Verification Continuum™

VC LP
LCA Features

Version T-2022.06-SP1, September 2022

SYNOPSYS®

Copyright Notice and Proprietary Information

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

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.

Contents

1 VC LP LCA Features

This chapter provides a brief description of the LCA features offered as part of VC LP:

1.1 Support for Transient State

VC LP supports transient state. Transient state is a state that a system may enter for a few milliseconds before reaching a steady state. These states are considered as temporary states, not the permanent states. These states do exist for long enough for certain types of electrical violations to cause problems. Therefore, VC LP performs some checks for the transient states, and ignores the rest of the checks.

VC LP has introduced the following two Tcl commands for this support:

- `set_pst_transient`

  Use this command to set a state in the PST as a transient state. This command accepts one hierarchical state name, such as a/b/T1/S1, where a/b is a scope, T1 is a power state table in that scope, and S1 is a state in that table.

  Syntax

  ```
  %vc_static_shell> set_pst_transient -help
  Usage: set_pst_transient    # sets a power state from a table to be transient
          -state <state>         (Fully qualified state name from a PST)
  ```

- `rebuild_pst_data`

  Use this command to rebuild the internal PST data after you have marked a state as transient.

Syntax

```
%vc_static_shell> rebuild_pst_data -help
Usage: rebuild_pst_data    # Rebuild internal PST data, for example after marking states
transient
```

Example:

Consider the following UPF snippet:

```
create_pst chip_top -supplies              {VDD   VDD1 VDD2  VSS}
add_pst_state ON    -pst chip_top -state   {HV    HV   HV    ON }
add_pst_state OFF_1 -pst chip_top -state   {HV    OFF  HV    ON }
add_pst_state OFF_2 -pst chip_top -state   {HV    HV   OFF   ON }
```

Use the following commands in the Tcl file to define a state as transient state, and rebuild the PST data:

```
set_pst_transient -state chip_top/OFF_1
rebuild_pst_data
```

1.1.1 Tcl Commands Supporting Transient Option

The following Tcl commands are enhanced to report the transient state:

- The `-transient` option is added in the `report_system_pst` command.
- The `-only_transient` option is added in the `report_pst_state` command.
- The `-only_transient` option is added in the `get_supply_net_combinations` command.
- The `-transient` option is added in the `get_supply_rail_order` command.
- The `-transient` option is added in the `analyze_invalid_power_state` command.

Examples

- **`%vc_static_shell> report_system_pst`**

```
VDD,VDD1,VDD2,VSS
1.2,1.2,?,0
```

`%vc_static_shell> report_system_pst  -transient`

```
VDD,VDD1,VDD2,VSS,Transient?
1.2,OFF,1.2,0,Y
1.2,1.2,OFF,0,N
1.2,1.2,1.2,0,N
```

- **`%vc_static_shell> report_pst_state`**

```
PST State            : OFF_2
Full Name            : chip_top/OFF_2
        VDD| VDD1| VDD2| VSS|
       chip_top/OFF_2:  HV|   HV|  OFF|  ON|
       ------------------------------------------------------------------------
       PST State            : ON
       Full Name            : chip_top/ON
                               VDD| VDD1| VDD2| VSS|
       chip_top/ON:  HV|   HV|   HV|  ON|
       ------------------------------------------------------------------------
```

`%vc_static_shell> report_pst_state -only_transient`

```
       PST State            : OFF_1
       Full Name            : chip_top/OFF_1
                               VDD| VDD1| VDD2| VSS|
      chip_top/OFF_1:  HV|  OFF|   HV|  ON|
```

```
------------------------------------------------------------------------
```

- **%vc_static_shell> get_supply_net_combinations VDD1 VDD2**

  ```
  {"HV OFF", "HV HV"}
  ```

  %vc_static_shell> get_supply_net_combinations VDD1 VDD2 -only_transient

  ```
  {"OFF HV"}
  ```

- **%vc_static_shell> get_supply_rail_order VDD VDD1**

  ```
  0
  ```

  %vc_static_shell> get_supply_rail_order VDD VDD1 -transient

  ```
  1
  ```

1.1.2 New Violations tags Introduced

VC LP has introduced the COMPAT_PST_TRANSIENT violation to check for transient states. This violation is reported when there is information regarding a state of PST which is defined as a transient state.

This violation is enabled by default and its severity is warning. It is reported at UPF stage and in family Compatibility.

The following is an example of the violation snippet:

```
Tag         : COMPAT_PST_TRANSIENT
Description : Design has power state tables with -transient
```

1.1.2.1 Impact on Existing Checks

VC LP performs the following checks for transient state and rests of the checks are ignored.

- PG_DIODE_STATE
- PG_DIODE_INSUFFICIENT
- CORR_CONTROL_STATE
- CORR_CONTROL_STATE_WITHISO
- CORR_CONTROL_STATE_NOISO
- CORR_CONTROL_ISO

1.1.3 Limitations

- Wild-card is not supported in the `-state` switch of the `set_pst_transient` command.
- The transient state support does not work in the save/restore session feature. It works with the checkpoint/restart session feature.
- VC LP reports the PST_STATE_INVALID violation for transient states as they are not considered as permanent states.

1.2 The LS_OUTPWR_PATH Violation

Starting with this release, the LS_OUTPWR_PATH violation is introduced. This violation is disabled by default. This violation is reported during the `check_lp -stage DESIGN` stage.

There may be multiple unrelated supplies that have equivalent PST states in the UPF. However, in a complex real world designs, these unrelated supplies may have different noise characteristics (digital vs analog) or may drift separately due to temperature. When choosing a level shifter supply, it is safer to use

one which is related to the sink it is driving. These supplies have similar noise and drift behavior and reduces the chance of transient errors due to noise or voltage. This check helps users to identify design instances which do not use the same or derived supplies.

