
```

### 5.3.79 Support for Defining and Applying Power Models

Power models provides a user friendly way to define power intent for designs that contains thousands of instances of the same macro/module. With this approach, a single power model defining the power intent of the macro can be created and instantiated for all the macro instances with supply mapping included.

Starting with this release, the `define_power_model` command is introduced. The `define_power_model` command defines a power model containing other UPF commands.

#### Syntax

```
define_power_model power_model_name [-for model_list] { [commands] }
```

Where,

- ❖ `power_model_name` is a simple name of the power model. The name is in the namespace of the scope. This is a global power model name.
- ❖ `model_list` is a list of names of the models to which power model applies. Wildcard characters are supported.

If the `-for` option is not specified, then `power_model_name` must be a valid model name in the design.

- ❖ Commands are the list of UPF commands that define the power model architecture. The commands inside curly braces are excluded from Tcl substitution and execution. These commands are processed only when the `apply_power_model` command is specified.

A power model can be referenced by its simple name anywhere in the power intent description. VC LP issues an error message when multiple power models with the same name are specified.

The global variables cannot be used in the commands.

### Example

```
define_power_model macro2 {
    create_power_domain PD_MACRO -include_scope
    create_supply_net VDD;
    create_supply_net VCC;
    create_supply_net VSS;
    add_port_state VCC -state {HV 1.08}
    add_port_state VDD -state {HV 1.08} \
        -state {LV 0.864} \
        -state {OFF off}
    add_port_state VSS -state {ON 0}
    create_pst chiptop_pst -supplies {VCC VDD VSS}
    add_pst_state TURBO -pst chiptop_pst -state {HV HV ON}
    add_pst_state STANDBY -pst chiptop_pst -state {HV LV ON}
    add_pst_state OFF -pst chiptop_pst -state {HV OFF ON}
}
```

### Notes

- ❖ VC LP does not support nested `define_power_model` commands. In this case, VC LP issues a passer error.
- ❖ The `define_power_model` command provides a protected evaluation environment for all `apply_power_model` calls irrespective of the model being a hard macro or not.
- ❖ All global variables in a power model must be defined using the `add_parameter` command and overridden using the `-parameters` option of the `apply_power_model` command.

#### 5.3.79.1 Applying Power Model

A power model can be applied to specific instances of a design using the `apply_power_model` command. A power model that is not referenced by an `apply_power_model` command does not have any impact on the power intent of the design.

The `apply_power_model` command describes the connections of the interface supply set handles of a previously loaded power model with the supply sets in the scope where the corresponding macro cells are instantiated.

### Syntax

```
apply_power_model power_model_name
    [-elements instance_list]
    [-supply_map {{ lower_scope_supply_set upper_scope_supply_set}*} ]
    [-port_map {{ lower_scope_supply_or_logic_port upper_scope_supply_or_logic_net}*} ]
    [-parameters {{power_model_parameter override_value}*} ]
```

Where,

- ❖ **-elements:** Allows you to load the power model for instances specified in the -elements list. Each instance in the list is a simple name or a hierarchical name rooted in the current scope.
- ❖ **-supply\_map:** Allows you to connect the upper level supply sets to the supply sets in the power model. The path of the lower\_scope\_supply\_set argument does not include either the power model name or the instance name to which the power model is applied. This is the path relative to the scope of the power model.
- ❖ **-port\_map:** Allows you to connect the supply port or logic ports in the power model to the supply net or logic net present in the parent scope. The lower\_scope\_supply\_or\_logic\_port argument is a path relative to the power model.
- ❖ **-parameters:** Allows you to override the value of any Tcl variable created through the add\_parameter command. This option accepts any string value. If you specify incorrect values, then the corresponding error messages are issued during UPF processing of the command.

#### Example

```
apply_power_model upf_model -elements I1
```

### 5.3.80 Support for Automatic Supply Net Generation from `create_supply_port`

Generally, it is possible anticipate the creation of a same name supply net which will be connected to the port at the generation of the supply port it self.

- ❖ User has to create the same name supply nets and connect to the port using `connect_supply_net` command.
- ❖ With this feature, VC LP automatically creates and connects the same name supply nets to supply ports in the `create_supply_port` command.

This feature is disabled by default and the following application variable should be set to enable the feature.

```
set_app_var plato_power_option bcid_2020_12_allow_implicit_supply_net_creation
```

When the application variable is set, the `create_supply_port` command creates a domain independent supply net in addition to the supply port if there is a reference to the supply net in the `create_supply_port` and `create_supply_set` UPF commands.

#### Example

```
create_supply_port VSS
create_supply_set SS1 -function {ground VSS} # supply net 'VSS' is referred here
```

With `set_app_var plato_power_option bcid_2020_12_allow_implicit_supply_net_creation`, this maps to:

```
create_supply_port VSS
create_supply_net VSS
connect_supply_net VSS -ports VSS
create_supply_set SS1 -function {ground VSS}
```

### 5.3.81 Support for `APS_LOGICEXP_EQUIVALENT` Violation

VC LP checks if the logic expressions of different power states for the same supply set are equivalent or not. The check is performed between each two power states of them, that could be between ON and OFF states or two ON states. As long as the expressions are logically equal, they are considered as equivalent, so it could be following cases:

```
> 1. "a & b" vs "b & a"
> 2. "a & b" vs "!( (!b) | (!a)) "
> 3. "(a&b) | (a&b&c)" vs "a&b"
```

The APS\_LOGICEXP\_EQUIVALENT violation is reported in such cases with debug field listing the power states and their logic expressions:

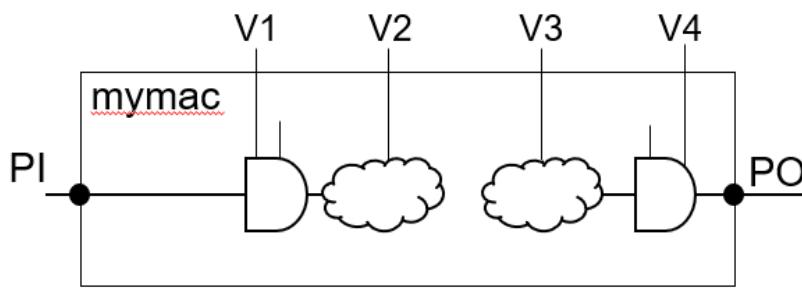
```
Tag : APS_LOGICEXP_EQUIVALENT
Description : add_power_state (APS) supply set [SupplySet] power states' logic expressions are equivalent
Violation : LP:3
SupplySet : SS_PD1
PowerStateExprList
  PowerStateLogicExprInfo
    PowerStateName : SS_PD1/v0p8
    PowerStateLogicExpr : (psw_en0 & psw_en1)
  PowerStateLogicExprInfo
    PowerStateName : SS_PD1/v1p8
    PowerStateLogicExpr : (! ((!psw_en0) | (!psw_en1) ))
```

### 5.3.82 Support for isolation\_sink\_power\_pin and isolation\_data\_power\_pinLiberty Attributes

Starting with this release, there are two new liberty attributes introduced to identify one level further inside of related power pin of the supply for macros having ISOLATED input/output.

These attributes enables you to allow/block checks on isolated macro inputs and outputs.

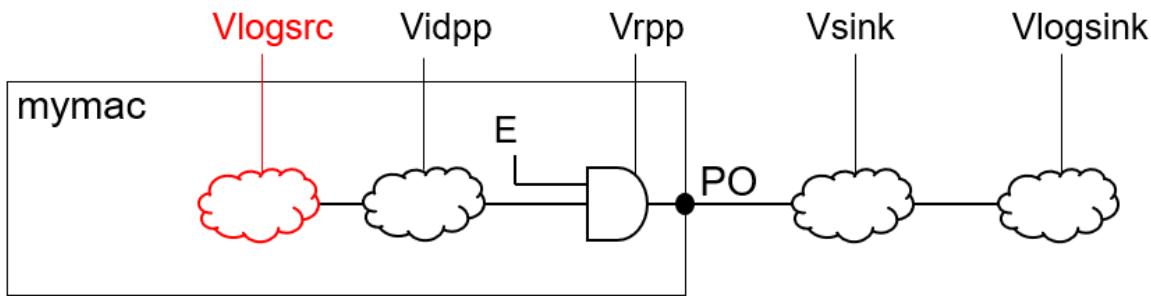
```
pin (PI) {
  related_power_pin : V1;
  isolation_sink_power_pin : V2; }
pin (PO) {
  related_power_pin : V4;
  isolation_data_power_pin : V3; }
```



If the new attributes are not specified, VC LP assumes they are the same as the related\_power\_pin

#### Example for allow and partial block checks on macro outputs

Consider the following example design:

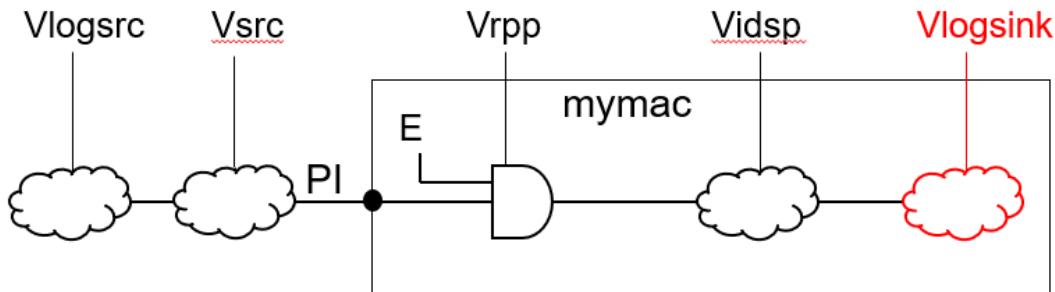


Given the supply of the `isolation_data_power_pin` (`Vidpp`) and a `lp_signal_supply` constraint on the enable `E`, VC LP can perform an “allow” check on the path

`Isolation_enable_condition` of `PO` must be a single pin (`E` in this example); this pin may have direction internal and does not need to be connected to a net in the design. The Real block check is not possible, since VC LP does not get the logic source supply; VC LP can approximate use `Vidpp`, but false positives are possible.

#### Example for allow and “partial block” checks on macro inputs

Consider the following figure



#### Same points as macro outputs, but on inputs:

Given the supply of the `isolation_data_sink_pin` (`Vidsp`) and a `lp_signal_supply` constraint on the enable `E`, VC LP can perform an “allow” check on the path. The `Isolation_enable_condition` of `PI` must be a single pin (`E` in this example); this pin may have direction internal and does not need to be connected to a net in the design

“Real” block check is not possible, since VC LP still does not get the logic sink supply; VC LP can approximate using `Vidsp`, but false positives is possible.

### 5.3.83 Support for coupled\_supplies States Check

In the following library model, a user defined `coupled_supplies` attribute, to specify a group of coupled supply pins and a group of ground pins. For each group, the pg pins are weakly coupled by resistors, and expected to share the same voltage in all legal system power states. In long term, this user-defined attribute will be defined by LC team and added to liberty spec.

```
define ("coupled_supplies",  "cell",  "string");
.
```

```
cell(AND2) {
    is_macro_cell : true;
    coupled_supplies : "(VDD1 VDD2 VDD3) (VSS1 VSS2 VSS3)";
}
```

VC LP supports the following DESIGN\_SUPPLY\_STATE and DESIGN\_SUPPLY\_VOLTAGE violations to check for coupled\_supplies.

These violations are reported in the design stage, with family *DesignConsistency*, and are disabled by default.

- ❖ DESIGN\_COUPLED\_STATE: A state exists in which supplies coupled inside a macro have different ON states
- ❖ DESIGN\_COUPLED\_VOLTAGE: A state exists in which supplies coupled inside a macro have different voltages.

### 5.3.84 Support for upf\_cmd\_wrapper\_prefix Application Variable

When you save the expanded UPF within VC LP, and Tcl proc wrappers are used to conditionally write UPF syntaxes, VC LP did not properly expand these syntaxes.

#### Example

Assume a SNPS\_<UPF syntax> proc is used to replace UPF syntaxes in the UPF.

Tcl containing procs:

```
proc SNPS_set_port_attributes {args} {
    ..... <Proc to describe how set_port_attributes should be written based on arguments
    and variable settings>
}
```

UPF

SNPS\_set\_port\_attributes ....

Use the following steps when this type of strategy is used, and to preserve the order in which syntaxes appeared in UPF.

1. If procedures are intended to be used instead of UPF syntaxes to replace existing UPF syntaxes, such procs should be in following form.  
`<prefix>_<UPF Syntax>`
2. Use the same prefix for the complete design.
3. Enable the following application variable  
`set_app_var upf_cmd_wrapper_prefix <prefix>`
4. Specify the `read_upf` command in the Tcl file after setting the `upf_cmd_wrapper_prefix` application variable.



#### Note

This support is intended and developed for especial user flows. It is intended to be used if you have such a specific flow for which this support is relevant.

### 5.3.85 Support for Atomic Power Domains

There are instances where the functional operation of a design would fail later along the process because the power domains specified in the constraint UPF are deployed together with the RTL (for an IP) undergo a change during implementation.

Therefore, there is a requirement to keep a power domain intact during implementation. IEEE 1801 supports this requirement through atomic power domains.

VC LP supports atomic power domains. Atomic power domains are used to inform IP users that the domain should not be split further for efficiency purposes. You should not define elements in the hierarchy of the scope of an atomic power domain as elements of any other power domain.

#### Syntax

To define an atomic power domain, use the `-atomic` option of the `create_power_domain` command:

```
create_power_domain -atomic  
  [-elements element_list]  
  [-exclude_elements exclude_list] [...]
```

To define an atomic power domain, specify the `-atomic` option during the first definition of that power domain. If the `-atomic` option is specified later with the `-update` option, the following error message is reported, and the constraint is dropped.

*[Error] TCL\_OPT\_NOT\_TOGETHER: Incorrect command options*

*Atomic power domains can be defined on an empty element list, macro/ETM/black box or leaf cells*

You can update the extent of an atomic power domain, that is, `-elements {}` and `-exclude_elements {}` is allowed.

VC LP does not allow any instance in the descendant subtree of an atomic power domain to be included in the extent of another power domain, unless that instance name is, or is in the descendant subtree of, an instance which is excluded from that atomic domain. Violating these conditions results in the following parser error.

*[Error] UPF\_INSTANCE\_EXISTS\_IN\_ANOTHER\_ATOMIIC\_PD*

### 5.3.86 Support for ISO\_STRATEGY\_SELF Violation

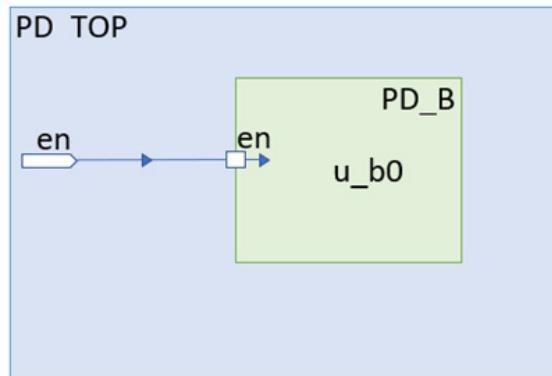
Starting with this release, the ISO\_STRATEGY\_SELF is reported when an isolation strategy has a self dependency with applied node and enable signal.

Suppose an isolation strategy is defined with `-applies_to inputs`, and the enable also is an input, then the result is also an isolation gate with the same data and enable signal, however, the output signal of that gate drives all the other isolation inputs. The ISO\_STRATEGY\_SELF violation is reported in such cases.

Furthermore, the isolation strategy cannot be applied on the enable directly. But it is applied on boundary node which is the fanout/fan-in of its enable. The applied node and the enable will have self dependency.

#### Example

Consider the `u_bo` design has an `en` port, and it has path `ub0/en` (enable signal). In UPF, `u_bo` instance is located under `PD_B` domain and `ISO_B` is applied on `u_bo/en` pin.



```
#upf
create_power_domain PD_B -supply {primary SS} -elements {u_b0}
set_isolation ISO_B -domain PD_B \
    -clamp_value 0 \
    -applies_to inputs \
    -isolation_signal u_b0/en \
    -isolation_sense low \
    -isolation_supply_set SS0 \
    -location self
```

The following is an example output of the violation report:

```
-----  
ISO_STRATEGY_SELF (1 errors/0 waived)  
-----  
Tag : ISO_STRATEGY_SELF  
Description : Isolation strategy [Strategy] has self dependency on element [StrategyNode] and  
Violation : LP:12  
Strategy : PD_B/ISO_B  
StrategyNode : u_b0/en  
UPFNet  
    NetName : u_b0/en  
    NetType : Design/UPF  
Source  
    PinName : u_b0/en  
SegmentSourceDomain : PD_B  
Sink : u_b0/u_b0/E0_tmp/D  
SegmentSinkDomain : PD_B0  
LogicSource  
    PinName : u_b0/en  
    Unconnected : True  
LogicSink : u_b0/u_b0/E0_tmp/D  
SourceInfo  
    PowerNet  
        NetName : VDD  
        NetType : UPF  
    PowerMethod : FROM_UPF_POWER_DOMAIN  
GroundNet  
    NetName : VSS  
    NetType : UPF  
    GroundMethod : FROM_UPF_POWER_DOMAIN  
SinkInfo  
    PowerNet  
        NetName : VDD0  
        NetType : UPF  
    PowerMethod : FROM_UPF_POWER_DOMAIN  
GroundNet  
    NetName : VSS  
    NetType : UPF  
    GroundMethod : FROM_UPF_POWER_DOMAIN
```

### 5.3.87 Support for LS\_COMBO\_FUNC Violation

The LS\_COMBO\_FUNC is reported when source and logic sink are being powered with same supply net, and there is a buffer/inverter/combo cell on the path which is at a different voltage. The LS\_COMBO\_FUNC violation is disabled by default, and can be enabled using the `configure_lp_tag -tag LS_COMBO_FUNC -enable` command.

The LS\_COMBO\_FUNC violation should be controlled using the `set_app_var lp_enable_iso_combo_checks true` application variable.

#### Example

