cadence™

Conformal® Low Power User Guide

Product Version 23.2
October 2023

Cadence Design Systems, Inc., 2655 Seely Avenue, San Jose, CA 95134, USA

Contents

1

Introduction to Conformal Low Power

Overview

Designers increasingly expect longer battery life and higher performance. Due to increased leakage, devices created using 90-nanometer and smaller process nodes consume as much power when they are not in use as when they are being used. Designers can optimize for leakage and dynamic power, which reduces energy use and lowers cooling and packaging costs. Additional advanced low power methods offer further power savings, but significantly complicate the verification process.

The Encounter® Conformal® Low Power software enables you to verify correct implementation of these low power design techniques and validate your silicon using formal techniques (versus simulation) early in the design process. It also decreases the risk of missed bugs which are often missed, before a product goes out the door. Conformal Low Power is the only solution in the industry providing full-chip low power functional verification.

Conformal Low Power accepts RTL/gate-level netlists with or without explicit power or ground nets, and accepts user-defined power pins, power domains, power switches, level-shifter cells, isolation cells, and state retention cells.

Note: This manual mostly describes the CPF flow. For 1801 related support, refer to the chapter Chapter 7, “Running the Conformal Low Power 1801 Flow”.

Accessing Online Help and Documentation

Launching Cadence Help

The online documentation system is called Cadence Help.

From the main GUI, click on the Help menu item and navigate to the HTML version of the document that you wish to view. This will bring up Cadence Help.

Some GUI windows also have a Help button that will launch Cadence Help.

Getting Help for Cadence Help

After launching Cadence Help, press F1 or choose Help - Contents to display the help page for Cadence Help.

Getting Help on Commands to Run Tools

You can display a list of options for any of the tools and utilities by typing the tool or utility name followed by the -help option as follows:

```
% tool_name -help
```

Example:

```
% ccd -help
```

Getting Help on Commands and Messages

Use the MAN command without any options to list all of the available commands. However, to view specific help information, use the following commands:

- command_name—To view command usage for a specific command, enter the MAN command followed by the command name. For example:

  ```
  man read design
  ```

- -verbose—To view expanded information about a specific command, enter the MAN command, followed by the command name, and the -verbose option. For example:

  ```
  man read design -verbose
  ```

- message_name—To view help for a particular rule check message, enter the MAN command followed by the message name. For example:

  ```
  man f10
  ```

For more information on the MAN command, use the following command from within the tool:

```
%man man
```

Searching the Help Database for Specified Strings

The SEARCH command searches the Help database of commands and options for matches to strings you specify. Include the -usage option to display the command and its options.

Using the Help Menu

You can use the *Help* menu to get more information on commands, licenses, documentation, and Cadence support.

Accessing Help from Command Windows

A *Help* button is available in many command windows. Unlike the *Help* button on the main window, when you left-click the *Help* button in command windows, the Conformal software automatically executes *Help – Commands* and displays the information for the related command in the Command Help window.

Accessing User Documentation

Use the following procedure to view the user guides and reference manuals.

1. Click the *Help* pull-down menu located at the far right end of the menu bar.
2. Click *<Book Name> (pdf)* or *<Book Name> (html)*.

   The PDF reader launches and displays the PDF version of the book. Or, Cadence Help launches the HTML version. If you choose the HTML version, you will have access to all the other books in the documentation set through Cadence Help.

   Note: You must have a PDF reader to access the documentation. To download the current version of Adobe Acrobat Reader, visit the following web page:

   http://www.adobe.com/support/downloads/main.html

Accessing Product Information

Use the following procedure to display the Cadence company logo, the product version number and date, mailing address, phone and fax numbers, and web page and E-mail addresses.

1. Click the *Help* drop-down menu located at the far right end of the menu bar.
2. Click *About*.

Accessing License Information

From the *Help* drop-down menu in the main window, click *License* to view information regarding all the installed Conformal software licenses. The report appears in the Transcript window and includes information such as the current user, feature, and expiration date.