Definition of parent and sink for this violation

- **Parent**: The parent of a switched supply is either the UPF `create_power_switch -input_supply_port` supply or the input supply of a PSW instance in the netlist. In the UPF only one `-input_supply_port` and `-output_supply_port` should be defined. Each switched supply must have only one parent.
- **Sink**: The violation considers it's immediate sink (not the logic sink) for the check. Therefore, the sink can be an LP cell/buffer and so on in this check.

This violation is reported if the LS output supply is not electrically connected (through the UPF or PG connections) to the parent supply of the immediate sink. The violation is not reported:

- If the sink supply is directly connected to the LS output supply as it is a valid solution.
- If the sink supply is not connected to a power switch output (UPF or PG) because connecting the sink parent supply to LS output is not possible in this case.

1.2.1 Limitations

If physical switches are implemented, the tool assumes that only one power switch output is connected to the sink. If multiple power switches are connected to the sink using different input supplies, the algorithm considers the first switch supply mentioned in the design.

Power switch strategies with multiple input supplies are not supported in the current implementation. You must not use more than one `-input_supply_port` statements in each power switch. Otherwise, none of the input supply ports are considered for this check.

1.2.2 Impact on Existing Tags

When the `enable_strict_ls_rail_matching` application variable is enabled, the LS_OUTPWR_CONN covers a superset of the LS_OUTPWR_PATH violation. Hence, the violation LS_OUTPWR_PATH violation is not reported when the LS_OUTPWR_CONN is reported.

1.3 Support for New Syntax of the associate_supply_set Command

Starting with this release, VC LP supports the new UPF 3.0 syntax for `associate_supply_set` command.

Syntax

```
associate_supply_set supply_set_name_list [ -handle supply_set_handle ]
```

- The UPF 3.0 syntax allows a list of supply sets or supply set handles to be specified, and also the `-handle` switch is now optional.
- If the associated supply sets are updated with different supply nets in the same scope, then VC LP reports the UPF_SSET_ASSOC_INVALID error.
- A group of supply sets can be associated only when the group contains a supply in their common ancestor scope. If no common ancestor is found, then VC LP reports the UPF_SSET_COMMON_HIER_ERR message.

- To change severity of message UPF_SSET_COMMON_HIER_ERR, you can use the `upf_enable_strict_supply_set_check` application variable. By default, its severity is warning. If you want to change the severity of the UPF_SSET_COMMON_HIER_ERR to error, set the `upf_enable_strict_supply_set_check` application variable to true.

Example 1

In the following UPF snippet, two supply sets are associated with each other.

```
create_supply_set SS1 -function {power VDD1}
create_supply_set SS2 -function {power VDD2}
associate_supply_set {SS1 SS2}
```

VC LP reports the *[Error] UPF_SSET_ASSOC_INVALID* message as both the supply sets have different supply nets defined in the same scope.

Example 2

In the following UPF snippet, the associated supply sets are defined with different supply net handles (not different supply nets). Therefore, VC LP does not report an error in this case.

```
create_power_domain TOP
create_supply set SS3
create_supply_set SS1 -function {power TOP.primary.power}
create_supply_set SS2 -function {power SS3.power}
associate_supply_set {SS1 SS2}
```

Example 3

In the following UPF snippet, different scope level supply sets are associated in the top scope and there is no supply set mentioned in `associate_supply_set` command that is created in common ancestor scope.

```
set_scope CORE1
create_supply_set SS1
set_scope ..
set_scope CORE2
create_supply_set SS2
set_scope ..
associate_supply_set {CORE1/SS1 CORE2/SS2}
```

Therefore, VC LP reports the UPF_SSET_COMMON_HIER_ERR message, and you can change the severity of this message using the `upf_enable_strict_supply_set_check` application variable.

Example 4

In the following UPF snippet, the scope level supply set and top level supply set are associated in the top scope.

```
set_scope CORE1
create_supply_set SS1
set_scope ..
create_supply_set SS2
associate_supply_set {CORE1/SS1 SS2}
```

There is common ancestor present as supply set SS2, therefore, VC LP does not report any error message.

1.3.1 Changes to the Supply Set Consistency Checks

Prior to this release, VC LP was reporting *[Warning] message UPF_SSET_FUNCTION_ASSOCIATED* when the supply set function is updated with different supply net or with the same supply net.

Starting with this release, VC LP reports the new *UPF_SSET_FUNC_ASSOCIATED* error message for this scenario.

1.3.2 Deprecated Supply Set Checks

Starting with this release, the following two supply set checks are removed:

- UPF_SSET_ALREADY_ASSOCIATED: Supply set has already been associated

  Consider the following UPF snippet:

  ```
  associate_supply_set SS1 -handle PD1.primary
  associate_supply_set SS2 -handle PD1.primary
  ```

  Previously, VC LP was reporting the UPF_SSET_ALREADY_ASSOCIATED error message for this example. Starting with this release, VC LP does not report error because this is equivalent to `associate_supply_set { SS1 SS2 PD1.primary }`.

- UPF_SSET_ASSOCIATION_LOOP: Supply Set association creates a loop

  Consider the following UPF snippet:

  ```
  associate_supply_set Top.primary -handle PD1.primary
  associate_supply_set PD1.primary -handle Top.primary
  ```

  Previously, VC LP was reporting the UPF_SSET_ASSOCIATION_LOOP error message for this scenario. Starting with this release, VC LP does not report this error because this is equivalent to `associate_supply_set { Top.primary PD1.primary }`.