```
top/mod11/ff1/Q (V1) > top/mod12/and1/Z (V2) > top/mod12/out --> top/mod13/ff1/D (V1)
```

The LS\_COMBO\_FUNC violation is reported for top/mod12/and1/Z.

LS_COMBO_FUNC		(9 warnings/0 waived)
Tag	:	LS_COMBO_FUNC
Description	:	Power supplies for source [LogicSource] and sink
LogicSource	:	
PinName	:	mod11/ff1/Q
LogicSink	:	mod13/ff1/D
CellPin	:	Z
Instance	:	mod12/and1
Cell	:	AND2

### 5.3.88 Supports for Full SoC Validation of LS Source - Sink Strategies

VC LP supports full SoC validation for the level shifter-source -sink strategies. To enable this support, set the `enable_fullsoc_strategy_validation` application variable to true.

When the `enable_fullsoc_strategy_validation` application variable is set to true, VC LP enables the new error checking mode where the segmented crossovers due to terminal boundary attribute are used for policy applicability and device to policy association, but error checking is done with the end-to-end crossover by ignoring terminal boundaries. The LS -source -sink strategies are also supported under the `enable_fullsoc_strategy_validation` application variable.

When the user specifies the terminal boundaries, the level shifter strategies that get associated on the power domain boundaries can be different from the strategies that get applied. The terminal boundary ports act as a valid source/sink, so the level shifter strategy that gets associated with boundary ports may differ depending on whether terminal boundary is specified or not. To indicate such differences in the level shifter strategy association, the LS\_ASSOC\_DIFFER (warning) violation is introduced. This violation is reported at the UPF stage.

The following is an example of the LS\_ASSOC\_DIFFER violation:

Tag	:	LS_ASSOC_DIFFER
Description	:	At node [StrategyNode], level shifter strategy applied with TB: [Strategy]; level shifter strategy applied without TB: [IgnoredStrategies]
StrategyNode	:	CORE1/in1
Strategy	:	CORE1/PD1/ls1
LogicSource	:	
PinName	:	in1
LogicSink	:	CORE1/ff_i1/D
ReasonCode	:	Strategy is applied only when terminal boundary is set

### 5.3.89 Support for is\_virtual Design Attributes

Starting with this release, VC LP supports a new way to model virtual supplies (supply ports and supply nets) with the `{is_virtual true}` attribute. The `is_virtual` is a new attribute introduced for `set_design_attributes` and `set_port_attributes`.

#### Syntax

```
set_design_attributes -elements . -attribute is_virtual <supply nets>
set_port_attributes -elements . -ports <supply ports> -attribute {is_virtual true}
```

#### Examples

```
set_design_attributes -elements {.} -attribute is_virtual {SN2'}
set_port_attributes -elements {.} -ports {SP2'} -attribute {is_virtual true}
```

The value of this attribute is supply net/port name, and also can be a list of supply net/port names.

### 5.3.89.1 Impacted Violations

- ❖ PG\_CSN\_CONN: The PG\_CSN\_CONN violation of is not reported when the UPF has attribute `is_virtual` is true.
- ❖ UPF\_PORT\_VIRTUAL: This violation is reported when the supply port <x> is declared virtual in UPF, but appears in the design. If it appears in design, the lib pin should be with direction: internal.
- ❖ UPF\_NET\_VIRTUAL: This violation is reported when the supply net<x> is declared virtual in UPF, but appears in the design. It is possible that a top level net may connect to some lower level nets, and some of those lower level nets may be virtual. It would only report on the lower level supply net.
- ❖ UPF\_SUPPLY\_NOSTATE: The debug field ReasonCode: IS\_VIRTUAL is added if the supply\_net is defined as virtual.
- ❖ Driver/load checks

The behavior of the following tags when virtual nets/ports are involved are updated as mentioned in the following table

- ◆ UPF\_SUPPLY\_UNDRIVEN
- ◆ UPF\_SUPPLY\_UNUSED
- ◆ UPF\_SUPPLY\_NOLOAD

		Receiver port types present			
		none	Only virtual	Only real	Both types
Driver port types present	None	UNUSED	UNDRIVEN	UNDRIVEN	UNDRIVEN
	Only virtual	NOLOAD		UNDRIVEN (OVD)	UNDRIVEN (OVD)
	Only real	NOLOAD	NOLOAD		
	Both types	NOLOAD	NOLOAD		



# 6 Using VC LP in Various Design and Verification Flows

This section describes how VC LP can be used in different hierachal verification flow.

- ❖ “Hierarchical Verification Flows”
- ❖ “Golden UPF Flow”
- ❖ “Support for Signoff Abstraction Model Based Hierarchical Flow”

## 6.1 Hierarchical Verification Flows

### 6.1.1 Black Box or Stub Flows

In the traditional Synopsys low power flow, UPF can be created only for instances for which the design hierarchies are fully available, that is, the design references in the UPF are fully resolved. These instances are referred to as hierarchical instances or hierarchical objects. The RTL or Tcl for these instances is complete and is a representation of the actual design. The instances for which the full design hierarchy is not available or visible are called non-hierarchical instances or non-hierarchical objects.

A black box model is a design block of which the HDL (Hardware Description Language) internals of the model (or module) are not visible to the tool, only the boundary ports are visible.

If an IP is completely verified with its UPF, such that it can be used at top design in full flat verification as a black box. It can be done by defining such IPs as black box flow is also used for parallel design development if boundary conditions are known to us.

Design elements must be available either as definitions in the library (.db) files which can be read, or as modules in the source files which can be read with the `read_file` or other commands. In the RTL or netlist flow, an empty module (that is, a module or an architecture where only port list is present) is named a black-box. An instance that is unresolved (that is, there is no corresponding module or architecture) is named as an unresolved module.

Once the elaboration complete, a list of black-box modules is reported with the following warning message:  
*[Warning] SM\_BB\_LIST: Black-box module list.*

A list of unresolved modules is reported with the following warning message:

*[Warning] SM UM\_LIST Unresolved module list.*

If a design is read by the `read_file -netlist` command and if there is an empty module, then it is considered as black box, and reports the following informational message:

[Info] SM\_EMPTY\_SKIP: Marking empty Module/Entity as blackbox and warning [Warning] SM\_BB\_LIST: Blackbox module list.

You can control the automatic black-boxing of the unresolved modules using the `autobb_unresolved_modules` application variable. By default, the `autobb_unresolved_modules` variable is set to true.

- ❖ When `autobb_unresolved_modules` are set to false, an instance that is unresolved is reported as an error message (*Error-[SM\_URMI] Unresolved module instance in design*) and the elaboration fails.
- ❖ When the `autobb_unresolved_modules` are set to true, the unresolved modules are considered as black boxes and is reported as a warning message (*[Warning] SM UM LIST Unresolved module list*).

Alternatively, use the `report_link` command to get a report of the black-boxes and unresolved modules.

#### 6.1.1.1 The `enable_blackbox_during_elab` Application Variable

VC Static gets the synthesis view of the design. In the synthesis view of the design, some of the RTL modules are set as black boxes. In some cases of dirty design, for example, port mismatch between module declaration and instances below the black boxed hierarchy, VC static reports elaboration errors which cannot be downgraded. To avoid such errors, you can set the `enable_blackbox_during_elab` application variable to true. By default, the `enable_blackbox_during_elab` application variable is set to false, and VC LP reports errors during elaboration in such cases.

When the `enable_blackbox_during_elab` application variable is set to true, VC Static elaboration honors the hierarchy below the black box scope.

#### 6.1.1.2 Advantages of Black Box Flow

- ❖ It allows use of a hierarchical flow with black box.
- ❖ The black box flow provides the advantage of huge capacity and runtime improvement for full chips verification and enables user by a single top only verification.
- ❖ In addition, some users write the UPF for their blocks in such a way that they are scoped at the block (which is loaded using `load_upf -scope`). Such UPF models are currently not supported for non-hierarchical instances like black boxes. Such UPF will be supported with the black box flow.

#### 6.1.1.3 Example Design Containing Black Boxes

```
module top();
  u1 block_stub_a(a,b,c); /*instantiate Verilog module (stub) */
  u2 block_stub_b(a,b,c); /*instantiate Verilog module (stub) */
  u3 block_stub_b(a,b,c); /*instantiate Verilog module (stub) */
endmodule
module block_stub_a(input a,b; output c); /* Empty module */
endmodule
module block_stub_b(input a,b; output c); /* Empty module */
endmodule
```

#### 6.1.1.4 Enabling Black Box Flow

Use the `set_blackbox` command to define a black box. VC LP considers all these user-specified modules as black box even if the design file has a complete module definition. All the internal logic of these modules are ignored during elaboration/crossover generation.

The `set_blackbox` command must be used before reading the design. It is not mandatory to have the list as domain boundary or non-domain boundary.

#### Syntax

```
%vc_static_shell> set_blackbox -help
set_blackbox # Marks BBOX in the design
[-cells <cell>] (List of cell to be set as black box. i.e. instances name)
[-designs <design>] (List of design to be set as BBOX. i.e module name)
#Note: list of instance/module names that are to be considered as black box should be
separated by " " (space).
```

## Use Model

```
%vc_static_shell> set_blackbox -cells "ul" ( using Instance name )
```

or

```
%vc_static_shell> set_blackbox -designs { block_stub_a block_stub_b } (using module
name)
```

The following is the output of the report\_link command which provides the list of User defined black box.

Module	Instances	Reason
block_stub_a	1	UserBB

VC LP supports sourcing tcl files once scope below black box scope.

Example:

```
set_blackbox -design core1
```

*core1 UPF:*

```
load_upf core_internal.upf -scope internal1
```

*core internal UPF:*

```
source test.tcl
```

```
load_upf test.upf
```

The content of test.tcl is sourced in VC LP run. Note that contents in test.upf will not be read in.

### 6.1.1.5 UPF Requirement for Black box Flow

The following are the UPF requirements for an accurate (safe) hierarchical verification methodology:

- ❖ The external boundary constraints that impact the functionality of the level being verified must be specified.
- ❖ These constraints must be visible and validated when verifying the parent level or the child level

```
set_port_attribute -driver_supply/-receiver_supply
```

or

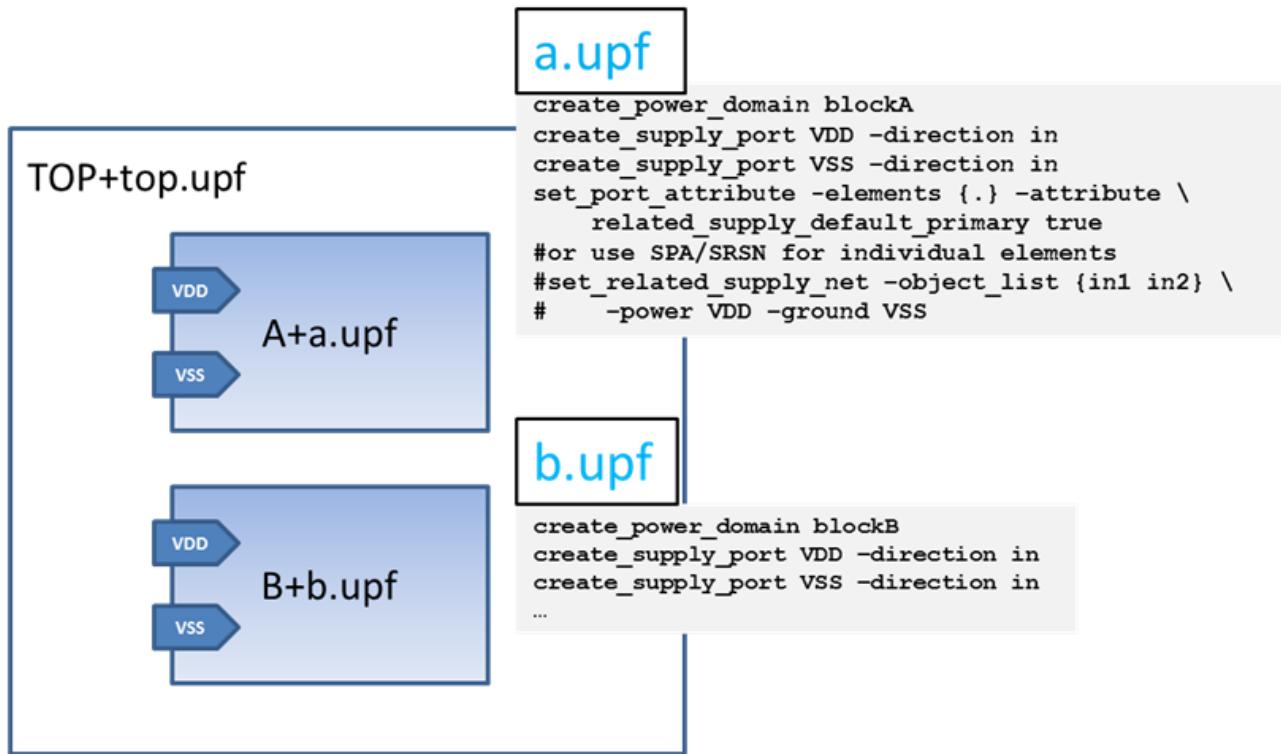
```
set_related_supply_net for ports
```

- ❖ In a LP design, these constraints must include the relationship between logic ports and their related supplies. This is necessary for correct corruption, and may be required for isolation policies.
- ❖ Each black box being compiled and verified should have its own UPF.
- ❖ Each block *design+UPF* must be self-contained, that is, all block ports should exist in design and control signals must be present as block ports/net inside the block (which is required for isolation, retention, switch control signals).

## Block-Level UPF Example

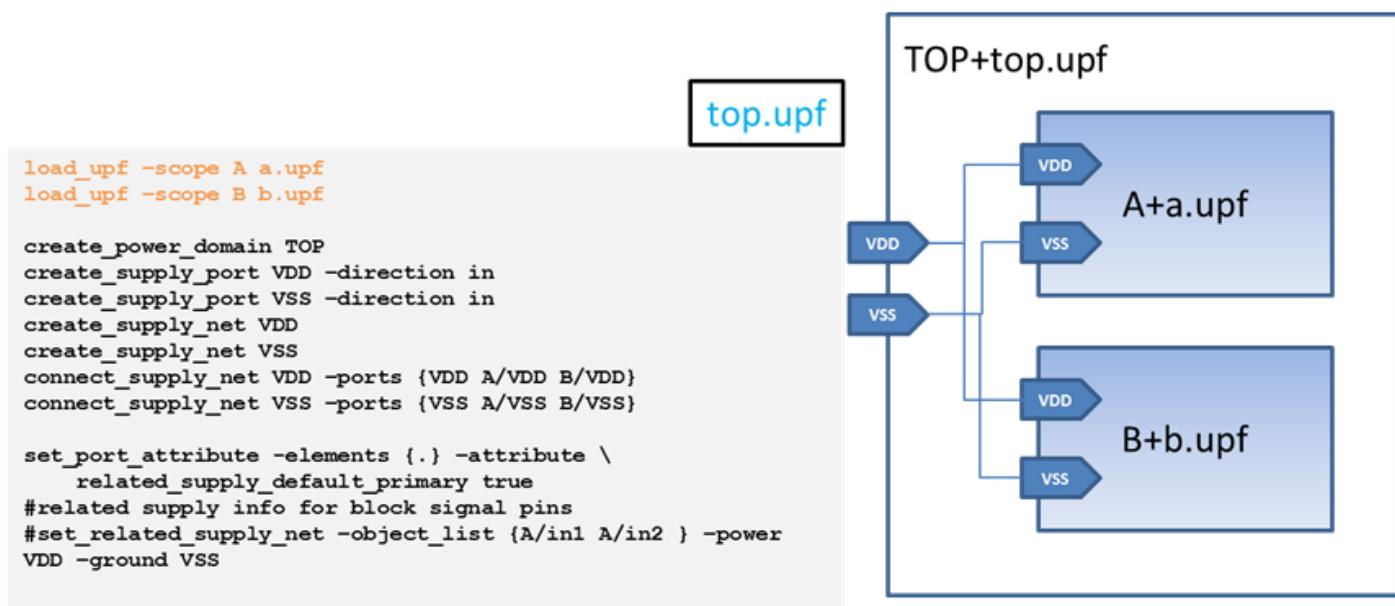
Block level boundary conditions can be in the form of SRSN or SPA.

Figure 6-1 Block-Level UPF Example



## Top-Level UPF Example

Top-level UPF should load block- level UPF into that scope.

**Figure 6-2 Top-Level UPF Example**

### 6.1.1.6 Detailed Black Box Flow

The following section describes what happens when the black box flow is enabled in VC LP.

#### 6.1.1.6.1 Loading Black Box UPF

To load a black box, use the following command:

```
load_upf Bbox.upf -scope Bbox
```

During Full chip verification, verified IP is defined as a black box and its UPF is loaded in Chip level UPF.

- ❖ Power Network at boundary of IP (black box) are preserved from IP UPF (black box- UPF)
- ❖ UPF commands with design reference inside IP (black box) are ignored from black box UPF.
- ❖ Inner-hierarchy command in black box UPF is also ignored.

#### 6.1.1.6.2 Power Domain

The `create_power_domain` command is honored if specified in the black box scope. A power domain construct is created at the black box scope. If the command refers to design elements of the nested black box scope, these nested elements are filtered out.

##### Examples

Top UPF snippet:

```
...
load_upf BBox.upf - scope BBOX
...
BB.upf snippet:
```

<code>create_power_domain bb_pd</code>	Applied
--	---------

<code>create_power_domain bb_pd -supply {primary ss1}</code>	Applied
<code>-update</code>	

create_power_domain <Nested_hierarchy>/power_domain -element {sub}	read & ignored
create_power_domain bb_pd -include_scope - element {sub/u1}	element sub/u1 is filtered out
set_domain_supply_net bb_pd -primary_power_net vdd1 \ -primary_ground_net vss1	Applied
set_domain_supply_net <Nested_hierarchy>/power_domain - primary_power_net sub/vdd_n1 - primary_ground_net sub/vss_n1	read & ignored

### 6.1.1.6.3 Supply Ports

The `create_supply_port` command is honored if specified in the black box scope. If specified, the supply port construct is created in the black box scope. Any `create_supply_port` command specified on nested black box scopes is ignored.

#### Examples

create_supply_port VDD -domain bb_pd	Applied
create_supply_port VDD_n1 -domain <Nested_hierarchy>/power_domain	read & ignored

### 6.1.1.6.4 Supply Sets and Supply Nets:

The `create_supply_set`, and `create_supply_net` commands are honored if specified in the black box scope. If specified, supply set or supply net constructs are created in the black box scope respectively. Any `supply_set`, or `supply_net` commands specified on nested black box scopes are ignored.

#### Examples

create_supply_net vdd1 -domain bb_pd	Applied
create_supply_net vdd_n1 -domain <Nested_hierarchy>/power_domain	read & ignored
create_supply_set ss1 -function {power vdd1} \ -function {ground vss1}	Applied
create_supply_set ss_n1 \ -function {power <Nested_hierarchy>/vdd_n1} \ -function {ground <Nested_hierarchy>/vss_n1}	read & ignored

### 6.1.1.6.5 Supply Connections

Explicit supply connections to existing supply ports or black box PG pins through `connect_supply_net` are honored. The `associate_supply_set` command is honored if all the supply objects are scoped at the black box scope.

**Examples**

connect_supply_net vdd1 -ports VDD	Applied
connect_supply_net -<Nested_hierarchy>/vdd_n1 -ports <Nested_hierarchy>/VDD_n1	read & ignored
associate_supply_set ss1 -handle bb_pd.primary	Applied
associate_supply_set ss_n1 -handle <Nested_hierarchy>/power_domain.primaryr	read & ignored

**6.1.1.6.6 Related Supplies for Ports**

The `set_port_attributes` command is allowed on black box scope. The `set_port_attributes -repeater_supply` is honored if the objects specified are at the black box scope. Otherwise, it is ignored. For `-receiver_supply / -driver_supply`, it is ignored by the implementation tools, but it is checked by verification tools.

The `set_related_supply_net` command is honored if the objects specified are at the black box scope. Otherwise, it is ignored.

**Examples**

set_port_attributes -elements {..} -applies_to inputs \ -repeater_supply rs1	Applied
set_port_attributes -elements {..} -applies_to inputs \ -driver_supply ss2 -receiver_supply ss1	read & ignored by implementation tools
set_related_supply_net -object_list [get_ports INPUT] \ -power vdd1 -receiver_supply vss1	Applied
set_related_supply_net -object_list [get_ports sub/INPUT] \ -power vdd_n1 -receiver_supply vss_n1	read & ignored

**6.1.1.6.7 Power Switches**

The `create_power_switch` command is honored at the black box scope. If specified, the power switch construct is created in the black box scope. Options refer to invisible objects will be ignored. Any power switch command specified on the nested black box domains is ignored.

**Examples**

create_power_switch sw1 -domain bb_pd \ -output_supply_port {vout VO1} \ -input_supply_port {vin VI1} \ -control_port {ctrl_small ON1} \ -on_state {full_s vin {ctr_small}}	Applied
---	---------

---