You can also use the `LICENSE` command to review the current license status. The current status of the license appears in the transcript output.

Additional Learning Resources

Cadence offers the following training courses on Conformal Low Power:

- *Low-Power Verification with Conformal*

Power Aware Equivalence Checking

Conformal Low Power Equivalency checking provides advanced equivalence checking for RTL designs to place-and-route netlists with low power implementation.

The following table lists the power aware equivalency checks performed at each verification stage (yes = performed; no = not performed).

	Verification Stage				
	Logic Function Compare	**ISO Insert Checks**	**Retention Control Function**	**Switch Control Function**	**Power Domain Compare**
RTL-to-gate equivalency checks	yes	yes	yes	yes	yes
Gate to place-and-route equivalency checks	yes	yes	yes	yes	yes

For more information, see Chapter 2, “Power Aware Equivalency Checks”.

Low Power Extended Checking

Conformal Low Power extended checking performs:

- Library-consistency checks
- Power-intent quality checks
- Structural checks
- User-defined, rule-based checks
- Functional checks
- Power-aware circuit analysis checks

Conformal Low Power extended checking also provides a power intent development environment (PIA). Users can use this environment for low power cell library creation/ modification and design power intent creation/modification.

Logical and Physical Netlists

Conformal Low Power logical netlist checking works on netlists without power and ground connections. It checks level shifter locations and isolation cell types and locations. Conformal Low Power physical netlist checking works on netlists with power and ground connections. It completes checks for level shifter, isolation, and state retention cells.

For more information, see Chapter 4, “Logical Netlist Checking” and Chapter 5, “Running Physical Netlist Checks”.

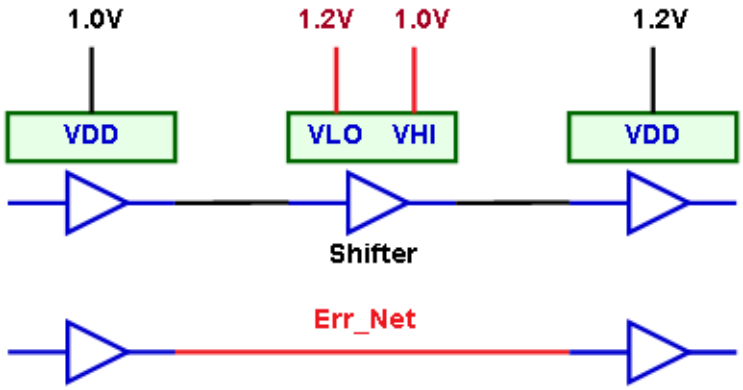
Structural Checking

A structural check is anything we can do that only involves looking at the connectivity and traversing through simple inverter and buffer gates. The structural check does not analyze any logic to do the checks.

- ISO7 checks flag missing isolation between switchable to non-switchable domains
- Primitives with no power domain or multiple power domain (data or setup problem)
- Nets with no drivers
- Obvious redundant isolation (between two non-switchable domains)
- Power ports not connected to a power domain

- Power and ground domain shorts
- Power and power domain shorts
- Level shifter power ports connection errors (to wrong voltage)
- Isolation cell power port connection (not to correct domain based on receiver)
- Power switch cell port checking

In the following example, the tool finds level shifter power-routing errors and the missing level shifter on net `Err_net`:



Rule-Based Checking

You can specify how you expect isolation or retention to be implemented by defining valid rules for checking isolation and state retention cells.