1.4 Support for Multi-threaded PG checks

By default, the PG checks run serially. VC LP has introduced the `enable_parallel_pg_checker` application variable. When this application variable is set, you can run the PG checks in a multi-threaded mode. In the multi-threaded mode, the performance in terms of the run time is improved as compared to the serial run. The multi-threading of PG checks take place while performing the `check_lp -stage PG` command.

```
%vc_static_shell> set_app_var enable_parallel_pg_checker true
```

Currently, multi-threading of PG checks is applicable on all tags except PG_VOLTMAP_INCONSISTENT and PG_BIAS_* checks.

Multi-threading can also be enabled using the `-j` switch with `check_lp` as shown below:

```
check_lp -stage <upf/design/pg/all> -j <num>
```

Where <num> can take values from 2-32, and the recommended values are between 4-12.

This feature requires a new license key (VT-VC-BETA). If this feature is enabled in the VC LP run, the VT-VC-BETA license key is checked-out.

- If license checkout is successful, then corresponding engine (`check_upf, check_lp, check_design, check_pg`) runs, and uses the parallelization feature.
- If the license checkout is unsuccessful, the engine throws an error message (unable to checkout license) and aborts.

1.5 Support for Multi-threading in Crossover Database Generation and Architectural checks"

By default, the crossover generation and architectural checks happens serially in VC LP. Starting with this release, the crossover generation and architectural checks can be parallelized thus reducing the time taken for performing crossover centric checks, Architectural checks, and resulting in significant improvement in run time.

To parallelize crossover generation, set the following application variables:

```
set_app_var enable_xover_multithread_lp true
```

Multi-threading can also be enabled using the `-j` switch with `check_lp` as shown below:

```
check_lp -stage <upf/design/pg/all> -j <num>
```

Where <num> can take values from 2-32, and the recommended values are between 4-12.

This feature requires a new license key (VT-VC-BETA). If this feature is enabled in the VC LP run, the VT-VC-BETA license key is checked-out.

- If license checkout is successful, then corresponding engine (`check_upf, check_lp, check_design, check_pg`) runs, and uses the parallelization feature.
- If the license checkout is unsuccessful, the engine throws an error message (unable to checkout license) and aborts.

1.6 Support for Multi-threading in Infer_source

By default, infer_source happens serially in VC LP. Starting with this release, the infer_source can be parallelized, reducing the time taken, and resulting in significant improvement in run time. To parallelize infer_source, -j option can be used with number of threads.

Syntax

```
infer_source -j  <num >
```

Where <num> can take values from 2-32, and the recommended values are between 4-12.

This feature requires a new license key (VT-VC-BETA). If this feature is enabled in the VC LP run, the VT-VC-BETA license key is checked-out.

- If license checkout is successful, then corresponding engine (infer_source) runs, and uses the parallelization feature.
- If the license checkout is unsuccessful, the engine throws an error message (unable to checkout license) and aborts.

1.7 Support for contain_switches Design Attribute

Starting with this release, VC LP supports the `contain_switches` design attribute. This support is added under the `upf_enable_mv_handling_psw_signals` application variable. By default, this application variable is set to false.

This feature helps to identify the hierarchy in which the indicated strategy must be implemented.

Syntax

```
set_design_attributes -elements {hierarhical_name} -attribute contain_sw
{psw_strategy_name}
```

1.7.1 Impacted Tags

If a PSW cell related to psw_strategy_name is present in some hierarchy other than mentioned in the `-elements`, then VC LP reports an existing violation PSW_LOCATION_WRONG providing the reason in the *ReasonCode* debug field.

The following are the other tags that are affected for this support:

- PSW_LOCATION_WRONG
- PG_CSN_CONN
- PSW_OUTPUT_CONN
- PG_STRATEGY_CONN

1.7.2 Parser Messages for contain_switches Design Attribute

- If an instance which does not exist in the design is provided in the `-element` list of the `set_design_attributes` command, then VC LP reports the UPF_OBJECT_NOT_FOUND error.

  ```
  set_design_attributes -elements {do_not_exist} -attribute contain_switches {PSW1}
  ```

  [Error] UPF_OBJECTS_NOT_FOUND: No valid objects in list

  Object List specified with command option 'set_design_attributes -elements' resolved to an empty list.

- If an element which is not in the same power domain as the PSW strategy is used in the `-elements` option of the `set_design_attributes` command, then VC LP reports the following parsing message and ignores the command:

  ```
  set_design_attributes -elements {CORE1} -attribute contain_switches {PSW1} //##
  ```
  CORE1 belong to PD1 domain and PSW1 is written for TOP domain

  [Warning] ELEM_NOT_IN_EXTENT_OF_SWITCH_DOMAIN_IN_DESIGN_ATTR: Element 'CORE1' specified in set_design_attributes command is not in the extent of domain 'TOP' of power switch strategy 'PSW1' in which switch cell will be placed, switch ignored for rule checking

  Please specify that element whose hierarchy is in the extent of power switch strategy domain

- If a lib cell is provided in the `-elements` list of the `c` command, then VC LP reports the parsing message and ignores the `set_design_attributes` command:

  ```
  set_design_attributes -elements {LIBCELL} -attribute contain_switches {PSW1}
  ```

  [Warning] LIBCELL_ELEM_IN_DESIGN_ATTR: Libcell instance 'and_i1' specified in 'set_design_attributes' command with attribute name 'contain_switches', command ignored for rule checking

  Please specify only hierarchical module with -elements in 'set_design_attributes' used with 'contain_switches' attribute

- If a black box instance is provided in the `-elements` list of the `set_design_attributes` command, then VC LP reports the parsing message and ignores the `set_design_attributes` command:

  ```
  set_design_attributes -elements {BBOX} -attribute contain_switches {PSW1}
  ```