```
create_power_switch sw_n1 -domain
<Nested_hierarchy>/power_domain\
-output_supply_port {vout_n VO_N1} \



---



```

### 6.1.1.6.8 Power State Tables

The `add_power_state`, `add_pst_state`, `add_port_state`, and `create_pst` power state commands are honored if it uses supplies at the black box scope. Otherwise, it will be ignored.

#### Examples

<pre>add_power_state bb_pd.primary -state HVp \ {-supply_expr {power == `{FULL_ON, 0.88, 0.90, 0.92}}}</pre>	Applied
<pre>add_power_state bb_pd.primary -state HPg \ {-supply_expr {ground == `{OFF}}}}</pre>	Applied

---

#### Command at Top scope

<pre>add_port_state BBOX/VDD -state {bb_on 0.88 0.90 0.92} \ -state {off_state off}</pre>	Applied
<pre>create_pst top_pst -supplies {VDD_TOP BBOX/VDD} Applied add_pst_state all_on -pst top_pst -state {top_on bb_on}</pre>	

---

### 6.1.1.6.9 Isolation Strategies

The location parent isolation strategies that are defined on the power domain scoped on black box are honored, all the others are ignored. For `set_isolation_control`, strategies other than location parent are ignored during top level verification. If an isolation strategy is honored, the corresponding `map_isolation` command is honored, or else if the isolation strategy is ignored, then the corresponding `map_isolation` is ignored.

#### Examples

<pre>set_isolation isol1 -domain bb_pd \ -elements special_port1 \ -isolation_supply_set iso_ss</pre>	iso1 will be read and stored
<pre>set_isolation_control isol1 -domain bb_pd \ -location parent \ -isolation_signal en \ -isolation_sense high</pre>	iso1 will be implemented at the top level
<pre>set_isolation isol2 -domain bb_pd \ -elements special_port2 \ -isolation_supply_set iso_ss</pre>	iso2 will be read and stored

---




---

set_isolation_control iso2 -domain bb_pd \ -location self \ -isolation_signal en\ -isolation_sense highi	iso2 will be ignored at the top level
set_isolation iso3 -domain bb_pd -elements sub/special_port1 \ -isolation_supply_set iso_ss	ignored at the top level
set_isolation iso4 -domain bb_pd \ -elements special_port3 \ -no_isolation	command will be honored

---

### Location parent policy at black box top scope is also allowed from TOP scope

set_isolation top_iso1 -domain BBOX/bb_pd \ -elements BBOX/special_port1 \ -isolation_supply_set BBOX/iso_ssi	so1 will be read and stored
set_isolation_control top_iso1 -domain BBOX/bb_pd \ -location parent \ -isolation_signal BBOX/en \ -isolation_sense high	iso 1 will be implemented at the top level

---

#### 6.1.1.6.10 Level-Shifter Strategies

The Level-shifter strategy (set\_level\_shifter) on black box is honored if -location parent is specified. Otherwise, it is ignored.

##### Examples

set_level_shifter ls1 -domain bb_pd \ -elements special_port1 \ -location parent	ls1 will be implemented at the top level
set_level_shifter ls2 -domain bb_pd \ -elements special_port2 \ -location self	ls2 will be ignored at the top level
set_level_shifter ls3 -domain bb_pd \ -elements special_port3 \ -no_shift	command will be honored

---

Location parent policy at black box top scope is also allowed from TOP scope.

set_level_shifter top_ls1 -domain BBOX/bb_pd \ -elements BBOX/special_port1 \ -location parent1	s1 will be implemented at the top level
---	---

---

#### 6.1.1.6.11 Retention Strategies

Retention strategies defined on black box are used for block level verification. They are ignored at the top level since they have no impact on top level verification.

### 6.1.1.6.12 Logic Port and Net

VC LP ignores the `create_logic_port` and `create_logic_net` commands define in black box UPF during top level verification.

### 6.1.1.7 Top Level Checks During Black Box Flow

#### 6.1.1.7.1 LP checks

During the top level verification, black box boundary ports become starting and ending point for crossovers. SPA (`set_port_attribute`) and SRSN can be used to constrain the supply for the black box ports. If SPA is mentioned for black box ports, electrical checks takes the `-receiver_supply` for input ports and the `-driver_supply` for output ports.

If SPA/SRSN is not mentioned, then black box domain supply is used for different LP checks.

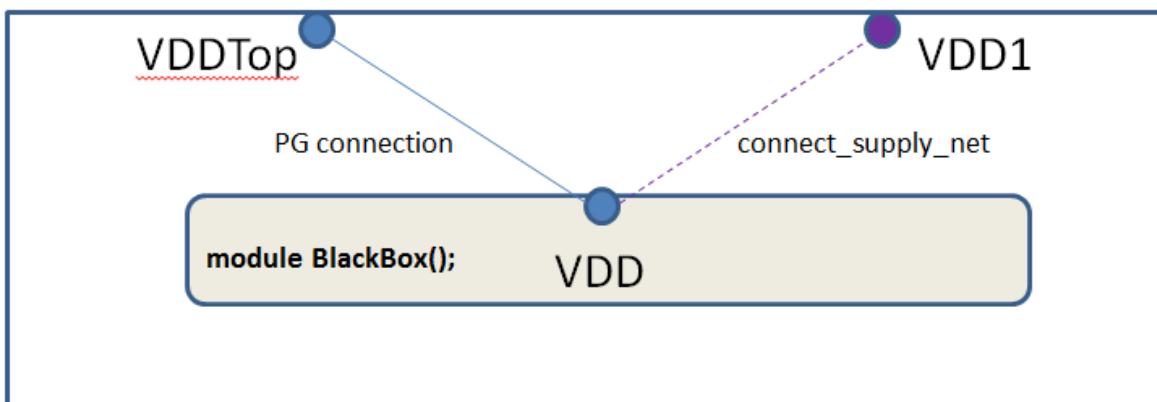
Black box boundary ports are not affected by `ignore_unconnected_*` or `handle_hanging_*` switches.

#### 6.1.1.7.2 PG checks

All the supply port of black box is considered as PG ports. All the PG checks are applicable on cells will be applicable to these ports of black box as well.

For any violation that are issued related to any black box pins, VC LP reports the debug field `BlackBoxScope : True`.

**Figure 6-3 PG Checks**



#### Example

```

Tag          : PG_CSN_CONN
Description   : UPF connect_supply_net [UPFSupply] does not match design supply net
[DesignNet] on pin [CellPin] of [CellType] instance [Instance] ([Cell])
UPFSupply    : VDD1
DesignNet
  NetName     : VDDTop
  NetType      : Design/UPF
CellPin       : VDD
Instance      : i_BlackBox
Cell          : BlackBox
BlackBoxScope : True

```

### 6.1.1.8 Black Box Boundary Ports Constraint Checks

#### 6.1.1.8.1 UPF Parsing Checks

When you create a black box at the block level design using the `set_blackbox -designs/cell {..}` command or undefined module or stub module, VC LP expects supply constraints (SPA/SRSN) for every non-PG ports of that black boxed instances. If no supply constraints are specified in those ports, VC LP reports the following warning message:

*[Warning] USERBBOX\_PORT\_UNCONSTRAINED: User Black box port has no supply constraint*

#### 6.1.1.8.2 SPA Consistency Checks on Black Box Ports

The SPA consistency checks are performed at all the black box boundary ports for the supply sets specified in the `set_port_attribute -driver_supply / -receiver_supply` command in UPF.

- ❖ UPF\_SPADRIVER\_STATE: This violation is reported when the SPA driver\_supply is on, but the supply of actual driver is off for the SPA constraint on the black box port.
- ❖ UPF\_SPARECEIVER\_STATE: This violation is reported when the supply of the actual receiver is on, but the SPA receiver\_supply is off for the SPA constraint on the blackbox port.
- ❖ UPF\_SPASUPPLY\_VOLTAGE: This violation is reported when Voltage difference between supply of actual driver/receiver and constrained supply for the SPA constraint on the black box port.
- ❖ UPF\_SPASUPPLY\_CONN: This violation is reported when the supply of the actual driver/receiver is not connected to the constrained supply for the SPA constraint on the black box port.
- ❖ UPF\_SPASUPPLY\_INTERNAL: Internal supplies may be part of a lower level block, and used for SPA which are applied at both block and top level. This tag is reported for the following scenarios:
  - ◆ For block level, the violation is reported on SPA where `driver_supply` or `receiver_supply` is an internal supply\_net with no connection to upper supply\_port.
  - ◆ For top level, the violation is reported on SPA where block LowConn is not accessible at top level.
- ❖ UPF\_SPA\_SUPPLY: This violation is reported when a SPA node is connected to a supply net. SPA nodes connected to supply net might impact runtime of SPA consistency checks (UPF\_SPASUPPLY\_CONN, UPF\_SPASUPPLY\_VOLTAGE, UPF\_SPADRIVER\_STATE, UPF\_SPARECEIVER\_STATE) as it can lead to huge PG network traversals.

The `set_port_attributes` are ignored completely on PG pins, supply ports and logic pins connected to supply nets. The UPF\_SPA\_SUPPLY violation reports this information along with the supply resolution details. The UPF\_SPA\_SUPPLY violation is also reported when the following pins/ports are listed in the SPA attribute.

- ◆ PG ports
- ◆ PG pins of lib cell
- ◆ CSN/PG connected design ports
- ◆ CSN/PG connected libcell data pins

There is priority in SPA consistency checks, \*\_STATE is checked first, if no violations are found, then \*\_VOLTAGE is checked, if no violations are found in these checks as well, then the \*\_CONN checks are performed.

### Note

If `set_port_attributes -driver_supply/receiver_supply` is set on hierarchical module port with power domain but without scope of its own, the setting would be ignored and the `UPF_SPAIGNORED_ON_INVSCOPE` warning message is reported.

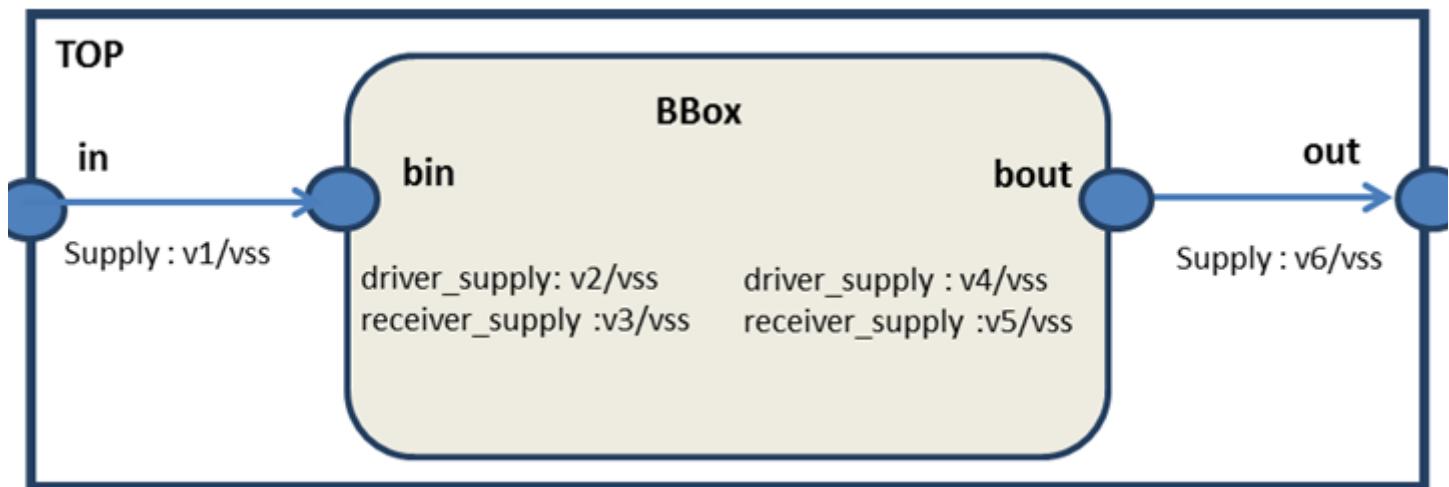
The SPA consistency checks also check for PST equivalence, and if there is any PST mismatch between the driver and receiver, then violation `UPF_SPASUPPLY_CONN` is reported. For the PST equivalent checks, the `PstEquiv` debug field is reported in existing violation `UPF_SPASUPPLY_CONN` which indicates if the PST is the same between the two supplies. These violations are reported at the `check_lp -stage upf` or `check_lp -stage upf -family { upfconsistency }`

### Note

These consistency checks are not performed when SRSN is applied.

#### Example 1

**Figure 6-4 SPA (set\_port\_attribute) Consistency Checks on Black Box Ports**



For the example in [Figure 6-4](#), the driver and receiver supply consistency checks details are provided in [Table 6-1](#) and [Table 6-2](#).

**Table 6-1 Driver Supply Consistency Checks**

Actual driver	PST Equivalent	SPA -driver_supply	Violation Name
v1 (off)	Not equal (!=)	v2 (more ON with v1)	UPF_SPADRIVER_STATE
v1 (1.0)	Not equal (!=)	v2 (> or < v1, 08 or 1.0)	UPF_SPASUPPLY_VOLTAGE
v1 (1.0)	Equal (==)	v2 (1.0)	UPF_SPASUPPLY_CONN with PstEqui : True
v1 (more ON with v2)	Not equal (!=)	v2 (off)	UPF_SPASUPPLY_CONN with PstEqui : False

**Table 6-2 Receiver Supply Consistency Checks**

<b>SPA -receiver_supply</b>	<b>PST equivalent</b>	<b>Actual Receiver</b>	<b>Violation Name</b>
v5 (off)	Not equal (!=)	v6 (more ON with v1)	UPF_SPARECEIVER_STATE
v5 (1.0)	Not equal (!=)	v6 (> or < v5, 08 or 1.0)	UPF_SPASUPPLY_VOLTAGE
v5 (1.0)	Equal (==)	v6 (1.0)	UPF_SPASUPPLY_CONN with PstEqui : True
v5 (more ON with v6)	Not equal (!=)	v6 (off)	UPF_SPASUPPLY_CONN with PstEqui : False

### 6.1.1.9 The lp\_limit\_spacheck Application Variable

By default, the UPF\_SPA\* checks were performed on the TOP module, black box modules, macro cells, power\_scope instances and terminal boundaries. The `lp_limit_spacheck` application variable is introduced to limit the UPF\_SPA\* checks only on TOP and terminal boundaries.

The `lp_limit_spacheck` application variable makes the behavior compatible with DC, as DC ignores SPA written on hierarchical ports if terminal boundary is not set on the instantiated module/entity.

By default, the `lp_limit_spacheck` application variable is set to false. When you set the `lp_limit_spacheck` application variable to true, the following checks are performed on TOP and terminal boundaries:

- ❖ UPF\_SPADRIVER\_STATE
- ❖ UPF\_SPARECEIVER\_STATE
- ❖ UPF\_SPASUPPLY\_VOLTAGE
- ❖ UPF\_SPASUPPLY\_CONN
- ❖ UPF\_SPAINOUT\_STATE
- ❖ UPF\_SPAINOUT\_VOLTAGE
- ❖ UPF\_SPASUPPLY\_MISSING
- ❖ UPF\_SPASUPPLY\_MISMATCH

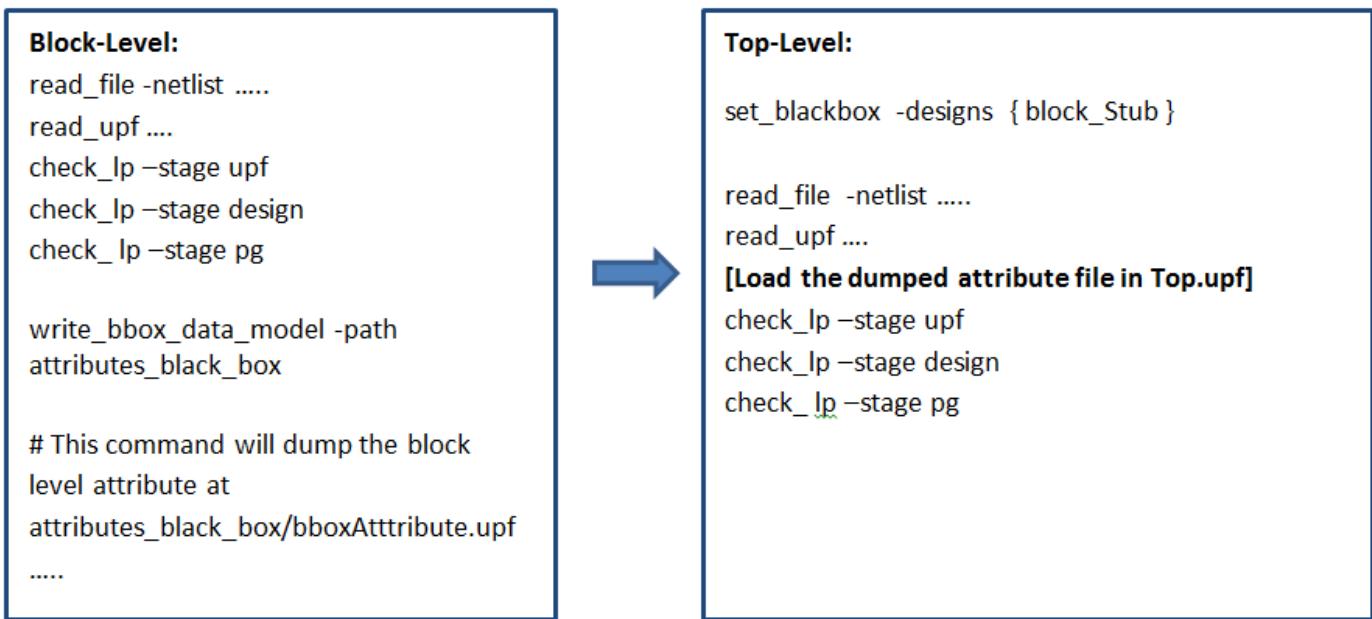


#### Note

When the `lp_limit_spacheck` application variable is set to true, the UPF\_SPA\_HIER violation is not reported, as the UPF\_SPA\_HIER violation checks if the SPA is set on a hierarchical ports whose instance is neither TOP, nor power\_domain elements, nor power\_scope instances.

### 6.1.1.10 Enhanced Black Box Flow (Attributes Dumping, Saving and Retrieving)

To perform relevant and correct checks at the top level, specific design characteristics need to be captured from block level runs and use them at the top level. These design characteristics of the blocks (which are black boxed at top level) are captured using `set_port_attributes` and dumped as attributes which can then be used for top-level runs. These attributes could be specified by the user or tool can generate it from block level analysis.

**Figure 6-5 Enhanced Black Box Flow Example**

### 6.1.1.10.1 Types of Attribute

#### Constant Attributes:

This attribute captures the list of black box ports that are driven by constants (logic-const/tie-high/tie-low/supply0/supply1)

#### Syntax