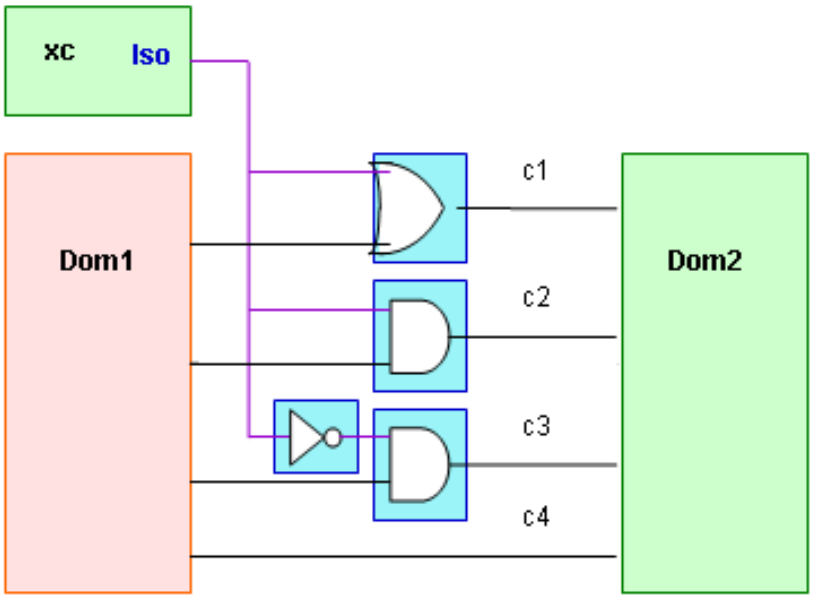
In the following example, the tool finds three errors:

- The isolation type of crossing `c1` is high when it should be low
- The isolation control polarity of crossing c2 is incorrect

- Crossing c4 does not have isolation

CPF isolation insertion rule:

```
create_isolation_rule
-from Dom1-to Dom2
-isolation_condition xc/Iso
-isolation_type low
```



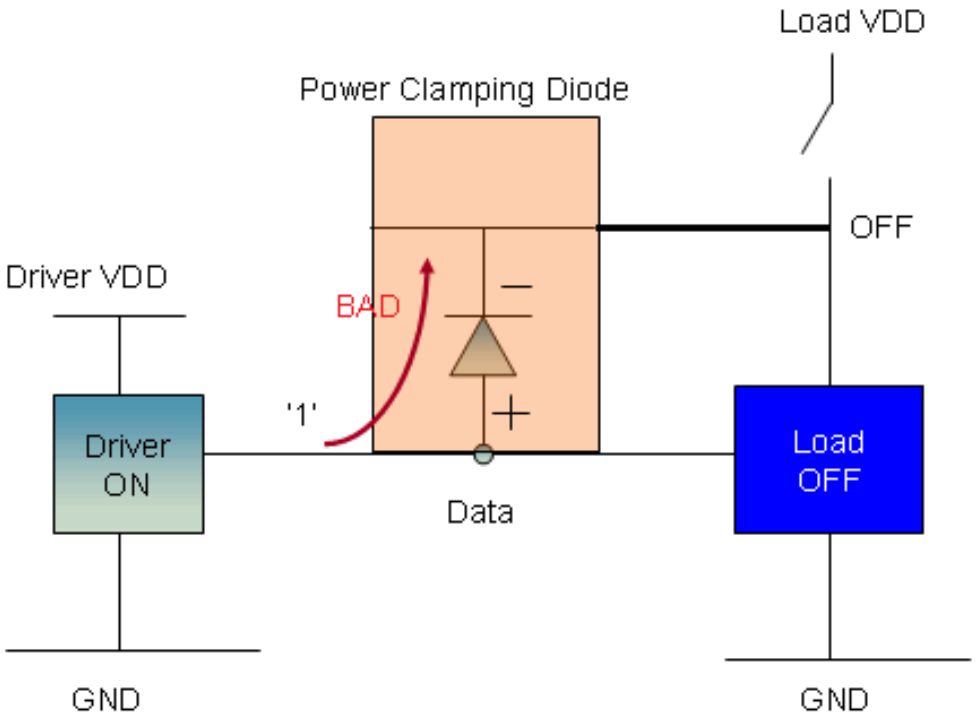
Circuit-Analysis Checking

Conformal Low Power extended checking includes the following power-aware, circuit analysis checks:

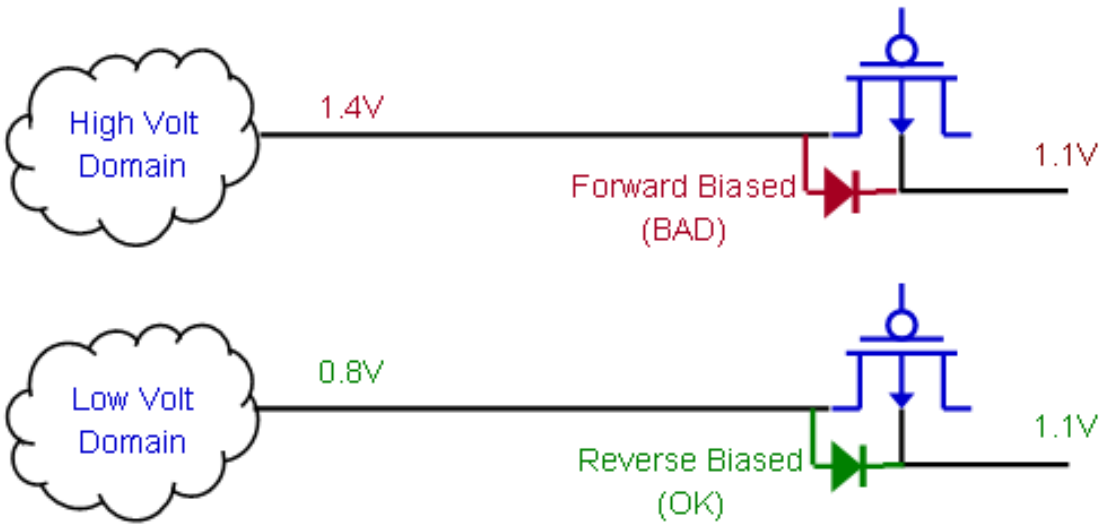
- Clamping diode checks
- Open-source input checks (requires a GXL license)
- Switch body connection checks (requires a GXL license)
- Substrate bias connection checks (requires a GXL license).

The following figures illustrate some of the power-aware, circuit analysis checks.

Example 1-1 Clamping Diode Checks



Example 1-2 Open-Source Checks



*There are unbuffered inputs at the High to Low voltage crossing

Common Power Format

The Common Power Format (CPF) is intended to address the current limitation in the design automation tool flow by enabling the capture of the designer's intent for advanced power management techniques. CPF provides support for all design and technology-related power constraints to be captured in a single file format for use throughout the RTL to GDSII design flow including verification, validation, synthesis, test, physical implementation, and signoff analysis.

The automation enabled through CPF infrastructure support will be the answer to the growing power management design challenges faced by the industry. The introduction of CPF and its support will bring productivity gains and improved quality of silicon to designers without requiring any change to current legacy RTL.

For more information, see the following documents for Common Power Format (CPF):

- *Common Power Format Language Reference*

  You can open this from the Main Conformal window by choosing *Help – CPF Reference Manual*.

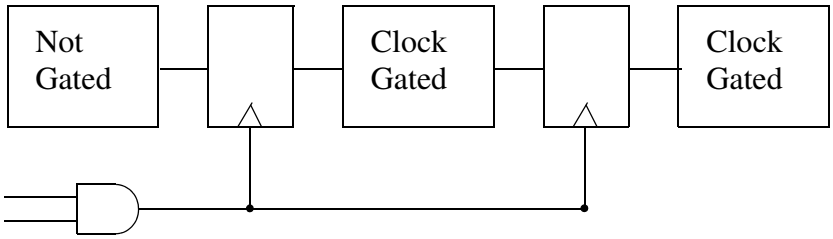
- *Common Power Format User Guide*

  You can open this from the Main Conformal window by choosing *Help – CPF User Guide*.