  [Warning] BBOX_ELEM_IN_DESIGN_ATTR: Blackbox instance 'BBOX' specified in 'set_design_attributes' command with attribute name 'contain_switches', command ignored for rule checking

  Please specify only hierarchical module with -elements in 'set_design_attributes' used with 'contain_switches' attribute

- If multiple `set_design_attributes` commands are present for the same PSW strategy, then VC LP reports parsing message and honors the first command:

```
set_design_attributes -elements {CORE1} -attribute contain_switches {PSW1}
set_design_attributes -elements {CORE2} -attribute contain_switches {PSW1}
```

[Info] SAME_SWITCH_WITH_MULTI_ELEM_IN_DESIGN_ATTR: Switch name 'PSW1' is specified with multiple elements in 'set_design_attributes' command with attribute name 'contain_switches', switch is associated with first element 'CORE1'

- If a PSW strategy has multiple input ports but the `-supply_set` option is not provided, then the following error is reported:

  [Error] SUPPLY_SET_MISSING_IN_MULTI_INPUT_PSW: Argument -supply_set missing in power switch strategy 'PSW1' with multiple input supply nets, switch ignored for rule checking

  Please specify the supply set in power switch strategy with multiple input supply nets specification

1.7.3 Example

Consider a design where both *top* and *CORE1* hierarchy belong to the power domain *TOP*. There is a PSW strategy *PSW1* written for domain *TOP*, and the related instance *psw_i1* is present in the *top* hierarchy.

The following is the UPF snippet:

```
set_design_attributes -elements {CORE1} -attribute contain_switches {PSW1}
…
…
create_power_switch PSW1 -domain TOP \
       -output_supply_port {VDDSW VDD1} \
       -input_supply_port {VDD VDD} \
       -control_port {NSLEEPIN1 EN} \
       -on_state {poweron VDD {!NSLEEPIN1}} \
       -ack_port {NSLEEPOUT1 ACK {NSLEEPIN1}}
```

The following violation is reported for this example.

```
Tag               : PSW_LOCATION_WRONG
  Description       : Power Switch cell [Instance] for strategy [Strategy] is
instantiated in wrong logical Location
  Instance          : psw_i1
  Strategy          : PSW1
  Cell              : DMPSWHEAD
  LocationMismatch
    UPFLocation     : top
    UPFScope        : top
    DesignScope     : top
  ReasonCode          : Design location does not match with location CORE1 specified in
set_design_attributes command for switch
```

1.8 Support for main_rail Attribute

You can implement parts of shutdown voltage area with always_on cells. You can define always_on paths of a shutdown domain in a different power domain called always_on domain. The support for synthesizing always_on cells are available under the `enable_main_rail_analysis` (default false) application variable.

The always_on domain has two main implicit supply sets:

- Primary supply set: This is the actual supply set of the always_on cells which is connected to the backup power pins of the always_on cells.

- main_rail supply set: The main_rail supply set represent the dummy primary supply rails of the always_on domain.

During place and route, cells in always_on power domains can be sprinkled in other power domains, which have primary supply sets same as the always_on domain main_rail supply set.

Note
The features described in the above section is currently under development. Please contact Synopsys team to check the readiness of the feature before using it.

1.8.1 Use Model

Use the main_rail keyword to define always_on block domain, which will resite in other domain.

```
create_supply_set SS_sw
create_supply_set SS_aon
create_power_domain PD_sw -supply {primary SS_sw}
create_power_domain PD_aon -supply {primary SS_aon} \
                           -supply {main_rail SS_sw}
```

You can also choose to specify main_rail supply later (during implementation). At the time of RTL verification, you must specify only the primary supply. The `main_rail` supply can be specified later using the `-update` option.

It is recommended to use full RTL designs, or designs with only AON cells within AON domains. Expectations of using LP cells, non-AON cells, single rail cells and so on inside AON domains are still under discussion. It is recommended to use the same bias nets for both `main_rail,` and the primary supply sets on the AON domain.

1.8.2 Implicit connections

VC LP adds the following implicit connections in AON domains, if there are no explicit connections available to the cells.

- Primary power/ground pins of single rail cells is implicitly connected to AON domain primary power/ground nets.
- Primary power/ground pins of dual rail cells is implicitly connected to AON domain main_rail power/ground nets.
- Backup power/ground pins of dual rail cells is implicitly connected to AON domain primary power/ground nets.

1.8.3 Support for JSON Dump

You can save the JSON dump of the power_domain UPF setting with main_rail supply set using the `dump_lp_db -component upfdatamodel` command.

This command saves the *vcst_rtdb/lpdb//powerdomaindump.json* file for power_domain UPF setting with main_rail supply set.

1.8.4 TCL Command Support

Starting with this release, the `-main_rail` option is added in the following Tcl commands:

- `report_power_domain <domain> -main_rail`: This command returns the `main_rail` supply set of the given power domain.

- `get_power_domain_elements <domain> -main_rail`: This command returns the `main_rail` supply set of the specified power domain.
- `get_supply_sets -of_object <domain> -main_rail`: This command returns the `main_rail` supply set of the specified power domain.

1.8.5 VC LP Checks

The following violation tags are introduced for this support:

- UPF_MAINRAIL_BIAS

  This violation is reported when the power on the bias nets of main rail supply set is less than the power and ground nets of the primary supply. The On ness of the nets are compared as nwell with power net and pwell with ground net.

- UPF_MAINRAIL_AON

  Each AON cell in an AON domain should have at least one backup power or backup ground pin. The UPF_MAINRAIL_AON violation is reported if at least one backup pin is not specified in the always_on lib cells in an AON domain.

- UPF_MAINRAIL_CSN

  For each always_on lib cell in the AON domain, at least one connect_supply_net command is expected, else the UPF_MAINRAIL_CSN violation is reported.