```
set_port_attribute -model <bbox_name> -ports {<bbox_port_name>}
-attribute SNPS_BLK_MODEL_TIED_VALUE {<1/0>}
-attribute SNPS_BLK_MODEL_TIED_NAME {hier_name_of_const}
```

#### Use Model

```
set_port_attributes -model core1 -ports {out1} \
-attribute SNPS_BLK_MODEL_TIED_VALUE {0} \
-attribute SNPS_BLK_MODEL_TIED_NAME {top/blackbox/LOGIC1}
```

#### Unconnected Attributes:

This attribute captures a list of ports of a black box model that are not connected to any internal logic. If such a port is an input port, it means there is no logic within the model driven by the port; if such a port is an output port, it means there is no logic within the model driving the port.

#### Syntax

```
set_port_attribute -model <bbox_name> -ports {<bbox_port_name>}
-attribute SNPS_BLK_MODEL_UNCONNECTED {<1/0>}
-attribute SNPS_BLK_MODEL_UNCONNECTED_NAME {hier_name}
```

#### Example

```
set_port_attributes -model bbox_top -ports {out2} \
-attribute SNPS_BLK_MODEL_UNCONNECTED {true} \
```



```
-attribute SNPS_BLK_MODEL_UNCONNECTED_NAME {sInst22/out}
```

### Isolation Enable Attributes:

This attribute captures the list isolation enable ports.

#### Syntax

```
set_port_attributes -model <bbox_name> -ports {<bbox_port_name>} \
-attribute SNPS_BLK_MODEL_DEVICE_PORT {ISOLATION device port name } \
-attribute SNPS_BLK_MODEL_DEVICE_CLAMP { <device clamp value>} \
-attribute SNPS_BLK_MODEL_DEVICE_POLICY {<Iso policy at block>} \
-attribute SNPS_BLK_MODEL_POLICY_CLAMP {<Iso policy clamp value at block>}
```

#### Example

```
set_port_attributes -model bbox_top -ports {iso} \
-attribute SNPS_BLK_MODEL_DEVICE_PORT {iso6/B} \
-attribute SNPS_BLK_MODEL_DEVICE_CLAMP {LOGIC_0} \
-attribute SNPS_BLK_MODEL_DEVICE_POLICY {PD_SUB/isoP2} \
-attribute SNPS_BLK_MODEL_POLICY_CLAMP {LOGIC_0}
```

### Saving Attributes

The `write_bbox_data_model` command saves the constant attributes (logic-const, tie-high, tie-low, supply0, supply1), unconnected attributes, isolation enable attributes from the block level run.

The `write_bbox_data_model` also saves the SPA -feedthrough attributes from primary inputs to primary outputs in the blackbox flow. The -feedthrough attributes saved from a block level run, can be used in the top level run. The -feedthrough is saved only when there is no logic between top input port and top output port.

You can write black box port attributes during block level run using the following command.

```
%vc_static_shell> write_bbox_data_model [-path <directory_name>]
```

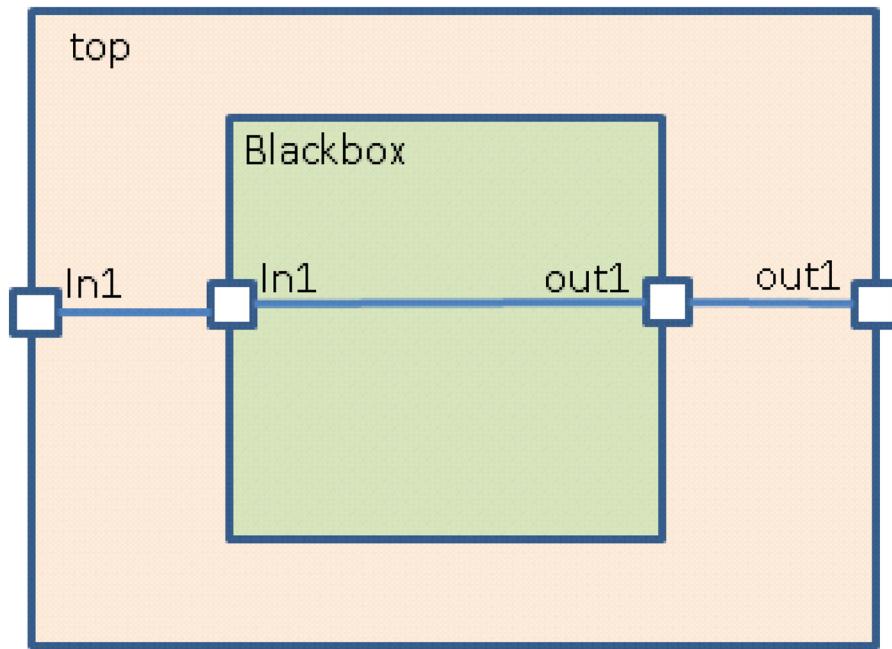
Where, -path is optional;

- ❖ If specified, a directory will be created with this name
- ❖ If not specified, a directory will be created with name \$(design\_top)\_bbox\_model in current working area.
- ❖ It will dump the attributes in a file called bboxAttribute.upf and saves into the specified directory or in \$(design\_top)\_bbox\_model in the current working area.

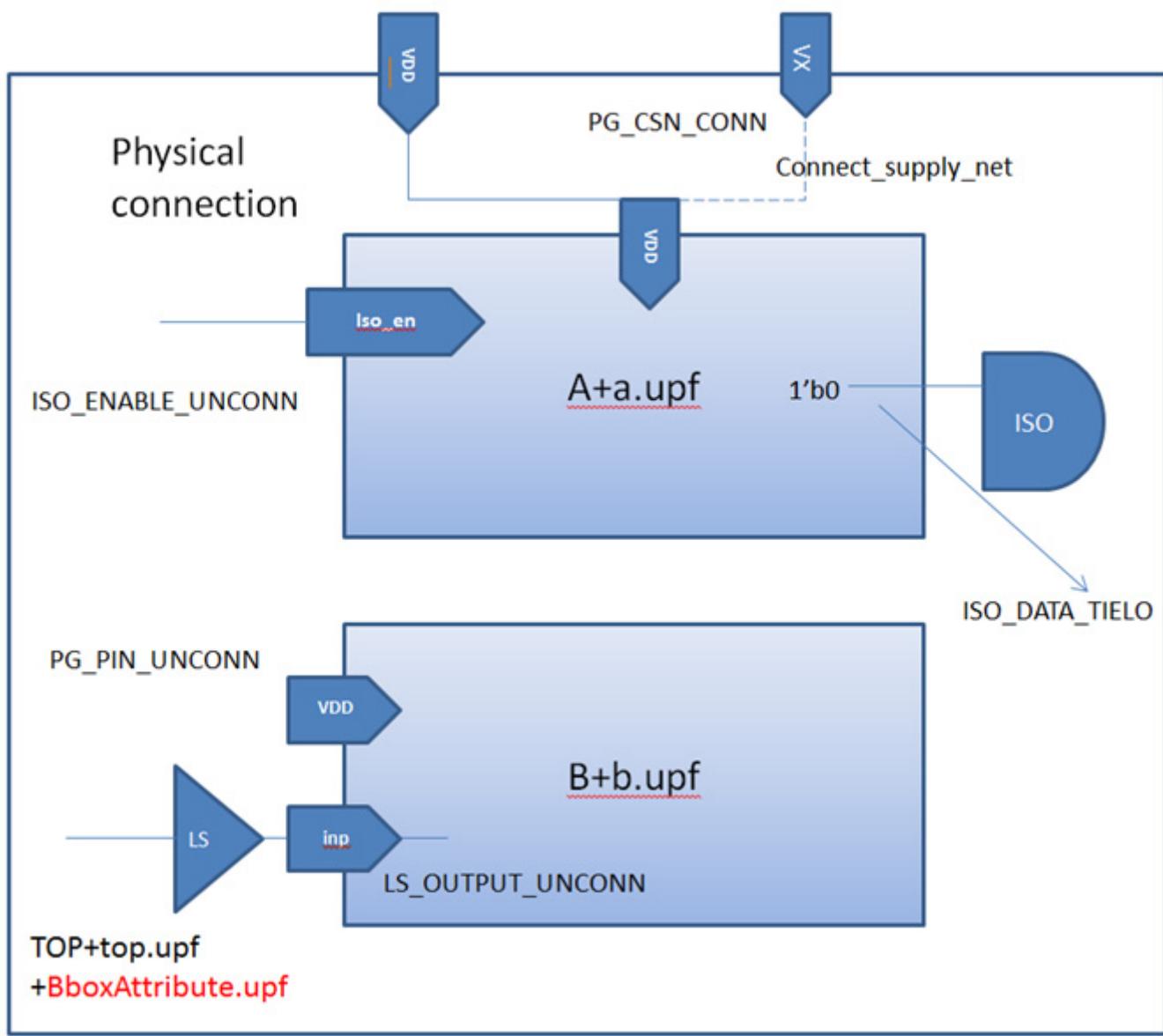
You must explicitly mention above command to write out these attributes to text files. You can also edit these files to add/remove/update specific attributes.

For the feedthrough paths shown in the example in [Figure 6-6](#), the `write_bbox_data_model` saves the following SPA in `bboxAttribute.upf`.

```
set_port_attributes -feedthrough -model Blackbox -ports {in1 out1}
```

**Figure 6-6 Example for -feedthrough paths****Note**

It is your responsibility to load this file in top.upf for top level runs.

**Figure 6-7 Enhanced Top level Verification (Illustration)**

### 6.1.1.11 Black Box Flow Tcl Support

#### 6.1.1.11.1 The get\_blackbox Command

Use the `get_blackbox` command to get a collection of black-boxed elements in the design.

##### Syntax

```
%vc_static_shell> get_blackbox -help
Usage: get_blackbox      # Returns list of objects which are listed as blackbox
      [-designs]          (Return only designs specified using -designs in
set_blackbox)
      [-automatic]        (Return only automatically generated blackbox)
      [-unresolved]        (Return only unresolved modules in the design)
```

The `get_blackbox` must be used after loading the design.

- ❖ By default, the `get_blackbox` command returns all the black box cells in the design.
- ❖ If `-designs` is specified, the `get_blackbox` command returns the user defined black box designs.
- ❖ The automatic means tool black box. For example, the module is empty or the module is unresolved. The automatic black box doesn't include the user black box.

## Use Model

```
%vc_static_shell>get_blackbox  
{ "chipCore_aop/pd_aop", "chipCore_aop/pd_dft_aop" }
```

### 6.1.1.12 The report\_blackbox Command

Use the `report_blackbox` command to get a report of the black box objects in the design. The report is printed on the screen. The report is divided into separate sections, for black-boxed cells (if any), black-boxed designs (if any), black-boxed libcells (if any), and automatic black-boxed items during design read.

#### Syntax

```
%vc_static_shell> report_blackbox  
Usage: report_blackbox      # Returns a report of blackboxed objects in the design  
       [-designs]           (Report only designs specified using -designs in  
       set_blackbox)  
       [-automatic]         (Report only automatically generated blackbox)
```

#### Use Model 1

```
%vc_static_shell> report_blackbox  
BlackBox Report  
Cells:  
Cell: chipCore_aop/pd_aop  
Cell: chipCore_aop/pd_dft_aop
```

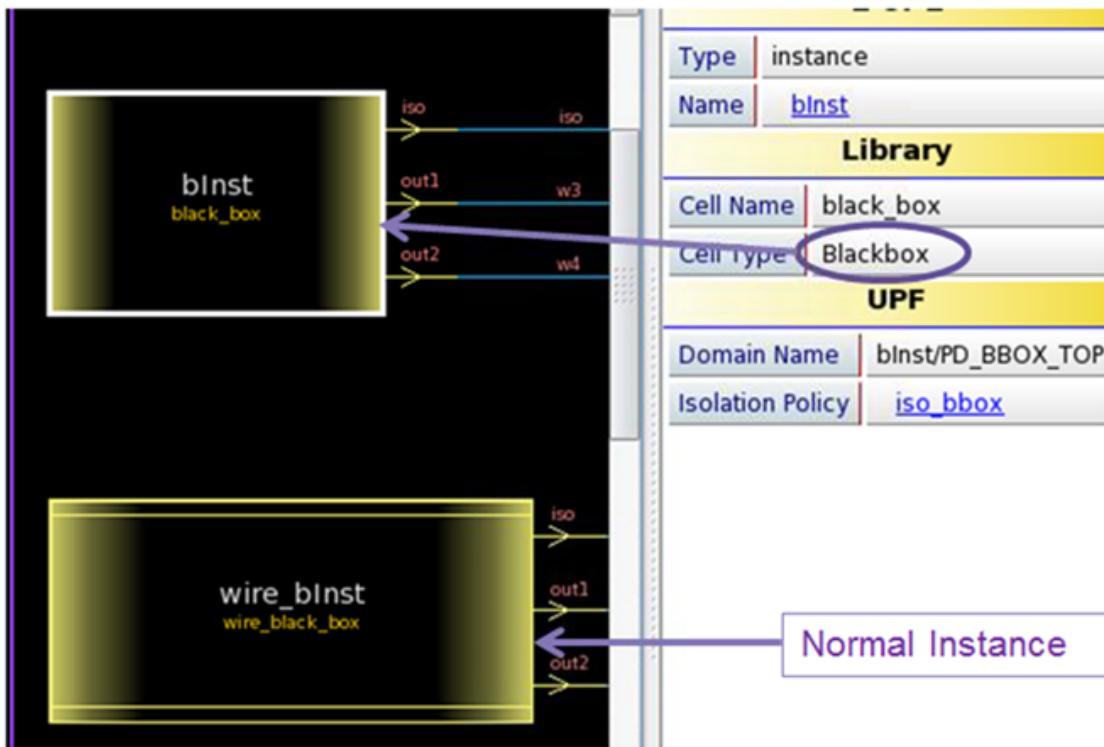
#### Use Model 2

```
%vc_static_shell> report_blackbox -designs  
BlackBox Report  
Designs:  
Design: pd_aop  
Design: pd_dft_aop
```

### 6.1.1.12 Black Box Flow GUI Support

The black box module does not have two horizontal lines, which indicates that the module is a black box and celltype is mentioned as black box.

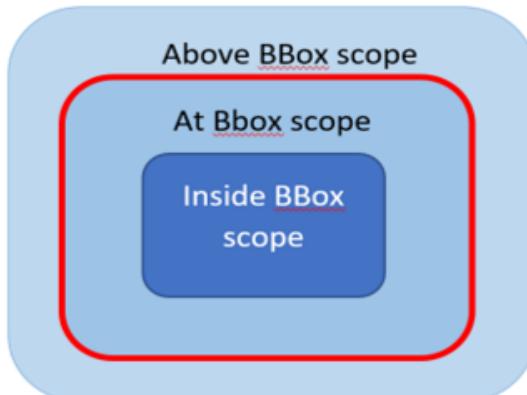
The normal instance has two horizontal lines which indicate there is logic inside.



### 6.1.1.13 Messages and Debug help

All the UPF commands above BBOX/ETM scope will be processed.

If these UPF commands above BBOX/ETM scopes refer to elements or objects are inside bbox/ETM scope, those commands are neglected and proceeded. **Warning - UPF\_WITHIN\_BBOX\_ACCESS\_WARN, Warning - UPF\_EMPTY\_OBJ\_LIST\_BBOX, Warning - UPF\_EMPTY\_OBJ\_LIST\_MIXED** are flagged depending on the scenario



All the UPF commands inside BBOX scope will be ignored

All the UPF commands AT BBOX/ETM scope will be processed.

Commands related to power network connection (create\_supply\_net, create\_supply\_port are accepted if the port/net is AT BBOX (interface of BBOX) scope.

Strategies (iso strategies, LS strategies etc....) where the --location is defined as parent are processed. If "--location self" is given that need to be ignored.

UPF commands with references to object inside bbox/ETM are ignored. **Warning - UPF\_CMD\_IGNORED\_BBOX** is be flagged.

### 6.1.1.14 Limitations

- ❖ The `set_blackbox` command is not allowed after design read.
- ❖ RTL (SIMON) Vs Tcl (VNR) flow differs for unresolved and stub models.
- ❖ Extracting attributes and propagating them across hierarchical black box (black box inside a black box) is not supported.
- ❖ The `map_isolation/level_shifter` commands are ignored even though location parent policies are not ignored.
- ❖ The unconnected attribute dumping is not happening properly for input ports.
- ❖ VC LP adds BB\_CELL\_ as prefix with Cell name (module name) of black box instance.
- ❖ The escape name in black box instance name is not treated as Blackbox\_instance.
- ❖ Wildcard with "?" in "set\_blackbox -cells" command gives an error.
- ❖ For Mx Design, hierarchical instance name is not supported in "set\_blackbox -cells " " ".
- ❖ For Instance designed using generate block, Instance is not treated as black\_box with "set\_blackbox -cells".

## 6.1.2 ETM Based UPF Hierarchical Flow

Extending the support for hierarchical low power verification flows, VC LP support ETM+UPF flow in addition to black box or stub flows where you can do top-only verification using various abstracted models of individual blocks. ETM (Extracted timing model) is a liberty abstracted model for a hard macro or block that has been implemented separately during the hierarchical implementation flows. ETM models are for the blocks that are used for top level integration and implementation.

If you are using UPF, you may want to use ETM as an abstraction model in UPF hierarchical flow to complete the designs. ETM started as a soft IP and becomes hardened in the flow. For ease-of-use, you can apply the same UPF file to the soft IP (block design) and to the ETM extracted from the block design. During ETM extraction, it is desirable to capture possible UPF information into the ETM .lib. When ETM is used as an abstraction model, VC LP takes MV/UPF information from the ETM .lib and takes the other part of needed UPF information from the ETM UPF. VC LP provides the capability of replacing the ETM model abstractions throughout the UPF hierarchical flow.

### 6.1.2.1 Enabling ETM+UPF Flow

There is no specific setting or Tcl variable required. VC LP allows loading on UPF for ETM/hard macro cells as long the liberty cell has `is_macro_cell:true` attribute.

UPF constructs can be created and referenced at top-most scope of ETM/hard macro module.

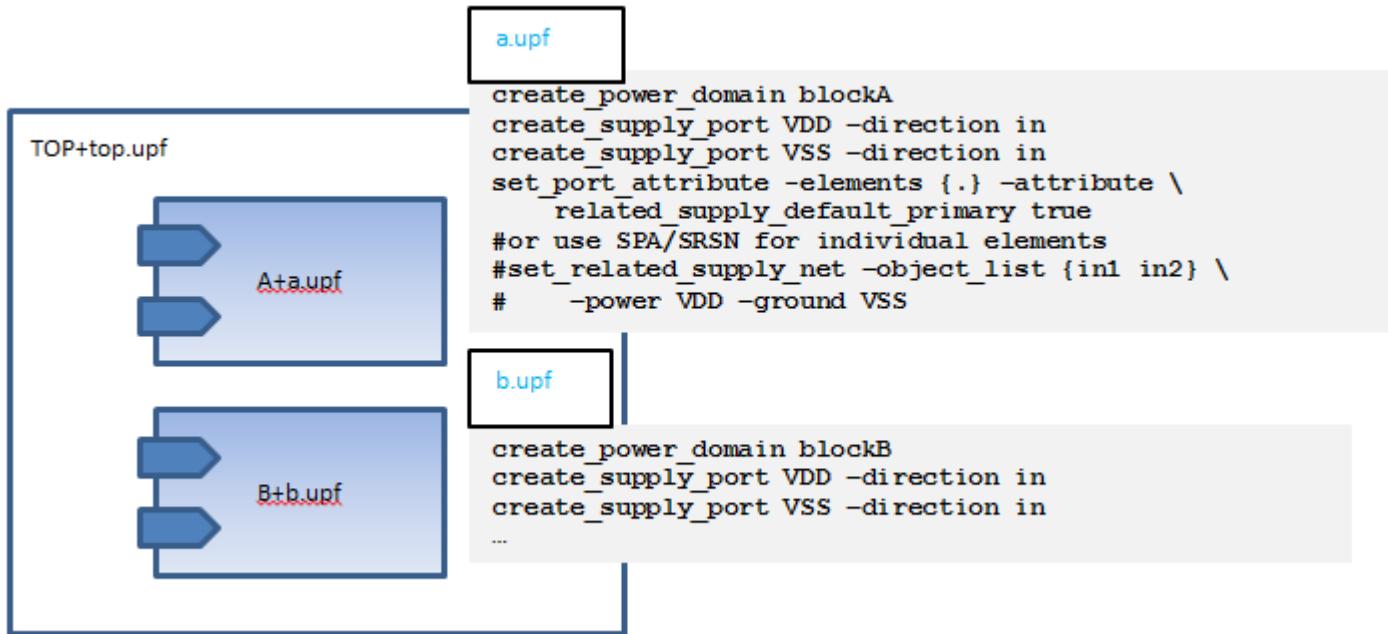
For example,