Low Power Concepts

Clock Gating

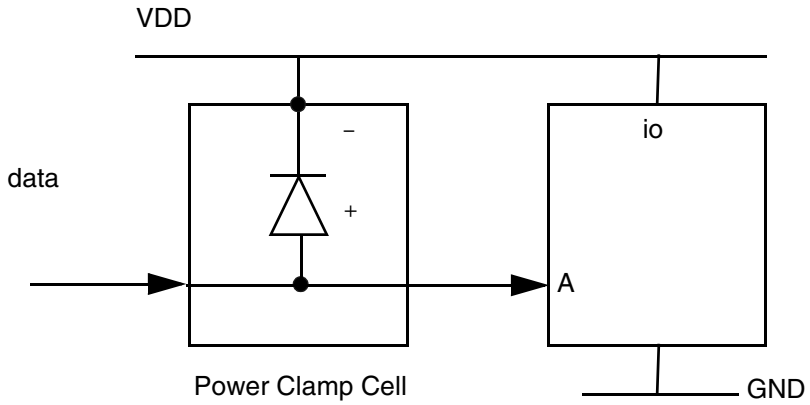
Clock Gating is when logic that receives signals from clock gated latches and registers do not switch. No dynamic power is consumed. Clock gating minimizes switching. In many designs, data is loaded into registers infrequently, but the clock signal continues to switch at every clock cycle often driving a large capacitive load. You can save a significant amount of power by identifying when the registers are inactive and by disabling the clock during these periods.



Diode Clamps

Power Clamp Cell

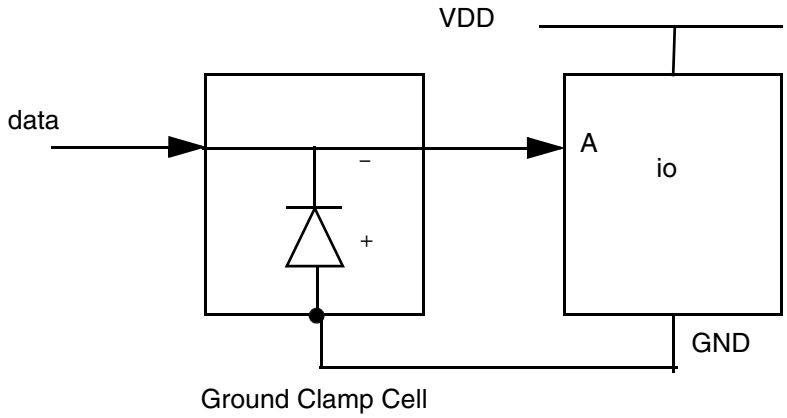
A power clamp cell has a diode connection to the power net from the data signal, as shown in the following diagram:



If the input signal voltage rises above VDD + Vth, the diode conducts, thereby clamping the signal voltage to VDD+Vth.

Ground Clamp Cell

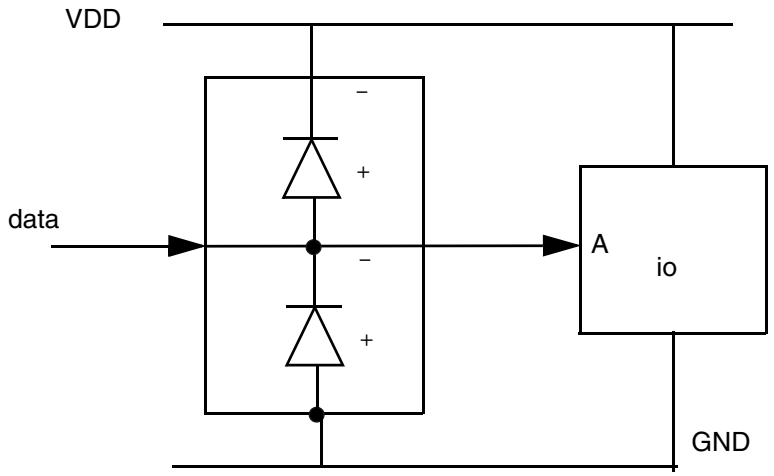
A ground clamp cell has a diode connection from the ground net to the data signal, as shown in the following diagram:



If the input signal voltage drops below -Vth, the diode conducts, thereby clamping the signal voltage to -Vth.