- PG_MAINRAIL_CONN

  For each always_on lib cell in the domain, if explicit PG connection exists for the lib cell, the primary pg pin of the always_on cell must match the main_rail supply of the power domain., else the PG_MAINRAIL_CONN violation is reported.

- PG_DOMAIN_CONN

  This violation is reported for the backup pins of the AON cells if they are not connected to the domain primary supply set.

1.8.6 UPF Passer Messages Support

The following parser messages are reported:

- UPF_MAINRAIL_IN_UPDATE

  This message is reported if the `main_rail` is specified with the `-update` option, and the -update value is not specified yet.

- UPF_MAINRAIL_ALREADY_DEFINED

  This warning is reported if the `main_rail` is specified with the `-update`, and it is already specified before UPF_MAINRAIL_ALREADY_DEFINED warning is triggered, and the updated value is ignored.

1.9 Support for -source/-sink in the set_level_shifter Command

VC LP supports the `-source` and `-sink` options in the `set_level_shifter` command. These options are supported when the `enable_src_sink_for_level_shifter` application variable is set to true.

By default, this application variable is set to false and VC LP does not support the `-source` and `-sink` options in the `set_level_shifter` UPF command. When such level shifter strategies was found, VC LP reports parsing warning TCL_OPT_NYI message and ignore these options.

When the `enable_src_sink_for_level_shifter` application variable is set to true, VC LP performs checks on strategies with the `-source` and `-sink` option.

For the level shifter strategies, supplies written with options `-source` and `-sink` refers to supplies of LogicSource and LogicSink respectively. You can provide the supply set name or the power domain name in the `-source` and `-sink` options.

Example 1

Consider the following UPF snippet:

```
set_level_shifter LS1 -domain PD1 -source SS_PD1 -sink SS_PD2 -location parent

## Below strategy will also give same results
##set_level_shifter LS1_using_power_domain -domain PD1 -source PD1 -sink PD2 -location
parent
```

There is a crossover from CORE1/and_i1/Z to CORE2/ff_i1/D. The source and sink belongs to power domain PD1 and PD2 respectively. PD1 is working on 1.0V (Supply set SS_PD1 having supply Nets: VDD1, VSS1). PD2 is working on 1.2V (Supply set SS_PD2 having supply Nets: VDD2, VSS2).

Level Shifter strategy written with `-source` and `-sink` options is required on the path, therefore, VC LP reports the following violations:

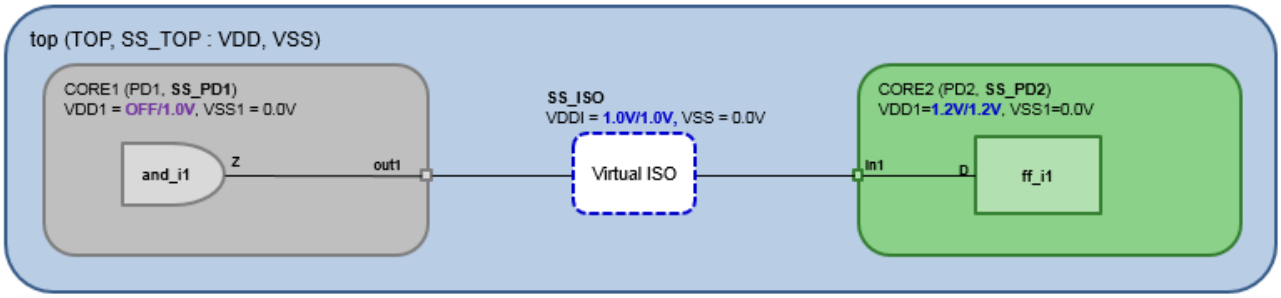
- LS_STRATEGY_OK
- LS_STRATEGY_UNUSED
- LS_INST_MISSING (LH Type)

Example 2

For the cases where ISO strategy is present on the path along with LS strategy, VC LP adds virtual ISO and breaks the whole crossover into segments and does analysis for each segment separately for LS checks in the UPF stage. This support can be enabled when the `enable_virtual_iso_insertion` application variable is set to true.

Consider the following UPF snippet:

```
set_isolation ISO1 -domain PD1 -source SS_PD1 -isolation_supply_set SS_ISO -location
parent
set_level_shifter LS1 -domain PD1 -source SS_PD1 -sink SS_PD2 -location parent
```



There is a crossover from CORE1/and_i1/Z to CORE2/ff_i1/D. Source and Sink belongs to power domain PD1 and PD2 respectively. Source can be OFF or ON with 1.0V (Supply set SS_PD1 having supply Nets: VDD1, VSS1). Sink is always ON with 1.2V (Supply set SS_PD2 having supply Nets: VDD2, VSS2).

Both ISO and LS strategies are required on path. VC LP applies the virtual ISO concept for the LS checks in the UPF stage. For the reporting purpose, the virtual ISO is assumed to be on StrategyNode that is CORE1/out1 in this case. The following two new debug fields enables you to distinguish a Virtual ISO node, and it resolves the supply from the ISO policy:

- `Source`

  ```
  PinName          : CORE1/out1
  IsVirtual        : True
  ```

- `SegmentSinkVirtual  : True`

VC LP applies the strategy on the whole crossover and break it into segments:

- Source > Virtual ISO `[CORE1/and_i1/Z > CORE1/out1]`
- Virtual ISO > Sink `[CORE1/out1 > CORE2/ff_i1/D]`

For both the segments, analysis is done and violations are reported separately, and VC LP reports the following violations:

- LS_STRATEGY_REDUND `[CORE1/and_i1/Z > CORE1/out1]`

```
Tag                 : LS_STRATEGY_REDUND
  Description       : Level shifter not required, but level shifter strategy
[Strategy] defined
  Violation         : LP:3
  Strategy          : PD1/LS1
  StrategyNode      : CORE1/out1
  Source
    PinName         : CORE1/and_i1/Z
  SegmentSourceDomain : PD1
  Sink              : CORE1/out1
  SegmentSinkVirtual  : True
  SegmentSinkDomain   : PD1
  DomainSource      : CORE1/out1
  DomainSink        : CORE2/in1
  SourceInfo
    PowerNet
      NetName       : VDD1
      NetType       : UPF
    PowerMethod     : FROM_UPF_POWER_DOMAIN
    GroundNet
      NetName       : VSS1
      NetType       : UPF
```

```
    GroundMethod        : FROM_UPF_POWER_DOMAIN
  SinkInfo
    PowerNet
      NetName           : VDDI
      NetType           : UPF
    PowerMethod         : FROM_UPF_POLICY
    GroundNet
      NetName           : VSS
      NetType           : UPF
    GroundMethod        : FROM_UPF_POLICY
```

- LS_STRATEGY_OK [CORE1/out1 > CORE2/ff_i1/D]

```
Tag                   : LS_STRATEGY_OK
  Description           : Correct level shifter strategy [Strategy]
  Violation             : LP:2
  Strategy              : PD1/LS1
  StrategyNode          : CORE1/out1
  Source
    PinName             : CORE1/out1
    IsVirtual           : True
  SegmentSourceDomain   : PD1
  Sink                  : CORE2/ff_i1/D
  SegmentSinkDomain     : PD2
  LogicSource
    PinName             : CORE1/and_i1/Z
  LogicSink             : CORE2/ff_i1/D
  DomainSource          : CORE1/out1
  DomainSink            : CORE2/in1
  SourceInfo
    PowerNet
      NetName           : VDDI
      NetType           : UPF
    PowerMethod         : FROM_UPF_POLICY
    GroundNet
      NetName           : VSS
      NetType           : UPF
    GroundMethod        : FROM_UPF_POLICY
  SinkInfo
    PowerNet
      NetName           : VDD2
      NetType           : UPF
    PowerMethod         : FROM_UPF_POWER_DOMAIN
    GroundNet
      NetName           : VSS2
      NetType           : UPF
    GroundMethod        : FROM_UPF_POWER_DOMAIN
```

- LS_STRATEGY_UNUSED [CORE1/and_i1/Z > CORE2/ff_i1/D]
- LS_INST_MISSING [CORE1/and_i1/Z > CORE2/ff_i1/D]

1.9.1 Precedence Rules for Choosing Level Shifter Strategies for a Port

VC LP analyzes the level shifter strategies based on the following filters and chooses the strategy based on the set precedences. Among multiple strategies present on a particular port, the most specific strategy gets associated with that port. The filters are mentioned in the decreasing order of precedence:

- *Granularity based filter: [-elements option containing Port > Instance > Domain]*

Strategies written by specifying ports in the `-elements` of the strategy gets 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.

- *force_shift based filter [-force_shift > (no –force_shift)]:*

  Strategies written with `-force_shift` gets higher priority over strategies that do not have it.

- *no_shift based filter [-no_shift > (no –no_shift)]:*

  Strategies written with `-no_shift` gets higher priority over strategies that do not have it.

- *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 strategies having exactly one, that is, either `-source` or `-sink`. Strategies which are not having any of the option `-source`/`-sink` gets the least precedence.

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.

1.10 Support for Running VC LP from VCS

You can run VC LP from the `vcs` command line using the `-vclp` switch. This support helps the NLP users to reduce their effort while debugging simulations. This support is also available for pre-compiled flow.

1.10.1 Use model

To run VC LP from the VCS command line, use the `-vclp` switch:

```
vcs bench.v chip.v -power_top top -upf top.upf -vclp
```

The `-vclp` switch extracts information from the vcs command line, writes a Tcl script, runs VC LP, and saves the report file and checkpoint. You can then review the text report, or load the checkpoint, or use the generated script as a starting point for your analysis.

Example

The following is an example VC LP Tcl script:

```
set vcs_opts {bench.v test.v }
read_file -single_step -vcs $vcs_opts -top top
read_upf top.upf
check_lp -stage upf
report_lp ;# show summary on screen
checkpoint_session -session vclpSimv.daidir/run_vclp
quit
```

Note

The -power_top /-upf options are mandatory options to run VC LP in vcs command line.

1.10.2 Providing Additional Information for VC LP

You can provide addition information like `search_path`, `link_library`, and `set_blackbox` in a configuration file using the `-power_config` switch in `vcs` command line option.

Example

```
db_search_path = { /remote/pv_dbs/dc_nlp_testing/pgdb}
db_link_library = {/demo_iso_10v_pg.db /demo_iso_12v_pg.db}
set_power_black_box -model <model> [-type DARK_BOX|GREY_BOX|POWER_BOX|IP_BOUNDARY]
```

1.10.3 Considerations for Reading Verilog/VHDL Files

- In customer simulation, verilog have several different versions of a module or process, each inside a different #ifdef, and then selects the correct version at runtime using +define+. All the options from the vcs command line (except -power_top, -upf) are simply passed back to vcs.

  ```
  vcs bench.v chip.v -power_top top -upf top.upf +define+UPF
  Results in these lines in the vclp script:
  set vcs_opts "bench.v chip.v +define+UPF"
  read_file -single_step -vcs $vcs_opts -top top
  ```

- Options including `+define+SYNTHESIS`, `skip_translate_body`, and `undefine VCS` cause the behavior of vcs to differ, compared to a vcs standalone run. Each of these options is controlled by an existing application variables in VC Static.

  ```
  set_app_var define_synthesis_macro false
  set_app_var honor_vcs_macro true
  set_app_var analyze_skip_translate_body false
  ```

- Simulation code has assertions, which are not synthesizable. VC Static passes the `deleteSVA` switch to simon, and therefore all the assertion logic is deleted.
- Simon will black box and glass box the non-synthesizable constructs.