```
load_upf block.upf -scope ublock
```

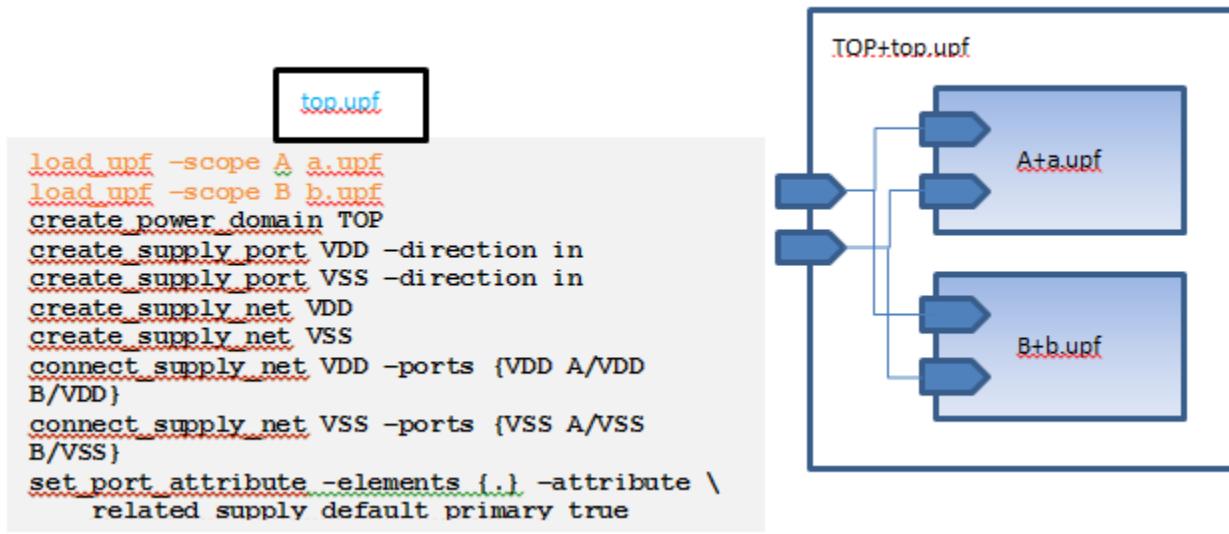
Where ublock is modeled as ETM

### 6.1.2.2 Top level LP verification using ETM+UPF flow

**Figure 6-8** ETM Flow - Block Level UPF Example



**Figure 6-9** ETM Flow - Top Level UPF Example



The following are the UPF requirements for an accurate (safe) hierarchical verification methodology:

- ❖ The external boundary constraints that impact the functionality of the level being verified must be specified.
- ❖ These constraints must be visible and validated when verifying the parent level or the child level

```
set_port_attribute -driver_supply/-receiver_supply
or
set_related_supply_net for ports
```

In an LP design, these constraints must include the relationship between logic ports and their related supplies. The ETM liberty model models this information and the associated UPF for the block has additional information about these constraints such as power state table (PST).

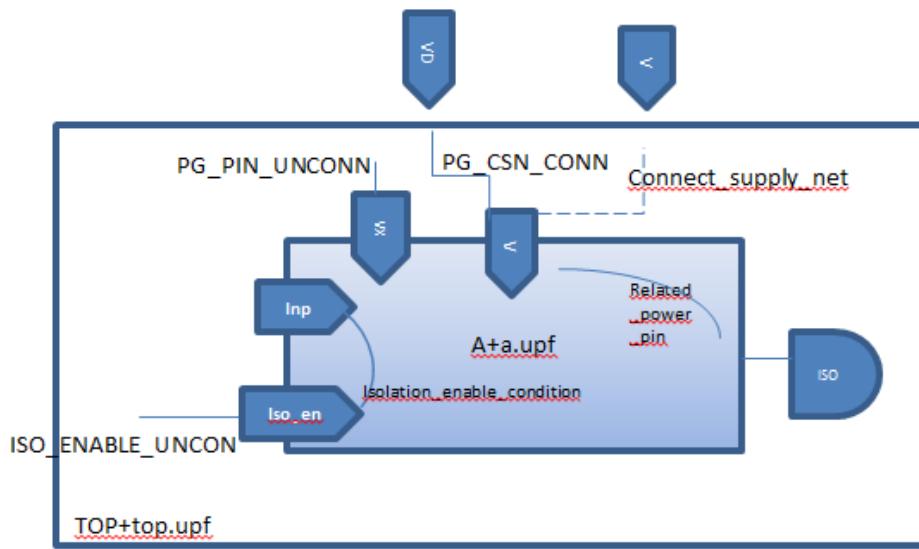
When a block level ETM UPF is loaded for the scope of ETM or hard macro for doing top level verification, VC LP:

- ❖ Automatically ignores block UPF constructs that are 'inside the scope of ETM' with a warning message  
*Example: nested scope, load\_upf inside ETM upf*
- ❖ Preserves the UPF constructs of the block UPF which are at the scope of ETM boundary. These will be used for top level verification.  
*Example: support ports, supply nets, switches, PST*

With the preserved ETM UPF, VC LP does necessary checks for top-only verification by performing the following checks:

- ❖ Driver/receiver Supply consistency checks at ETM logic boundary ports/pins
- ❖ PG pin consistency checks for the ETM pg\_pins
- ❖ Crossover and other signoff checks for the ETM logic boundary ports/pins

**Figure 6-10 ETM UPF Requirement**



### 6.1.2.3 Power Domain

The `create_power_domain` command is honored if it is specified at the top-most ETM scope. A power domain construct is created at the top-most ETM scope. If the command refers to design elements from the nested ETM scope, these nested elements are filtered out. If all the elements come from the nested ETM scope, the whole `create_power_domain` command is ignored.

## Examples

```
set_scope myetm
```

create_power_domain etm_pd	applied
create_power_domain etm_pd -supply {primary ss1}-update	applied
create_power_domain etm_pd_nested -element {sub}	(read & ignored)
create_power_domain etm_pd -include_scope -element {sub/u1}	element sub/u1 is filtered out)
set_domain_supply_net etm_pd - primary_power_net vdd1 \ -primary_ground_net vss1	applied
set_domain_supply_net etm_pd_nested - primary_power_net sub/vdd_n1 - primary_ground_net sub/vss_n1	read & ignored

### 6.1.2.4 Supply Ports

The `create_supply_port` command is honored if it is specified in the top-most ETM scope. If specified, the supply port construct is created in the top-most ETM scope. Any `supply_port` command specified in the nested ETM scope is ignored.

#### Example

```
create_supply_port VDD -domain etm_pd (applied)
create_supply_port VDD_n1 -domain etm_pd_nested (read & ignored)
```

When the `create_supply_port` command is executed, the port is checked against the PG pins of the cell. If the port matches to a particular PG pin, the command is executed successfully and the port is created. If the port does not match to any pin, then the below error is reported.

```
UPF_PORT_EXTRA
Description : UPF supply port [SupplyPort] defined but does not
              exist in cell [CellType]
SupplyPort  : V3
Instance    : u1
CellType    : my_macro
```

### 6.1.2.5 Supply Sets and Supply Nets:

The `create_supply_set` and `create_supply_net` commands are honored if it is specified at the top-most ETM scope. If specified, supply set or supply net construct is created at the top-most ETM scope respectively. Any `supply_set`, or `supply_net` commands specified on the nested ETM scopes are ignored.

#### Examples

create_power_domain etm_pd	(applied)
create_supply_net vdd1 -domain etm_pd	applied

```
create_supply_net vdd_n1 -domain etm_pd_nested (read & ignored)
```

```
create_supply_set ss1 -function {power vdd1}\- (applied)
function {ground vss1}
```

```
create_supply_set ss_n1 -function {power vdd_n1}\- (read & ignored)
function {ground vss_n1}
```

### 6.1.2.5.1 Supply Connections

Explicit supply connections to the existing supply ports or ETM PG pins through `connect_supply_net` are honored. The `associate_supply_set` command is honored if all the supply objects are scoped at the top-most ETM scope.

#### Examples

connect_supply_net vdd1 -ports VDD	applied
connect_supply_net vdd_n1 -ports VDD_n1	read & ignored
associate_supply_set ss1 -handle etm_pd.primary	applied
associate_supply_set ss_n1 -handle etm_pd_nested.primary	read & ignored)

By default, when multiple `connect_supply_net` are set on one library cell pg pin, VC LP would take the first CSN, ignore the later ones, and report UPF\_SUPPLY\_PORT\_ALREADY\_CONNECTED on the ignored CSN at `read_upf` parsing stage.

When the `lp_allow_pg_pin_reconnection` set to true, VC LP would take the last CSN, ignore the previous ones, and report UPF\_CONNECT\_SUPPLY\_NET\_OVERWRITE on the ignored CSN at `read_upf` parsing stage.

### 6.1.2.6 Related Supplies for Ports

The `set_port_attributes` command is executed on ETMs/Macros. If SPA on the top level inout port is specified and if it has different supplies for `.driver_supply` and `receiver_supply`, the following ERROR is reported:

*[Error] UPF\_SPA\_INCONSISTENT: Driver supply and receiver supply, specified on inout port, are not same.*

Driver supply 'TOP\_ss' and receiver supply 'SS', specified on inout port 'io1', are not the same.

If SPA on the top level inout port is specified and if the `driver_supply` and `receiver_supply` are found to be the same supply net/set, then that supply overrides the default domain supply of the inout port. The SPA supply is used for all protection and electrical checks on crossovers starting or ending at the inout port of the DUT top.

The `-receiver_supply / -driver_supply` commands are checked.

#### Examples

`set_port_attributes -ports {port_list}` needs to be used for applying SPA driver/receiver\_supply on inout port.

`set_port_attributes -elements { . } -applies_to { inputs | outputs | both }` does not apply SPA on inout ports

### Example

```
set inout_port_list [find_objects . -pattern * -object_type port -direction inout ]
set_port_attributes -ports $inout_port_list -driver_supply <supply_set1> -
receiver_supply <supply_set1>
```

These commands apply `set_port_attribute driver/receiver supply` on top level ports with inout direction.

If SRSN on the top level inout port is specified, it overrides the default domain supply on the inout port. The SRSN supply will be used for all protection and electrical checks on crossovers starting or ending at the inout port of the DUT top.

It is a known DC compatibility issue that when a top level inout port has SPA or SRSN and it is honored, VC LP uses the overriding SRSN/SPA supply for path based policy association, whereas DC does not.

<code>set_port_attributes -elements { . } -applies_to inputs \ -repeater_supply rs1</code>	applied
<code>set_port_attributes -elements { . } -applies_to inputs \ -driver_supply ss2 -receiver_supply ss1</code>	read & ignored by implementation tools
<code>set_related_supply_net -object_list [get_ports INPUT] \ -power vdd1 -receiver_supply vss1</code>	applied
<code>set_port_attributes -ports { clk rst } -related_power_port VDD_AON -related_ground_port VSS</code>	<p>read and the following warning message is reported</p> <p>&gt;[Warning] UPF_SPA_IGN_OPT: SPA option ignored Option '-related_power_port' specified in command 'set_port_attributes' is not supported.</p> <p>[Warning] UPF_SPA_IGN_OPT: SPA option ignored Option '-related_ground_port' specified in command 'set_port_attributes' is not supported.</p>

---

The `set_related_supply_net` command is honored if the objects are specified at the top-most ETM scope. Else, it is ignored.

The `set_port_attributes -related_power/related_ground_port` command is read, and a warning message is reported, if these attributes are provided.

```
set_port_attributes -ports { clk rst } -related_power_port VDD_AON -related_ground_port
VSS
```

[Warning] UPF\_SPA\_IGN\_OPT: SPA option ignored Option '-related\_power\_port' specified in command 'set\_port\_attributes' is not supported.

[Warning] UPF\_SPA\_IGN\_OPT: SPA option ignored Option '-related\_ground\_port' specified in command 'set\_port\_attributes' is not supported.

### 6.1.2.7 Power Switches

The create\_power\_switch command is honored at the top-most ETM scope. If specified, the power switch construct is created at the top-most ETM scope. Options that refer to invisible objects are ignored. Any power switch command specified on the nested ETM domains is ignored.

#### Examples

```
create_power_switch sw1 -domain etm_pd \ - applied
output_supply_port {vout V01} \ -
input_supply_port {vin VII} \ -control_port
{ctrl_small ON1} \ -on_state {full_s vin
{ctr_small}}
```

```
create_power_switch sw_n1 -domain read & ignored
etm_pd_nested \ -output_supply_port {vout_n
VO_N1} \ -input_supply_port {vin_n VI_N1} \ -
control_port {ctrl_small_n ON_N1} \ -on_state
{full_s vin_n {ctr_small_n}}
```

### 6.1.2.8 Power State Tables

The add\_power\_state, add\_pst\_state, add\_port\_state, and create\_pst power state commands are honored if it uses supplies at the top-most ETM scope. Else, it is ignored.

#### Examples

```
add_power_state etm_pd.primary -state HVp \ {- applied
supply_expr {power == `{{FULL_ON, 0.88, 0.90,
0.92}}}}
```

```
add_power_state etm_pd.primary -state HPg \ {- applied
supply_expr {ground == `{{OFF}}}}
```

```
set_scope ..add_port_state myetm/VDD -state applied
{etm_on 0.88 0.90 0.92} \ -state {off_state
off}
```

```
create_pst top_pst -supplies {VDD_TOP applied
myetm/VDD}
```

```
add_pst_state all_on -pst top_pst -state applied
{top_on etm_on}
```

### 6.1.2.9 Isolation Strategies

The location parent isolation strategies may be defined on the power domain scoped on ETM. The `set_isolation` command is honored if the domain, isolation signals and supplies are scoped at the top-most ETM scope. Else, it is ignored. For `set_isolation_control`, strategies other than location parent are ignored during top level optimization. If an isolation strategy is honored, the corresponding `map_isolation` command is honored, else the corresponding `map_isolation` command is ignored.

#### Examples,

<code>set_isolation iso1 -domain etm_pd \-elements special_port1 \-isolation_supply_set iso_ss</code>	iso1 is read and stored
<code>set_isolation_control iso1 -domain etm_pd \-location parent \-isolation_signal en \-isolation_sense high</code>	iso 1 is implemented at the top level
<code>set_isolation iso2 -domain etm_pd \-elements special_port2 \-isolation_supply_set iso_ss</code>	iso2 is read and stored
<code>set_isolation_control iso2 -domain etm_pd \-location self \-isolation_signal en \-isolation_sense high</code>	iso2 is ignored at the top level
<code>set_isolation iso3 -domain etm_pd \-elements sub/special_port1 \-isolation_supply_set iso_ss</code>	ignored at the top level

The `set_isolation iso4 -domain etm_pd \-elements special_port3 \-no_isolation` command is honored

The `-elements` option supports ports and instances of a design. You can specify macros, hierarchical cells, black boxes and pads in the `-elements` option.

To control filtering based on the `-applies_to` and `-elements` options, set the `set_app_var upf_isols_allow_instances_in_elements` and the `set_app_var upf_iso_filter_elements_with_applies_to` application variables to enable.

The default value of the `upf_isols_allow_instances_in_elements` variable is true.

The default value of the `upf_iso_filter_elements_with_applies_to` variable is ENABLE.

### 6.1.2.10 Level-Shifter Strategies

The location parent level-shifter strategies may be defined on the power domain scoped on ETM. Level-shifter strategy (`set_level_shifter`) on ETM is honored if the domain is created and `-location parent` is specified during top level optimization. Else, it is ignored.

#### Examples

```
set_level_shifter ls1 -domain etm_pd \-elements special_port1 \-location parent ls1  
will be implemented at the top level  
set_level_shifter ls2 -domain etm_pd \-elements special_port2 \-location self ls2 is  
ignored at the top level  
set_level_shifter ls3 -domain etm_pd \-elements special_port3 \-no_shift command is  
honored
```

The -elements option supports ports and instances of a design. You can specify macros, hierarchical cells, black boxes and pads in the -elements option.

To control filtering based on the -applies\_to and -elements options, set the `set_app_var upf_isols_allow_instances_in_elements` and the `set_app_var upf_iso_filter_elements_with_applies_to` application variables to ENABLE.

The default value of the `upf_isols_allow_instances_in_elements` variable is true.

The default value of the `upf_iso_filter_elements_with_applies_to` variable is DISABLE.

VC LP drops the strategy when the same element is referred by two different strategies written on same domain.