Power and Ground Clamp Cell

A clamp cell that has diode connections to both power and ground nets, as shown in the following diagram, clamps to both high and low voltages on the data signal. Any voltage higher than VDD+Vth is clamped to VDD+Vth, and voltage lower than -Vth is clamped to -Vth.



The diode clamp cell protects the data input pin A of instance i0 from ESD, and also from voltage overshoots and undershoots on the signal net.

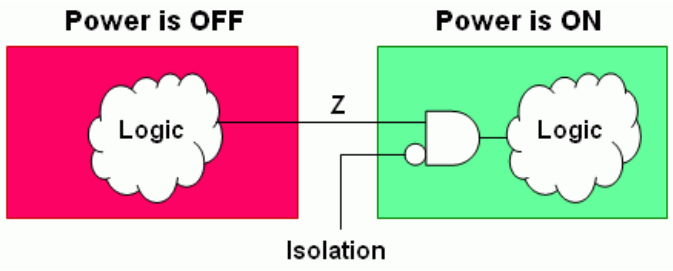
Dynamic Power

Dynamic power is the power dissipated by an instantaneous short-circuit connection between the voltage supply and the ground when the gate transitions, and the switching power dissipated when charging or discharging internal and net capacitances.

Isolation

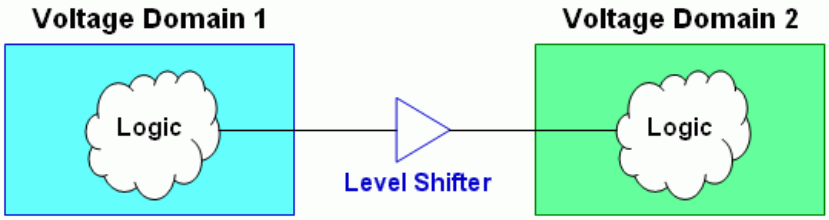
Isolation is a power management technique that prevents undriven outputs of switched OFF blocks from causing electrical problems in active blocks. Active isolation provides driven value to active logic, setting a logic cell input to a level that makes all other inputs a don’t care, i.e.

the voltage level of the other data inputs do not have any impact on the internal nets or the outputs. Serious problems occur if in-active logic is not isolated.



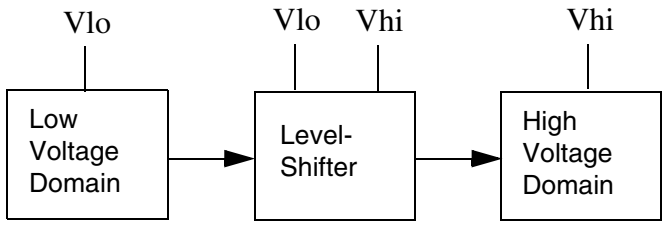
Level Shifting

Level shifting is a technique to transport a signal from one voltage value to a higher or lower voltage value. A level-shifter is a cell that is used as a boundary for signals that cross voltage domains. A low to high crossing always requires a level shifter to prevent an abnormally high power consumption. It is common to use level shifter which is a buffer powered by the higher voltage for a high to low voltage domain crossing. Serious problems occur if signals go from a low to high voltage domain without proper level shifting.