1.10.4 Considerations for UPF Options

- When you use different UPF files for different tools, you can use the following script:

  ```
  if {$synopsys_program_name == "vcs"}
     <sim upf>
  else
     <synth upf>
  ```

  In this case, the `vcs -vclp` switch reads the `<synth upf>` code, because it is parsed in VC Static, and the value of the `synopsys_program_name` application variable is `vcst`. To successfully read `<sim upf>`, it is necessary to override the value of this application variable to `vcs`. This can be set using the following application variable:

  ```
  set_app_var upf_tool vcs
  ```

- The design independent mode can be used by simply giving a command line `vcs -vclp -upf <file>.upf (no designs)`. When this mode is activated, VC LP runs a script containing the equivalent VC LP command:

  ```
  read_upf -no_design <file>.upf
  ```

- There is the usage of *supply_net_type* ports defined in a UPF:: package in VCS NLP. These ports are basically structs with fields Supply State and Voltage value. These ports are used in simulation-specific designs and UPF. In the vcs-vclp flow, customers use the same UPF/Design which is used in VCS NLP flow. VC LP identifies such ports as special ports of type *supply_net_type* and treats them as a special data type and flagged violations.

1.10.5 Limitations

- When an instance in the chip refers to a signal in the testbench, VC Static elaboration at the chip scope reports an XMRE error and fails.

- Sometimes you may have standardized on one release of VCS, and a different incompatible release of VC Static. When you run designs in pre-compile flow, forward compatibility (VC Static 2017.03 reading VCS 2017.12) is not guaranteed.
- Partition compile flow not supported. When the `-partcomp` option is specified in `vcs` command line, the following warning is reported:

  Warning-[RUNVCLP_PARTCOMP_NYI] Partition compile is not yet implemented in vcs -vclp -power_top option is mandatory in vcs-vclp run

1.11 Support for SAM Hierarchical Flow for RTL Designs

VC LP has been supporting SAM (Static Abstraction Model) flow for netlist and pgnetlist. Starting with Q-2020.03, VC LP supports SAM flow for RTL designs. Hierarchical RTL designs with high runtime can adapt this flow and observe run time performance while protecting the QoR.

1.11.1 Use Model (Abstraction)

Netlist Flow

```
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
```

RTL Flow

```
set search_path ".  "
set link_library " demo_psw_10v_pg.db tiny.db"
configure_lp_abstract_model
read_file -top and_module -f verilog {module.v}
read_upf module.upf
create_lp_abstract_model
write_abstract_model -path abs
```

Note

In netlist, `mark_lp_abstraction` is not mandatory. You can use `check_lp` for marking. In RTL, `create_lp_abstract_model` is must.

1.11.2 Use Model (Readback)

Netlist Flow

```
set search_path ". "
set link_library " demo_psw_10v_pg.db tiny.db "
set_verilog_abstract_model -module and_module
read_file -netlist -top top -f verilog {top.v module_abs.v}
read_upf top.upf
check_lp -stage all
```

RTL Flow

```
set search_path ". "
set link_library " demo_psw_10v_pg.db tiny.db "
set_abstract_model -path abs -module and_module
```

```
read_file -top top -f verilog {top.v module.v}
read_upf top.upf
check_lp -stage upf
```

Notes

- This feature is supported under new license "VC-LP-ULTRA-RTL".
- Only debugging purposes, VC LP dumps the abstracted design in near matching Verilog format.

  `write_abstract_model -debug_verilog`
- In RTL SAM readback, original RTL designs are read in `read_file` and in elaboration stage, VC LP replaces the particular modules with abstracted designs.

Limitations

- RTL abstracted models are written in binary formats due to tool limitations.

1.12 Support for Viewing VC LP Violations in nSchema and nWave

Starting with this release, when debugging on a Verdi schematic or nWave, you can view the VC LP violations that exist on or near any specific instance or pin.