```
set_level_shifter ls1 -domain PD1 -elements {in1} -rule low_to_high -location self
set_level_shifter ls2 -domain PD1 -elements {in1 out1}-rule low_to_high -location self
```

In this case, in1 port is referred by two strategies PD1/ls1 and PD1/ls2. VC LP reports an error for PD1/ls2 strategy and completely ignores the PD2/ls2 strategy.

#### 6.1.2.11 Retention Strategies

Retention strategies that are defined on ETMs are used for block level optimization. They are ignored at the top level since they do not have an impact on top level optimization.

#### 6.1.2.12 Scope Specific

UPF scope specification commands (`set_design_top` and `set_scope`) are honored on ETMs. Block level UPF may set the scope to a child instance inside ETM and execute commands at that scope. To bring back the scope to the top level, you can use the `set_scope` variable.

SRSN Overrides Related\_power/ground

SRSN overrides related\_power/ground attributes of IO pins of lib cell.

#### 6.1.2.13 Violations and Messages Introduced for ETM UPF Flow

##### 6.1.2.14 Violations introduced for ETM flow

- ❖ `UPF_PORT_EXTRA`: Check that identifies if ETM.db is missing any pg\_pins for corresponding supply ports described in ETM.upf.

##### 6.1.2.15 Messages

- ❖ [Warning] `UPF_SUPPLY_PORT_DIR_MISMATCH`: Supply Port direction mismatch.

#### 6.1.2.16 Limitations

Missing consistency checks between ETM.db and ETM.upf

- ❖ PG port direction mismatch checks.
- ❖ VC LP does not do `voltage_map` based consistency checks for the ETM liberty model and corresponding UPF's port states for supply ports.
- ❖ Presence of `internal_power` and `switch_function` in ETM.db for corresponding `create_power_switch` in ETM.upf.

### 6.1.3 Common Parser Level Limitations for Black box and ETM flow

The following UPF commands written above/at blackbox scope,

- ❖ For a list containing only elements which are inside bbox/ETM scope written above/at blackbox scope, the [*Warning*] *UPF\_EMPTY\_OBJ\_LIST\_BBOX* is not reported.
- ❖ When referring designed objects which are inside bbox/ETM scope, from "at" bbox/ETM scope, the [*Warning*] *UPF\_WITHIN\_BBOX\_ACCESS\_WARN* is not reported.
- ❖ For a list containing elements which are inside bbox/ETM scope and non bbox elements which are not available written above/at blackbox scope, the [*Warning*] *UPF\_EMPTY\_OBJ\_LIST\_MIXED* is not reported.

List of commands

- ❖ *set\_design\_attributes -elements*
- ❖ *set\_isolation -exclude\_elements*
- ❖ *set\_isolation -isolation\_signal*
- ❖ *set\_isolation -isolation\_supply\_set*
- ❖ *set\_level\_shifter -exclude\_elements*
- ❖ *set\_port\_attributes -elements*
- ❖ *set\_port\_attributes -exclude\_elements*
- ❖ *set\_port\_attributes -ports*
- ❖ *set\_port\_attributes -exclude\_ports*
- ❖ *set\_related\_supply\_net -object\_list*
- ❖ *set\_repeater -exclude\_elements*
- ❖ Pixl2/Plato messaging differences
- ❖ Reason: the empty list checks are not added for these commands to preserve cross tool compatibility.

## 6.2 Golden UPF Flow

VC LP has introduced a new method for maintaining the UPF multi-voltage power intent of a design. This method uses the original golden UPF file throughout the synthesis, physical implementation, and verification steps, along with supplemental UPF files generated by the Design Compiler (DC) and IC Compiler (ICC) tools.

[Figure 6-11](#) compares the traditional UPF-Prime flow with the golden UPF flow.

In the traditional flow, each stage of the flow uses a single UPF file that is modified during synthesis and physical implementation in the following ways:

The original UPF file, written in the RTL description, is modified by DC, resulting in the UPF' (UPF prime) file. The UPF' file reflects the changes that occurred during synthesis such as insertion of always-on buffers and inverters, object name changes, and PG connections to cells.

The UPF' file is modified by ICC, resulting in the UPF'' (UPF double prime) file. The UPF'' file reflects changes that occurred during physical implementation, such as the insertion of power switch cells, isolation cells, and foreign-domain physical feed-through buffers.

As the traditional flow modifies the original UPF file to make the UPF' and UPF'' files, it is called the UPF-prime flow.

The golden UPF flow maintains and uses the same, original golden UPF file throughout the flow. The DC and ICC tools write power intent changes into a separate supplemental UPF file. Additionally, in some cases, the DC or ICC may generate a name mapping file containing renaming rules.

VC LP uses a combination of the golden UPF file, the supplemental UPF file, and the name mapping file (if it is generated) instead of a single UPF' or UPF'' file.

The golden UPF flow offers the following advantages:

- ❖ The golden UPF file remains unchanged throughout the flow, which keeps the form, structure, comment lines, and wild-card naming used in the UPF file as originally written.
- ❖ You can use tool-specific conditional statements to perform different tasks in different tools. Such statements are lost in the traditional UPF-prime flow.
- ❖ Changes to the power intent are easily tracked in the supplemental UPF file.
- ❖ You can optionally use the Verilog netlist to store all PG connectivity information, making connect\_supply\_net commands unnecessary in the UPF files. This significantly simplifies and reduces the overall size of the UPF files.

By default, the golden UPF flow is enabled in VC LP.

**Figure 6-11 UPF-Prime (Traditional) and Golden UPF Flows**

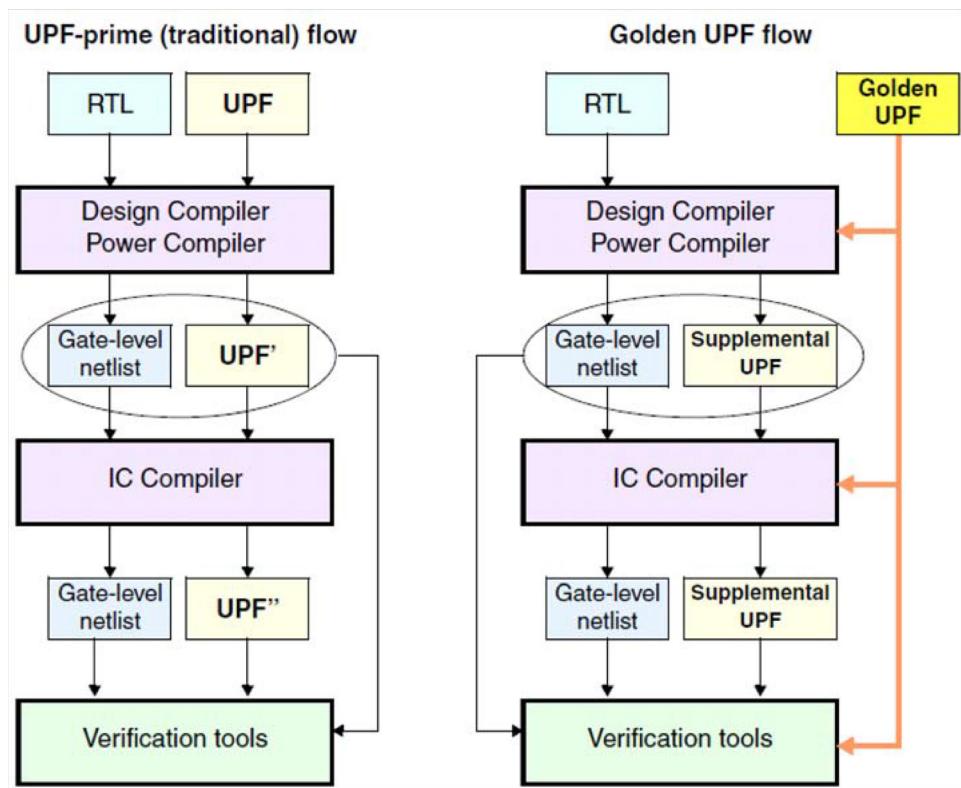
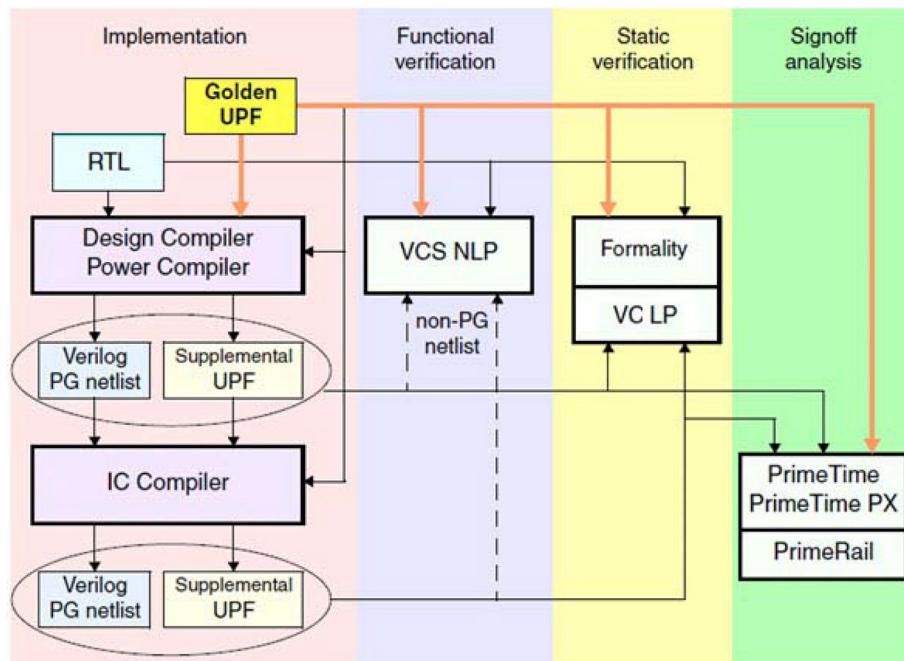


Figure 6-12 shows how the golden UPF file, supplemental UPF files, name mapping files, and Verilog netlists with PG connectivity information are used by VC LP and other tools in the golden UPF flow.

**Figure 6-12 Verification Tools in the Golden UPF Flow**



### 6.2.1 Supplement UPF File

A supplemental UPF file is a UPF file generated by the DC or ICC tool to complement the original golden UPF file. The supplemental file contains the power intent changes that occur during synthesis or physical implementation. Apply the golden UPF and supplemental files together to fully specify the power intent for a synthesized netlist.

To pass the supplemental UPF file to the `read_upf` command, use the `-supplemental` option. You can use the option in the following way:

- ❖ Pass the `-supplemental` option in `read_upf` command as follows:

```
%vc_static_shell> read_upf golden.upf -supplemental supplemental.upf -strict_check false
```

The `-strict_check false` option means that the golden UPF is being reapplied, so design name-match checking is not strictly exact, but uses rule-based renaming and the name mapping file, if any. The `-supplemental` option applies a UPF file as a supplement to the golden UPF, with strictly exact name-match checking.

- ❖ The supplemental UPF file contains any or all of the following information:
  - ◆ Policies of the `-no_isolation` type.
  - ◆ The `create_power_domain` command with the `-update` option.
  - ◆ The `connect_supply_net` commands to define an exception connection.

The following is a sample supplemental UPF file:

```
set_isolation snps_no_iso_0 -domain core1_pd -elements i_core1_i_core11/iso -no_isolation
create_power_domain core1_pd -elements {new_root_cell} -update
connect_supply_net top_vdd -ports {core11_out1_1_UPF_LS/VDD}
connect_supply_net top_vss -ports {core11_out1_1_UPF_LS/VSS}
```

For the golden UPF flow, the Synopsys Galaxy platform recommends generating a PG netlist, that is, all PG connections exist within the netlist. Hence, the generated supplemental UPF file should be concise and only contain information regarding the `-no_isolation` policies and the `create_power_domain -update` command.

Therefore, it is expected that the netlist is either fully PG netlist connections in the design or is completely a non-PG netlist (that is, all PG connections are in the supplemental UPF consisting of the `connect_supply_net` commands). The golden UPF flow does not allow a partial mix of both.

## 6.2.2 Name Mapping File

During synthesis and optimization, the DC and ICC tools might remove, rename, or duplicate objects specified in the golden UPF flow. Object names in the golden UPF file might not match the corresponding object names in the generated Verilog netlist. When you apply the golden UPF and supplemental UPF files to a design in VC LP, the object names used in the golden UPF file must be properly mapped to the new names used in the design netlist.

To match the names of objects in the golden UPF file with the new and renamed objects in the design netlist, use the rule-based name matching approach. Using this approach, the tools can automatically match names resulting from the following:

- ❖ Standard Verilog renaming rules
- ❖ Un-grouping
- ❖ Wildcard name expansion

The name mapping files are specified in the VC LP input file using the `upf_name_map` variable. Enter one or more pairs of the design names and their respective mapping files. For example,

```
%vc_static_shell> set_app_var upf_name_map {{TOP TOP.nmf}}
```

or

```
%vc_static_shell> set upf_name_map {{TOP TOP.nmf} {MID MID.nmf} {BOT BOT.nmf}}
```

The following example shows the contents of a typical name mapping file.

```
# Query Rules
set_query_rules -hierarchical_separator{/x} -bus_notation{[] __ ()}

# Tcl built-in: local variables
set hier_1 A/B/C/D
set hier_2 A_B/B2/this/is/a/long/path

# Explicit object name map
define_name_maps -application golden_upf -design_name TOP \
-columns {class pattern options names} \
{cell mid/inst/U3 {} {mid_inst/U4 mid_inst/U5}} \
{cell mid/inst/bot+U5 {} {mid_inst/bot/U9} }

# cleanup
unset hier_1
unset hier_2
```

## 6.2.3 Use Models

You can use the following script examples as guides for creating your own golden UPF flow scripts.

### 6.2.3.1 RTL and Golden UPF

```
read_upf top.upf -strict_check true #By default strict_check is true
```

The `-strict_check true` option means that the VC LP performs a strict check for any object naming mismatches between the UPF file and objects in the design

### 6.2.3.2 Non-PGNelist/PGNetlist and Golden UPF

```
set upf_name_map {{TOP TOP.nmf}}
```

```
read_upf full_rtl.upf -supplemental full_supp.upf -strict_check false #Reads golden UPF along with supplemental UPF. Rule based matching is enabled.
```

## 6.2.4 Limitations

- ❖ The following are the limitations with the Golden UPF (GUPF) feature/flow support:
  - ◆ `synopsys upf_name_map` pragma is not supported.  
`//synopsys upf_name_map design_name "verilog_file_name.nmf"`
- ❖ The following is the limitation with the VC Static tool features when the GUPF flow is enabled:
  - ◆ The `all_upf_error` application variable is not supported.  
 This variable is used when the UPF and/or the design is dirty. VC LP reports all UPF error at once by using this application variable.
- ❖ The following UPF parser syntax checks are not yet added to the Golden UPF flow:
  - ◆ *[Error] UPF\_MULT\_DRV: Multiple driver cannot be added*
  - ◆ *[Warning] UPF\_SRSN\_PDBP: SRSN on hierarchical boundary port.*
  - ◆ *[Warning] UPF\_LNSPEC: Location not specified. Please specify one of 'self/parent/automatic' as location*
- ❖ The `golden_upf_report_missing_objects` application variable is not supported.  
 This variable enables the reporting of missing (unmapped) objects during UPF parsing. Design Compiler (DC) supports this variable.  
 However, since the VC LP verification solution always reports on missing objects during UPF parsing, there is no support for this variable.

## 6.3 Support for Signoff Abstraction Model Based Hierarchical Flow

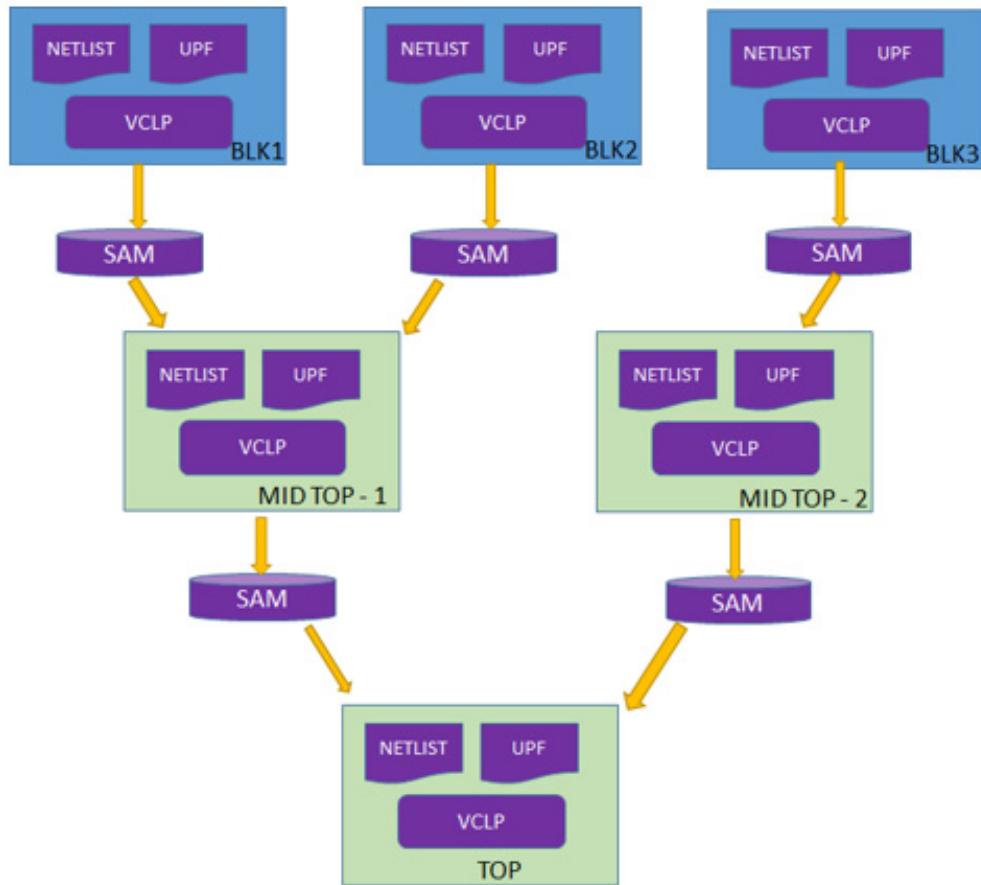
Designers adopt hierarchical verification flows because of growing design sizes, low power complexity, and for early stage verification. For hierarchical verification, you can use black box (bbox) flow, ETM flow or Stub/glass box flows that offer various degrees of trade-offs for accuracy of QoR and performance. While black box flow is best for performance, the flat runs give the best QoR as full design is available for checks.

VC LP has introduced the Signoff Abstraction Model (SAM) flow for hierarchical verification that gives the best QoR by retaining enough logic needed from hierarchical modules, while the performance of the runs would be better than flat runs. Also, this flow enables top level integrator to focus on top-level violations only, and integration related issues, and not worry about violations deep inside the hier-blocks that block owners would anyways take care.

In the SAM flow, blocks that are integrated into SoC must be abstracted first using VC LP. During abstraction, the lib cells/ hierarchical instances and net connections that are not needed for top only verification activities are removed and a new model (SAM) is saved into a new Verilog netlist file. This Verilog model can be loaded into the SoC instead of the original block to do the Top + SAM verification. The

benefits of using this flow (Top + SAM) includes less memory usage, improved run time, and focused violations related to the top level integration.

**Figure 6-13 SAM Flow**



### Note

This feature is an LCA feature. Please contact [mvsupport@synopsys.com](mailto:mvsupport@synopsys.com) for more details on this feature.

#### 6.3.1 Prerequisites

The following are the prerequisites for using the SAM feature.

- ❖ The VC-LP-ULTRA license key must be checked-out to create SAM models.
- ❖ The SAM flow is supported for netlist and RTL designs. You can have a Verilog netlist or a pg-netlist.

### Note

The SAM support documented below is for NETLIST and PGNETLIST designs. Support for RTL designs is documented separately in section “[New SAM Abstraction \(Lightweight SAM\) Model](#)”.

- ❖ The same UPF design can be used at any stage of the run. You need not modify the UPF for the SAM flow.

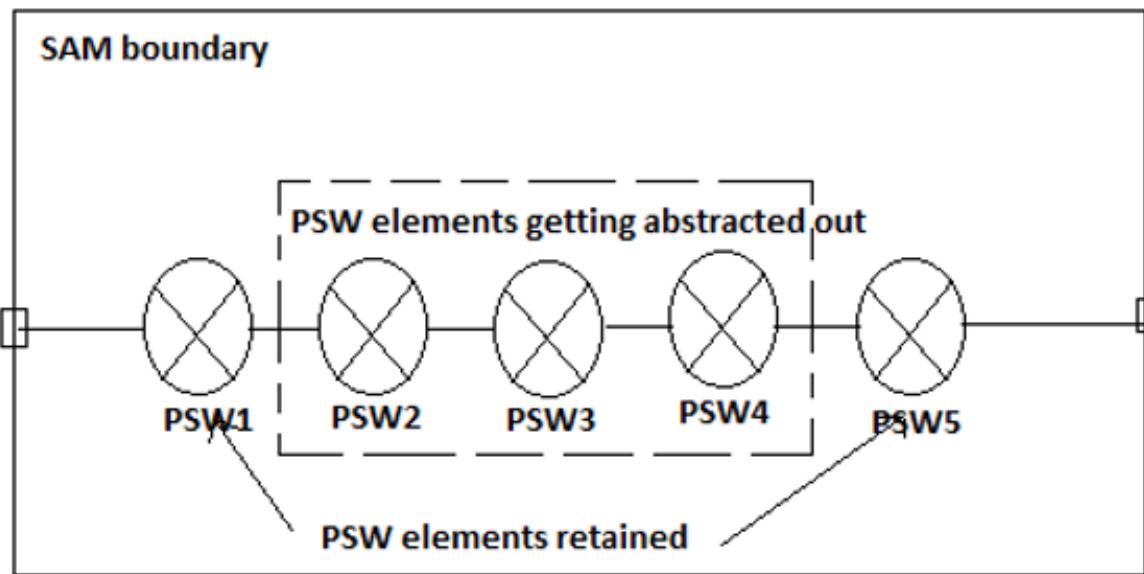
- ❖ If a PSW chain is available inside the abstraction boundary, a power switch strategy to match the PSW chain must be present at the hierarchical UPF of the module. Else, the PSW chain is broken at the nearest PSW cell in the SAM scope.

### Example

There is a PSW chain inside the SAM boundary. Only the PSW instances connected to the SAM boundary ports are retained. The other elements in the chain gets abstracted out. The following path is retained:

- ◆ SAM input port -> PSW1\_EN
- ◆ PSW5\_ACK -> SAM output port

**Figure 6-14 Example for PSW Chain Inside SAM Boundary**



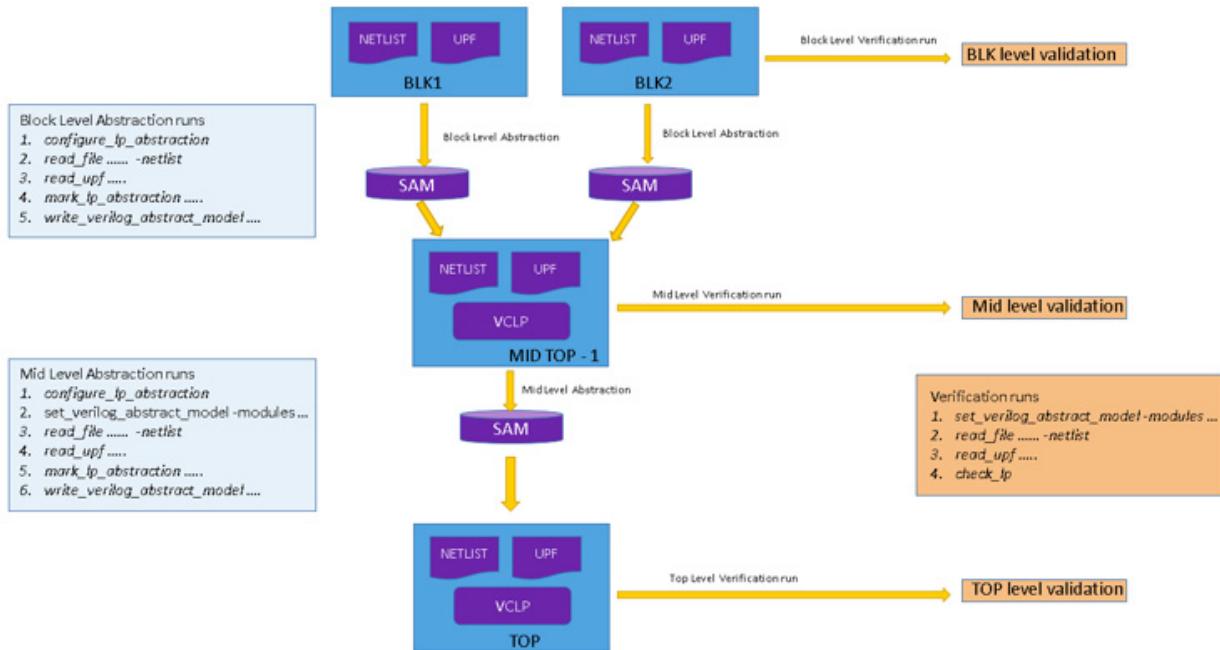
This will result in noise in the violations. For example, the PSW\_CONTROL\_CONN violation is reported on the highlighted path.

### 6.3.2 Use Model

The SAM flow includes the following stages:

- ❖ Writing and Abstracting SAM models
- ❖ Reading and Integrating SAM models for SoC runs

[Figure 6-15](#) shows key steps, and VC LP script snippets for writing abstract/SAM models for a 3-layer hierarchy design.

**Figure 6-15 Writing Abstract SAM Models for 3-layer Hierarchy Design**

### 6.3.2.1 Writing SAM model

You can create a SAM model for any block or IP or design.

- ❖ Configure the run for abstraction.
- ❖ Perform the required VC LP checks.
- ❖ Save the SAM model.
- ❖ The `configure_lp_abstract_model` command enables and disables a certain set of LP tags (`configure_tag -app LP` and `report_app_var`). These are minimal settings needed for preparing the design to abstract enough logic needed for top level verification. These settings may be different from the ones used in the user verification/sign-off flows. Hence, abstraction (writing out SAM model) and verification (block signoff) is not recommended in the same run. However, you can write SAM model and generate QoR result in same run using single run SAM flow (see section “[Support for Single Run SAM Flow](#)” for more details).
- ❖ The `mark_lp_abstraction` command runs VC LP checks needed for abstraction process. The command comes with two options `-netlist` and `-pgnetlist`. The options are to be specified for each netlist and PG netlist designs. Default is PG netlist.
- ❖ Abstraction will change the original module names of modules that are instantiated inside SAM module boundary. The updated module names will have  
`<original_module_name>_abs_<name_of_the_abstracted_module_top>`

#### 6.3.2.1.1 Abstracting Block Level (IP) Designs

The following steps are required to abstract a design:

```
set search_path "."
set link_library " demo_psw_10v_pg.db tiny.db"
configure_lp_abstract_model
```

```
read_file -netlist -top and_module -f verilog {module.v}
read_upf module.upf
mark_lp_abstraction
write_verilog_abstract_model -file module_abs.v
```

- ❖ If the `-path` option is specified, the abstracted design is saved in the following file:  
`<path_name>/<Top_module_name>/verilog/<Top_module_name>_SNPS_VCSTATIC_INM_abstract.v`
- ❖ If the `-file` option is specified, the abstracted design is saved in the specified file in the option.

**Note**

In netlist, `mark_lp_abstraction` is not mandatory, You can use `check_lp` for marking.

### 6.3.2.1.2 Abstracting Sub Systems (partition) Designs

For partition level designs, you may have used SAM models or flat models for the blocks. Either way, the sub system (partition) level SAM model can be created. The recommendation would be to use the SAM models for the blocks at the time of abstracting the partition level designs.

1. Manually replace original Verilog file for the block modules with the abstracted Verilog module in the design file list for partition level VC LP runs.
2. Execute the following commands:

```
configure_lp_abstract_model
set_abstract_model -modules_list and_module -app lp -path abs -format text
read_file ..... -netlist
read_upf .....
mark_lp_abstraction -netlist/-pgnetlist
write_verilog_abstract_model -path/-file
```

### 6.3.2.1.3 Enabling Aggressive Optimization

Use the `lp_abstraction_aggressive_opt` Tcl command to enable aggressive optimization while abstracting model generation. Aggressive optimization refers to all those optimizations which will reduce the size of abstracted output, but it may lead to some noisy violation in the TOP+SAM run.

This optimization includes homogeneous fanout optimization, buffer chain compression, PSW chain compression, and pg to logic path optimization. These optimizations are categorized in various categories like homogeneous sink/psw/pg/buffer chain. You can pick one or multiple types of optimization as per your requirement.

#### Use Model

```
lp_abstraction_aggressive_opt [-psw <psw>] [-pg <pg>] [-bufchain <bufchain>] [-homosink <homosink>] [-all <all>]
```

- ❖ `[-psw <psw>]`: To enable optimization done for power switch chains
- ❖ `[-pg <pg>]`: To enable optimization done in the area of pg to logic connection
- ❖ `[-bufchain <bufchain>]`: This option optimizes a homogeneous buffer chain to mark only the first and last buffers
- ❖ `[-homosink <homosink>]`: This option enables common net-based optimization for homogenous logical sinks
- ❖ `[-all <all>]`: To enable all recommended optimizations

#### Example

```
lp_abstraction_aggressive_opt -psw -pg -bufchain
```

### 6.3.2.2 Reading SAM Models for SoC Verification

At SoC level, you can integrate all SAM models for the partitions and run full flat verification:

To read back the abstracted SAM design, perform the following steps.

1. Manually replace original Verilog file for the module with the abstracted Verilog module in the design file list.
2. Execute the following commands

```
set search_path ". "
set link_library " demo_psw_10v_pg.db tiny.db "
set_abstract_model -modules_list and_module -app lp -path abs -format text
read_file -netlist -top top -f verilog {top.v module_abs.v}
read_upf top.upf
check_lp -stage all
```

#### Notes

- ❖ The `set_abstract_model` command is used to specify the abstracted blocks. This helps VC LP parser to deduce the SAM module scopes, and to ignore missing objects related issues created due to abstraction.
- ❖ By default, UPF parser messages related to abstraction modules are suppressed to improve the performance. To enable these messages, set the `lp_disable_quietabs_in_read_upf` application variable to true.
- ❖ Only debugging purposes, VC LP dumps the abstracted design in near matching Verilog format.  
`write_abstract_model -debug_verilog`

### 6.3.3 New SAM Abstraction (Lightweight SAM) Model

VC LP supports new SAM flow (Lightweight SAM) which can reduce more logics based on SPA - receiver\_supply. For more information on this support, contact the Synopsys support center.

### 6.3.4 Enhancements to Stub and Black box Modules

VC LP retains the black box and STUB modules even if they have only PG connections. These modules can be removed using the optimization algorithm using the `lp_abstraction_pg_marking_opt` application variable. By setting the application variable to true, the black box and STUB modules with only PG connections is removed from the abstracted design. By default, the `lp_abstraction_pg_marking_opt` application variable is set to false.

#### Example

Verilog:

```
core2 CORE2 ( .VDD(VDD) , .VSS(VSS) );
module core2 (in1,in2, VDD, VSS);
input in1 ,in2, VDD ,VSS;
DMEN and1 (.Z(w1));      // This logic inside core2 will not be retained during
abstraction.
endmodule
```

#### Abstracted output: (default)

```
core2_abs_top CORE2 (.VDD(VDD) ,
```

```

.VSS(VSS)) ;

module core2_abs_top(in1,
    in2,
    VDD,
    VSS) ;

    input          in1;
    input          in2;
    input          VDD;
    input          VSS;