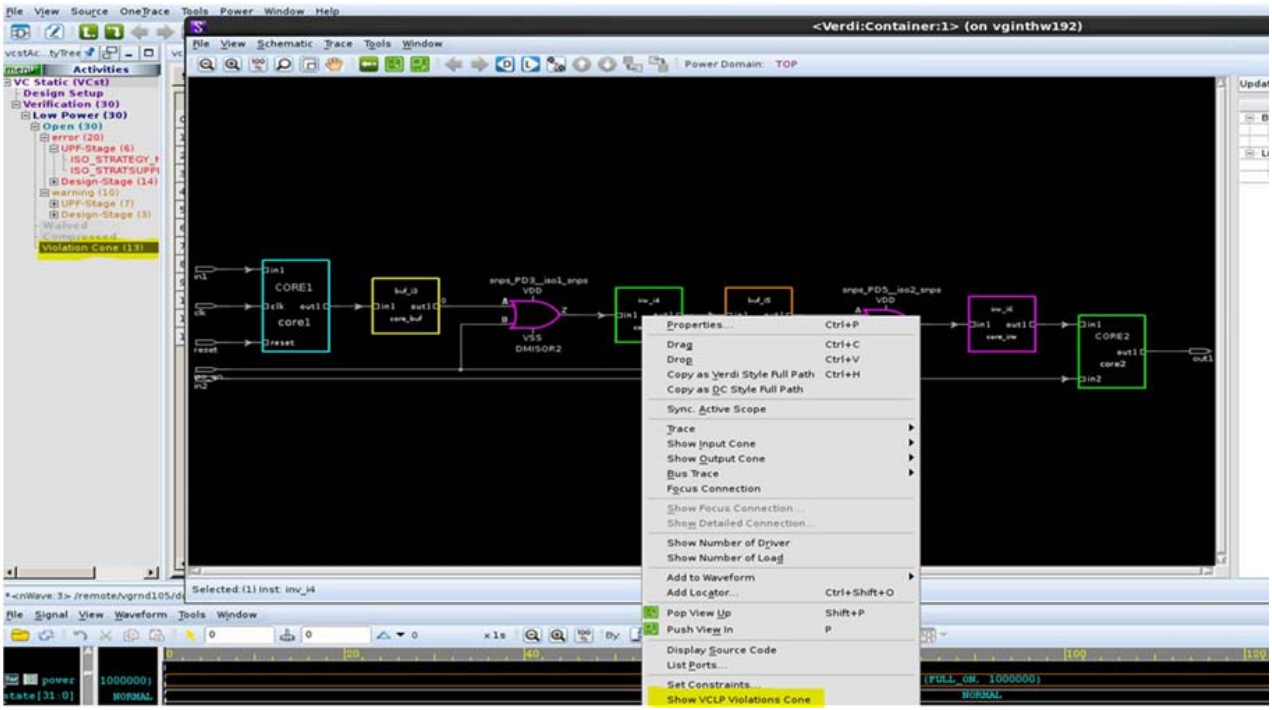
For example,

- In a schematic, you can find how many different VC LP violations contribute to a wrong buffer.
- In a waveform, if you find an X in simulation, you can use trace-x to find the source of the X, and you can also find the violations reported by VC LP.

To view the VC LP violation in nSchema/nWave:

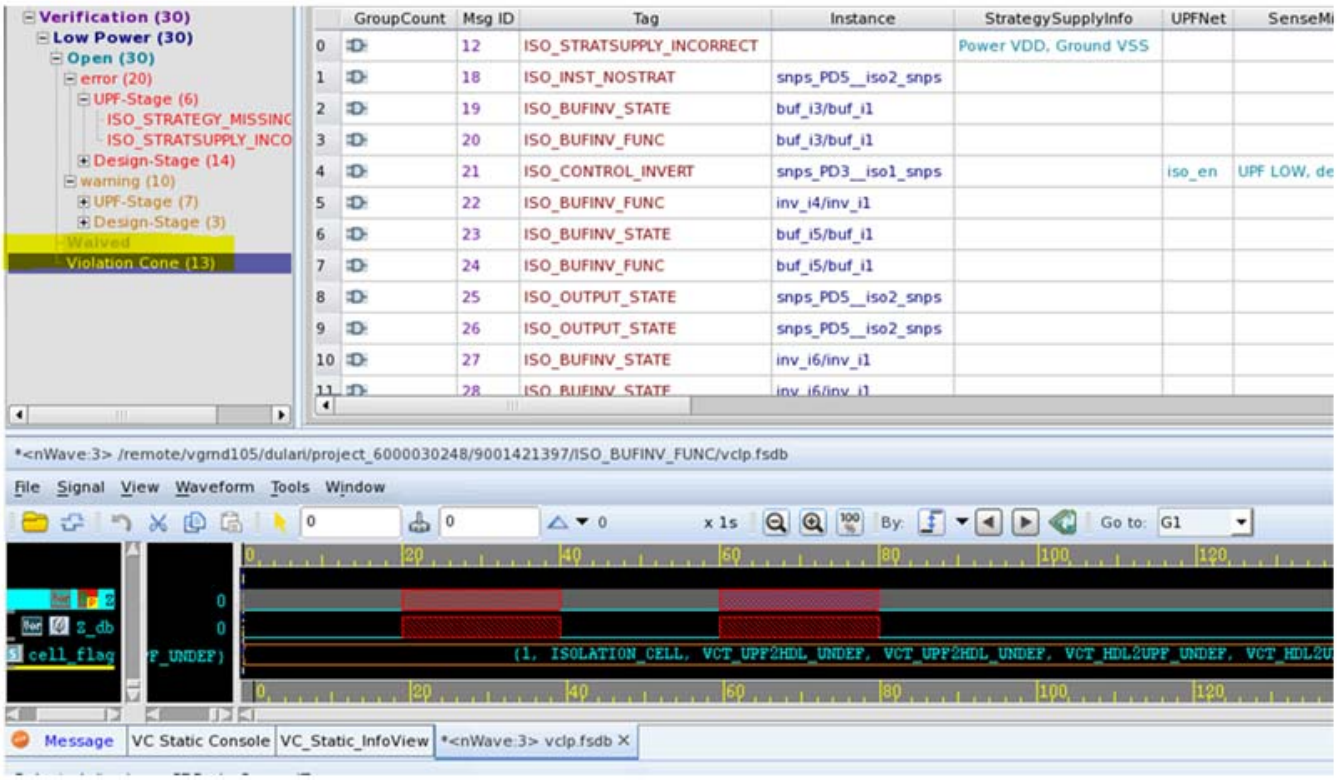
1. From the RMB menu of the selected signal in nSchema/nWave view, click **Show VCLP Violations Cone** to open VC LP to get the violation reports on the violation cone of the given signal.

Figure 1-1 Show VCLP Violations Cone Option in nSchema



2. The violations are reported in the violation cone in **nWave** as shown in the following figure:

Figure 1-2 Show VCLP Violations Cone Option in nWave



3. The violations are reported in the violation cone in **Activity view** as shown in the following figure:

Figure 1-3 Violation Cone in view_activity