//wire           w1; // Abstracted out

    // Instance and1 is abstracted out
endmodule

```

#### Abstracted output: (with lp\_abstraction\_pg\_marking\_opt set to true)

<No CORE2 instance, no core2\_abs\_top module >

### 6.3.5 Macro Cells with Internal Isolation

Pins of macro cells which are related to internal isolation applied in the macro cells are forcefully retained to reduce noisy \*UNCONN violations. This support is added under the lp\_abstraction\_macro\_iso\_marking application variable (default true). You can disable this optimization by setting the application variable to false.

### 6.3.6 PSW Port Punching

The generate\_psw\_ctrl\_signal\_crossover application variable is recommended to be used for RTL designs. Hence, the generate\_psw\_ctrl\_signal\_crossover application variable should not be enabled in SAM flow to avoid unexpected results.



#### Note

Under the lp\_enable\_virtual\_protection application variable, function of generate\_psw\_ctrl\_signal\_crossover and generate\_upf\_ctrl\_signal\_crossover will also be enabled.

### 6.3.7 Debug diagnostics

#### 6.3.7.1 Logic reduction in SAM model

During abstraction process, some information of abstraction is printed on the vc\_static\_shell screen. An example is shown below.

Paths (keep/total)	= 611595/777863 (1.27x)
Nets (keep/actual)	= 411858/4879156 (11.85x)
Ports (keep/actual)	= 186491/564259 (3.03x)
Pins (keep/actual)	= 5870374/22101283 (3.76x)
Non-sequential Connectivity Operators (keep/actual)	= 64154/1091127 (17.01x)
Non-sequential Other Operators (keep/actual)	= 16070/333222 (20.74x)
Libcell Instances (keep/actual)	= 648851/2764161 (4.26x)
Hierarchical Instances (keep/actual)	= 28390/34229 (1.21x)
Total Design Objects (keep/actual)	= 7226188/31767437 (4.40x)

Other additional information on crossovers and paths are saved into the file `vcst_rtdb/internal/lp/abstractionStatistics.rpt`.

#### Example:

Crossover Paths (From/To Primary Ports /Actual)	= 546016/672113 (81.24%)
Crossover Paths (Completely Contained inside/Actual)	= 126097/672113 (18.76%)
Marked crossover Paths (Marked/Total)	= 667422/672113 (99.30%)
Marked crossover Paths (From/To Primary Ports /Actual)	= 546016/667422 (81.81%)
Marked crossover Paths (Completely Contained inside/Actual)	= 121406/667422 (18.19%)

### 6.3.8 Parser Messages, Tcl Commands and Application Variables Introduced

The following tables lists the new violation messages, Tcl commands, and application variables that are introduced for the SAM flow:

**Table 6-3 Violation messages**

Category	Scenario	Error/Warning
VC LP parser output	Object inside SAM scope is referred in the UPF.	[Warning] UPF_WITHIN_SAM_ACCESS_WARN: SAM object accessed from outside. Trying to access object '' of type '' in command option '' which is inside a SAM scope. Please specify an object which is outside or at the SAM boundary.
VC LP parser output	Object referred with wildcards in the UPF is resolved into a missing object inside SAM scope.	[Warning] UPF_WILDCARD_WITHIN_SAM_ACCESS: Wildcard Object not found Wildcard object '' of expected type '' specified with command option '' could not be resolved. Please specify a valid object.
VC LP parser output	Object list referred in a UPF command has become empty. But the objects in the list are all in the SAM scope.	[Warning] UPF_EMPTY_OBJ_LIST_SAM: No valid objects in list. Object List specified with command option '' resolved to an empty list. Please specify at least one valid object in the list.
VC LP shell output	Design is not read as a netlist. But abstraction is attempted.	[Warning] DESIGN_READ_NOT_NETLIST: Design was not read in using the -netlist switch. Verilog netlist will not be written out for the abstracted module
VC LP shell output	enable_lp_abstraction_marking is set to false in the design and abstraction is attempted.	[Warning] TCL_CMD_DEPENDS_ON_APP_VAR: Tcl command is depending on application variable. write_verilog_abstract_model command should only be executed when Tcl variable 'enable_lp_abstraction_marking' is set to true
VC LP shell output	enable_lp_abstraction_marking is set from false to true after read_upf.	[Error] TCLVAR_INVALID_UPDATE: Invalid update of tcl variables. The tcl variable 'enable_lp_abstraction_marking' cannot be changed from 'false' to 'true'. Tcl varibale should only be changed before executing tcl command 'read_upf'.
VC LP shell output	Writing abstract model before reading design successfully.	[Error] SHOULD_AFTER_DESIGN_LOADED: write_verilog_abstract_model should be used after design is loaded.

**Table 6-3 Violation messages**

Category	Scenario	Error/Warning
VC LP shell output	Writing abstract model before reading UPF successfully.	[Error] TCL_CMD_DEPENDS_ON_TCL_CMD: Tcl command is depending on Tcl command. write_verilog_abstract_model command should only be executed after executing Tcl command 'check_lp'
VC LP shell output	Setting Verilog abstract model after reading UPF.	[Error] COM_OPT015: set_abstract_model should be used before design is loaded.
VC LP shell output	Specifying a SAM module not available in the design and reading the UPF	[Warning] SAM_MODEL_NOT_FOUND: SAM Model not found. SAM Model 'UNAVAIL_MODULE' specified not found in design. Please specify a valid model name for SAM.
VC LP shell output	Specifying a SAM module not available in the design and reading the UPF.	[Warning] SAM_MODEL_NOT_FOUND: SAM Model not found. SAM Model 'UNAVAIL_MODULE' specified not found in design. Please specify a valid model name for SAM.
VC LP shell output	write_verilog_abstract_model -path defines a hierarchical path which has missing folders in the path.	[Info] ABSTRACT_MODEL_DIR_INVALID: The designated directory 'a/b' for abstract model output data cannot be created
VC LP shell output	dump_debug_data -during_abstract specified without setting enable_lp_abstraction_marking to true	[Error] ABSTRACT_DEBUG_NON_ABSTRACT_RUN: -during_abstract specified in dump_debug_data without enabling app var enable_lp_abstraction_marking. This option will be ignored.
VC LP shell output	Path specified in the dump_debug_data command using option -path does not exist.	[Error] DIR_PATH_NOT_FOUND: Directory '<dir-path>' specified in command 'dump_debug_data' option '-path' does not exist. Data will be dumped in the current working directory.

**Table 6-4 Tcl Commands Introduced**

TCL Command	Usage	Description
configure_lp_abstract_model	configure_lp_abstract_model	Enables a minimal set of Low Power checks and application variables for LP SAM model generation.
mark_lp_abstraction	mark_lp_abstraction	Enables a minimal set of checker stages for LP SAM model generation.
write_verilog_abstract_mode l	write_verilog_abstract_mode l -path <unix_dir_name>	Command to write Verilog abstract model from a given VC LP run.

**Table 6-4** **Tcl Commands Introduced**

TCL Command	Usage	Description
set_abstract_model	set_abstract_model - modules_list and_module - app lp -path abs -format text	Command to use abstracted design models in a VC LP run.
dump_debug_statistics	dump_debug_statistics - modules <module_name>	Command to pass names of the abstracted design models. Various debug dumps related to cross-overs, root supplies, PSTs are dumped associated to the given abstracted models.

**Table 6-4** **Tcl Commands Introduced**

TCL Command	Usage	Description
compare_json	<pre>compare_json -mode {filtered_xover, rootsupply, global_pst} - original {TOP+SAM.json} - current {Full-Flat.json} - dump_file {diff_file_name}</pre>	<p>This command has been introduced to compare Low power database difference between TOP+SAM run and Full-Flat run.</p> <p>The compare_json command can be used without loading any design.</p> <p>The difference of content are dumped in the user given diff file mentioned in -dump_file option.</p> <p>The -dump_path option enables you to specify the exact location for the file. This not a mandatory option. If the path is provided, the file will be dumped in <i>session_dir/</i> <i>vcst_rtbd/lpdb/&lt;path&gt;/&lt;file_name&gt;</i>, else file will be dumped in <i>vcst_rtbd/lpdb</i></p> <p>The -mode is a non-mandatory option, and -mode regr is made the default if no mode option is given.</p> <p><b>Example 1</b> To compare JSON files collecting two directories which is dumped using dump_lp_db command following command can be used and report will be in lpdb/json_diff_dir/diff.txt</p> <pre>compare_json -current lpdb_current -original lpdb_original -dump_file diff -dump_path json_diff_dir</pre> <p><b>Example 2</b> To compare two JSON files, following command can be used and report will be in lpdb/diff.txt</p> <pre>compare_json -current lpdb_current/xover.json - original lpdb_original/xover.json - dump_file diff</pre>
waive_lp_blockViolation	<pre>waive_lp_blockViolations - blocks &lt;block_names&gt; (list of block names)</pre>	Apply waiver to all violation with block instance as container instance

**Table 6-4** **Tcl Commands Introduced**

TCL Command	Usage	Description
dump_debug_data	<pre>dump_debug_data [-during_abstract] (Enables debug dump during block abstraction) [-crossing_modules &lt;crossing_module_name&gt;] (crossing module name list) [-components &lt;component_names&gt;] (component list) [-roots &lt;crossover_root_list&gt;] (crossover root list) [-path &lt;unix_dir_name&gt;] (Directory where dump data will be available) [-report_system_pst] (Dump system pst) [-all] (Dump everything)</pre>	When this command is specified, tool will dump the abstraction statistics in an internal report present at the following path <rtdb_dir>/internal/lp/abstractionStatistics.rpt
report_violations -app LP	<pre>report_violations -app LP - verbose -xml -limit 0 -file &lt;moduleViolationReport.xml&gt; -supply_relation_file &lt;supply_relation_file_name.xml&gt;</pre>	Dump supply relationship file for additive flow
generate_supply_map	<pre>generate_supply_map - violation_file &lt; moduleViolationReport.xml&gt; -module &lt; SAM module &gt; - output_supply_file &lt;generated supply mapping file&gt;</pre>	Generate supply map to map flat violations with block level violations
generate_cloned_violations	<pre>generate_cloned_violations -violation_file &lt; moduleViolationReport.xml&gt; -module &lt;SAM module name&gt; -supply_mapping_file &lt;supply name mapping file from flat run to read&gt; - supply_relation_file &lt;supply relation file generated at module run to read&gt;</pre>	Clone block level violations into SAM run database

**Table 6-5 Application Variables**

Application Variable	Default Value	Description
enable_abstraction_extra_iso_marking	TRUE	When this application variable is enabled, VC LP perform bus merging which is going to merge set of bus compatible violations. and replace it with one merged violation. See man page for more details.
enable_lp_abstraction_debug	FALSE	This application variable enables/disables abstraction related debug dump in VC LP flow. This variable is not used, and it will be removed from next release. Use the dump_debug_data command instead.
enable_lp_abstraction_marking	FALSE	This application variable enables/disables marking abstracted design for a block that needs to be retained for the top level runs for Low Power.
lp_disable_quietabs_in_read_upf	FALSE	This application variable enables or disables – quiteAbs option in the read_upf command.
lp_abstraction_psw_marking_opt	FALSE	<b>Power Switch Chain Optimization</b> PSW chains (or part of chains) with only one enable and acknowledgement signal with same lib cells and same input/output power connections, without multi-fanout paths are optimized by removing intermediate PSW cells and stitching the cells in the ends.
lp_abstraction_unmark_unconnected_inst	TRUE	While creating the SAM model, some empty modules and unconnected instances have boundaries present within the model. When this application variable is set to true, all unconnected instances and empty modules are unmarked.
lp_abstraction_generate_light_sam	FALSE	If the application variable is set to true, light weight SAM model is generated. Light weight SAM refers to a SAM generation which contains only essential logic. Using light weight SAM in TOP+SAM run may lead to population of some noise related to UNCONN/UNDRAIVEN violations.
lp_abstraction_extra_logic_marking	false	When set to true, VC LP performs extra logic markings to forcefully retain the LP devices (like ISO/ELS/RET/..) inputs and outputs to reduce noises like *UNCONN*/*CONN*. The default value is false. These forcefully retained paths may not be connected to primary input or primary output.

**Table 6-5 Application Variables**

Application Variable	Default Value	Description
lp_report_common_parent	false	This application variable is introduced for SAM subtractive flow and additive flow. When this application variable is set to true, VC LP reports <i>ParentInstance</i> and <i>FullyContained</i> debug fields with all VC LP violation tags. The <i>ParentInstance</i> can help to waive violations which are fully inside a SAM block in SAM subtractive flow. The <i>FullyContained</i> debug fields can help to integrate SAM block violations in SAM additive flow.
lp_abstraction_retain_full_psw	false	In SAM, only 1st power switch connection in power switch chains was retained after skipping buffer/inverters. In a power switch chain directly connecting to a top level port, only the 1st PSW cell was getting retained. Subsequent connections were discarded. This caused noisy violations on psw control and ack connectivity such as PSW_CONTROL_CONN. The new application variable lp_abstraction_retain_full_psw is introduced to allow retention of the full psw chain during abstraction when set to true.

### 6.3.9 Support to Retain Control Sources Driving Primary Outputs of a SAM Boundary

You can retain control sources driving primary outputs of a SAM boundary. You are advised to specify such ports using one of the below methods.

Application Variable Name	Support Provided
lp_abstraction_retain_path_wo_combo	Retain the paths to primary outputs till it reaches a FF. If no FF encounters, the path till the primary inputs of the block will retain. Clock gating cells will be traversed through.
lp_abstraction_retain_path_wo_combo	Retain the paths to primary outputs till it reaches a FF or a combination cell. Clock gating cells will be traversed through.
port_list_arch_check {port_list}	The same support specified for lp_abstraction_retain_path_wo_combo is available for the port list specified in this application variable.

### 6.3.10 Support for Comparing Architectural Checks

The violation comparison support has been enhanced to improve matching on architectural checks.

#### Use Model

- ❖ In the full-flat run, with `report_violations -app LP` execute the following command to save the supply source maps of the arch violations.

- ```
report_violations -app LP -xml -dump_arch_source <file_name>
```
- ❖ In TOP + SAM run during violation cloning, execute the following option:  
`generate_clonedViolations -arch_source_map <file_name_from_full_flat_run>`
  - ❖ The compare\_violations ARCH command, the violations cloned from block runs will be categorized into multi-match.

After merging the violation for a single full flat violation, we will have two violation in TOP+SAM run.

Both the violation will have the same source and sink, but corruption list will be different, so we need to move SOURCE\_OF\_CORR\_REC debug field as priority2 debug field. The compare\_violations command reports these violations as multi-match violations.

The compare\_violations command supports partial matching, and checks are performed as follows:

- ❖ Violations are considered for partial matching only if they are not an exact match.
- ❖ For priority 1: partial matched violations, for priority 2/ priority 3: debug field matching is not considered.
- ❖ Implementation violations can partially match with multiple reference violations

The following new files are saved:

- ❖ `OneToOne_Primary1DebugField_Partial_Matching.rpt`
- ❖ `OneToOne_Primary1DebugField_Partial_Matching.csv`
- ❖ `OneToMany_Primary1DebugField_Partial_Matching.csv`

You should specify the partial matching candidate tags as follows:

*configure\_tag\_updated.xml snippet*

```
<Tag name="ISO_INST_MISSING">
<Priority value="0">
...
...
<Object>Source</Object>
<Object>Sink</Object>
</Priority>
<Priority value="1" partial="1">
<Object>LogicSink</Object>
<Object>LogicSource</Object>
</Priority>
</Tag>
```

## Limitations

Currently this support has limitations. The fix do not properly categorize the multi-match.

It is recommended to use only up-to two *Priority 1* debug fields for the partial matching as the initial requirement was to support partial matching of *LogicSource/LogicSink*. Providing more partial matching debug fields may impact on performance.



### Note

The SAM architectural checks support has limitations in support and gaps in testing which is planned to be addressed in the coming releases.

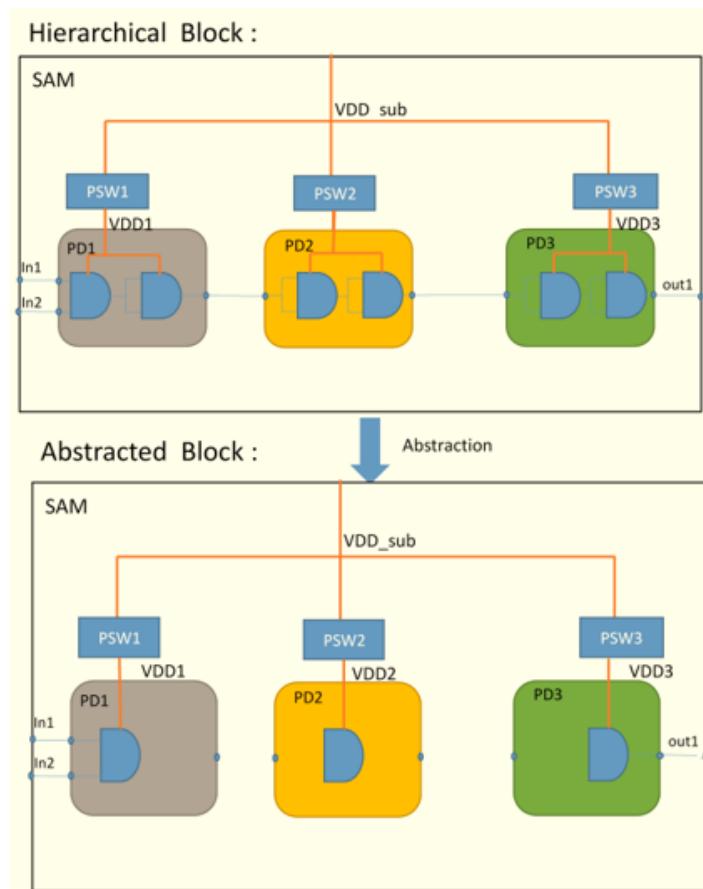
### 6.3.11 Support for SAM with NPS Flow

SAM with NPS (Non-Peripheral Supply) flow is introduced to reduce PST merging time. If a customer design has a huge amount of supplies that has no connection to SAM boundary or UPF strategies associated with SAM boundaries, these supplies are unnecessarily adding run time to VC LP run. With this support, VC LP ignores such supplies and improve on VC LP runtime without any QoR impact.

#### 6.3.11.1 Identifying NPS Supplies

[Figure 6-16](#) is an example design which illustrates how to identify Peripheral Supplies (PS) and Non-Peripheral Supplies (NPS).

**Figure 6-16 Example Design with NPS Supplies**



In the abstracted block, **VDD2** is fully internal supply. In other words, it is not connected to any object driving the SAM boundary or a strategy supply related to SAM boundary. Hence, it can be identified as non-peripheral supply (NPS).

**VDD1** and **VDD3** supplies are driving cells which connected to the top, those can be identified as peripheral supplies (PS).

#### 6.3.11.2 SAM With NPS Methodology

NPS supplies has no impact on 'TOP+SAM' run because they are not connected to SAM boundaries. Hence, you can identify NPS supplies and remove them from UPF using SAM+NPS flow.

1. Dump all Non-Peripheral supplies in block level with SAM run.

```

set_app_var lp_pst_build_method reduce → 1. Application variable to optimize pst calculation
set search_path ". /remote/arch-work/harsha/LIB/DEMO_LIB/pgdb"
set link_library " demo_psw_10v_pg.db tiny.db"
configure_lp_abstraction → 2. Configure SAM flow app_vars and tags
read_file -netlist -top and_module -f verilog {module.v}
read_upf module.upf
mark_lp_abstraction → 3. Run SAM marking on design
lp_abstraction_generate_noninterface_supplies -file interface.xml → 4. Command for dump Non-Peripheral Supplies
write_verilog_abstract_model -file module_abs.v → 5. Write out SAM model

```

Example of dumped non-peripheral supplies (interface.xml):

```

<NonPeripheralSupply>
<supply>VDD1</supply>
<supply>VDD2</supply>
<supply>VDD3</supply>
</NonPeripheralSupply>

```

VDD1, VDD2 and VDD3 are marked as Non-Peripheral supplies

2. Load the dumped non-peripheral supplies with TOP+SAM run.

#### TOP+SAM run with NPS readback

```

set_app_var lp_pst_build_method reduce → 1. Application variable to optimize pst calculation
set search_path ".../remote/arch-work/harsha/LIB/DEMO_LIB/pgdb"
set link_library " demo_psw_10v_pg.db tiny.db"
set_verilog_abstract_model -module and_module → 2. Defining abstracted model list
read_file -netlist -top top -f verilog {module_abs.v hanging_nets.v}
lp_abstraction_read_noninterface_supplies -module and_module -file interface.xml → 3. Read NPS supply list (from respective abstraction runs)
read_upf hanging_nets.upf
check_lp -stage all -force

```

#### 6.3.11.3 Limitations:

- The add\_power\_state -group support is not available for NPS.

#### 6.3.12 Support for Single Run SAM Flow

Use the following command for a single pass SAM flow:

```

read_file
read_upf
check_lp
write_abstract_model

```

Previously, in SAM flow, it was not recommended to get the QoR report while abstraction. (QoR not accurate). So, the user has to run this block-level two times to get an abstract model and QoR report.

Now, you can generate block-level QoR reports during abstraction, and this feature is guarded by app var.

#### Use Model

```
set_app_var lp_abstraction_samgen_during_qor true
```

By default, the `lp_abstraction_samgen_during_qor` application variable is set to false

#### Limitations

- ❖ DESIGN\_FORK is not supported.

### 6.3.13 Support for OnTheFly SAM

Currently, you should identify the SAM boundaries of a chip and run SAM abstraction manually and then have to run the TOP+SAM readback. This process is time-consuming. With OnTheFly SAM flow, you do not need any of the above. You can set the `lp_abstraction_onthefly_sam_run` application variable and TCL commands in the flat level TCL file. VC LP will identify SAM boundaries and run those in parallel and run the readback sessions for the user automatically.

#### User Model

- ❖ `set_app_var lp_abstraction_onthefly_sam_run true`
- ❖ Use `set_host_options` command to provide machine configuration:
  - ◆ `set_host_options -num_processes 2 -protocol CMC -submit_command "sh" -wait_time 1`
  - ◆ If no. of the processes given in the command is less than the final no. of job, then more than one jobs will be posted on the machines
- ❖ After the proper setting distributed run will be invoked by the new command `start_onthefly_distributed_run`

This command should be added after `read_file` and `read_upf` commands.

#### Example Setup

```
set_app_var lp_abstraction_onthefly_sam_run true
set_host_options -num_processes 3 -protocol CMC -submit_command "sh" -wait_time 1

set search_path " /remote/arch-proj/testcases/pv_dbs/nupurg/DB/ "
set link_library "tiny.db"

configure_lp_tag -tag * -enable

read_file -netlist -top top -f verilog {top.v mod1.v mod2.v mod3.v}
read_upf top.upf

start_onthefly_distributed_run

exit
```

#### Notes

- ❖ All the application variables and tag configurations set in the setup file will be available in the block level runs and also readback run
- ❖ The number of machines provided in the `set_host_option` should be more or equal to no expected SAM modules. Otherwise, more than one SAM gen job will be posted on one machine and runs will be executed in parallel.
- ❖ If the number of candidate module for SAM generation is "n", then the total n+1 license will be consumed.

- ❖ Current sessions cannot be accessed and GUI, and TCL debug is not possible in the current session.
- ❖ All Block level runs and readback run is saved, and you can restore the sessions in the run location

## Limitations

- ❖ TCL commands except for application variable and tag configurations are ignored (ex, infra\_source)
- ❖ Only `load_upf` command is treated for identification of SAM module (`set_scope` is not supported)
- ❖ `load_upf` with different upf file for multiple instances of the same module is not acceptable
- ❖ Multi-layer SAM generation is not supported (If a design has multiple UPFs with a hierarchical structure, then the block which is most close to the top level will be considered as SAM boundary)
- ❖ Cannot control the On The Fly SAM run
- ❖ Not supported for SAM-NPS and SAM++ flows
- ❖ Report and waiver files path cannot be provided from the user, reports are defined at the pre-decided path.