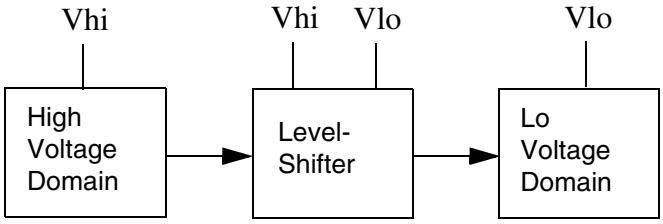


Multi-Voltage Supplies

Level shifters are required when a lower voltage signal drives a gate supplied by a higher voltage. These cells are placed between a source and receiver powered by different voltages to protect the receiver from too high or too low voltage. A level shifter requires both voltages internally to function.

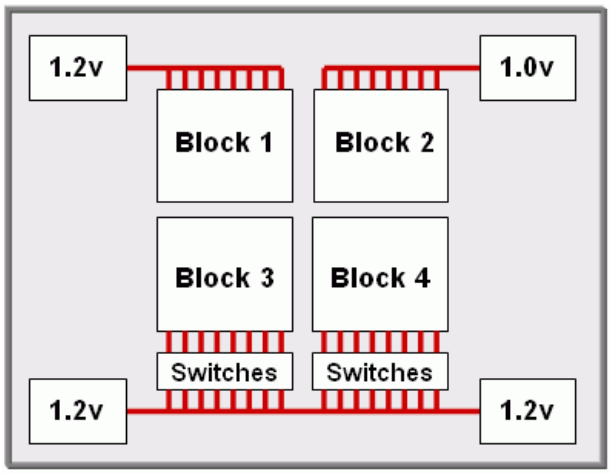


When a higher voltage signal drives a gate supplied by a lower voltage, a level shifter might not be used. When it is, it will use the lower voltage for the output stage and not use either lower or higher voltage for the input stage.



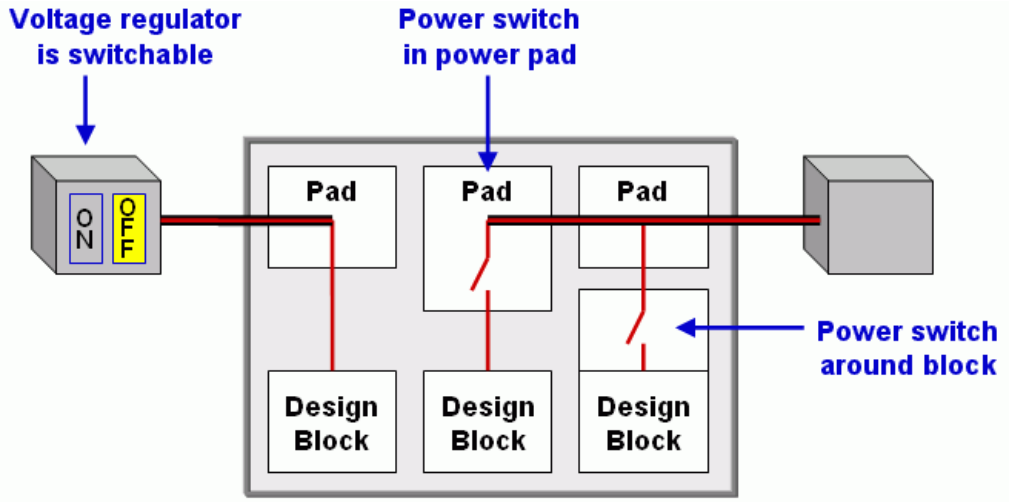
Power Domain

A power domain is all circuits whose power source has the same voltage (same supply net). All circuits which share the same power source, and if switchable, share the same power enabling and disabling control. In the following illustration, all blocks are in separate power domains:



Power Gating

Power gating is a technique to save leakage power, which switches a power domain's connection to a power source OFF, as shown in the following illustration:

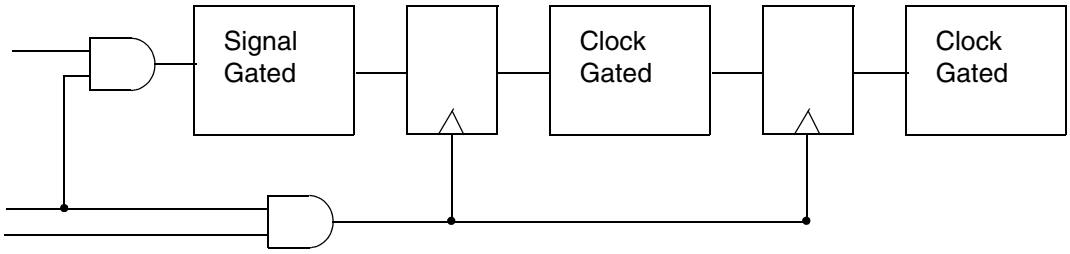


Power Switching

A Power Switch is a PMOS device between a power pin and the power to a module. When this transistor is turned off, the power domain of the supplied circuit is disabled. The power output net of a power switch is known as a switched power domain. The module power is turned OFF by turning OFF the PMOS device.

Signal Gating

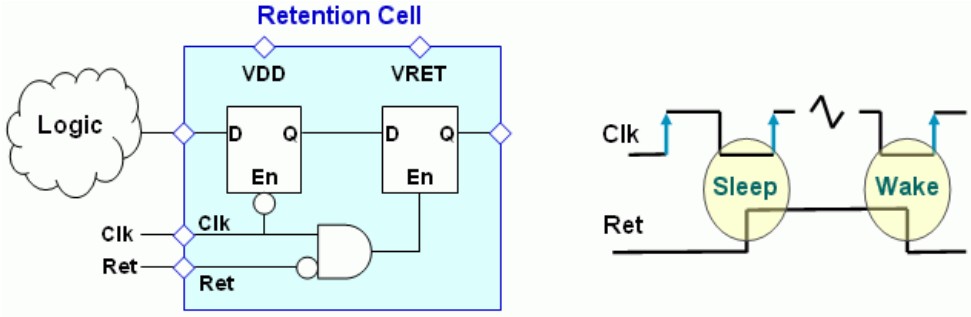
Signal is a technique to reduce dynamic power consumption by stopping the switching of signal value when they are not needed. Signal Gating is a common technique for RAM address and data pins. This makes the inputs to the logic receiving the gated signal a constant and eliminates dynamic power consumption.



State Retention

State retention is a power management technique that saves states before a block is powered down. This is the process of saving an important state of a memory element such as a latch or flip flop before the main power or ground of the module is switched off. The circuit which saves the state will require non-interruptible power and ground to ensure state preservation.

In the following illustration, Sleep = Ret is asserted during low clock then VDD is switched OFF. After the Wake - VDD is switched ON, Ret is de-asserted during low clock:



2

Power Aware Equivalency Checks

Power Aware Equivalency Checking Overview

Power aware equivalency checking is an advanced method for RTL through place-and-route functional verification. Power aware equivalency checking replaces boundary checks, which can be used only to verify two netlists that have isolation already inserted.

Power aware equivalency checking offers the following features:

- Low power design equivalency checks
- State element domain consistency checks
- Can handle cell modeling, switch modeling, and retention instances
- Power intent comparison
- State retention strategy comparison

Note: Shifters, retention cells, and switch cell insertions are not required in power aware equivalency checking.

Starting the Low Power (EC) Software

Power aware equivalency checking is done through Encounter® Conformal® with Low Power Verification.

To start the software in non-graphical mode, run the following command:

```
lec –NOGui –lp
```

To start the software in graphical mode, run the following command:

```
lec [-Gui] –lp
```

Power Aware Equivalency Checking Flow

The power aware equivalency checking flow involves the following tasks:

- Set Low Power Options for power aware equivalency checking
- Read the libraries and designs
- Read and Compare Power Intent
- Perform equivalency checking
- Perform state element domain consistency checking

Set Low Power Options

You can enable low power check options with the `SET LOWPOWER OPTION` command. This command specifies the style of the read-in netlist.

Note: When a power intent file is read in, the domain consistency check is enabled by default.

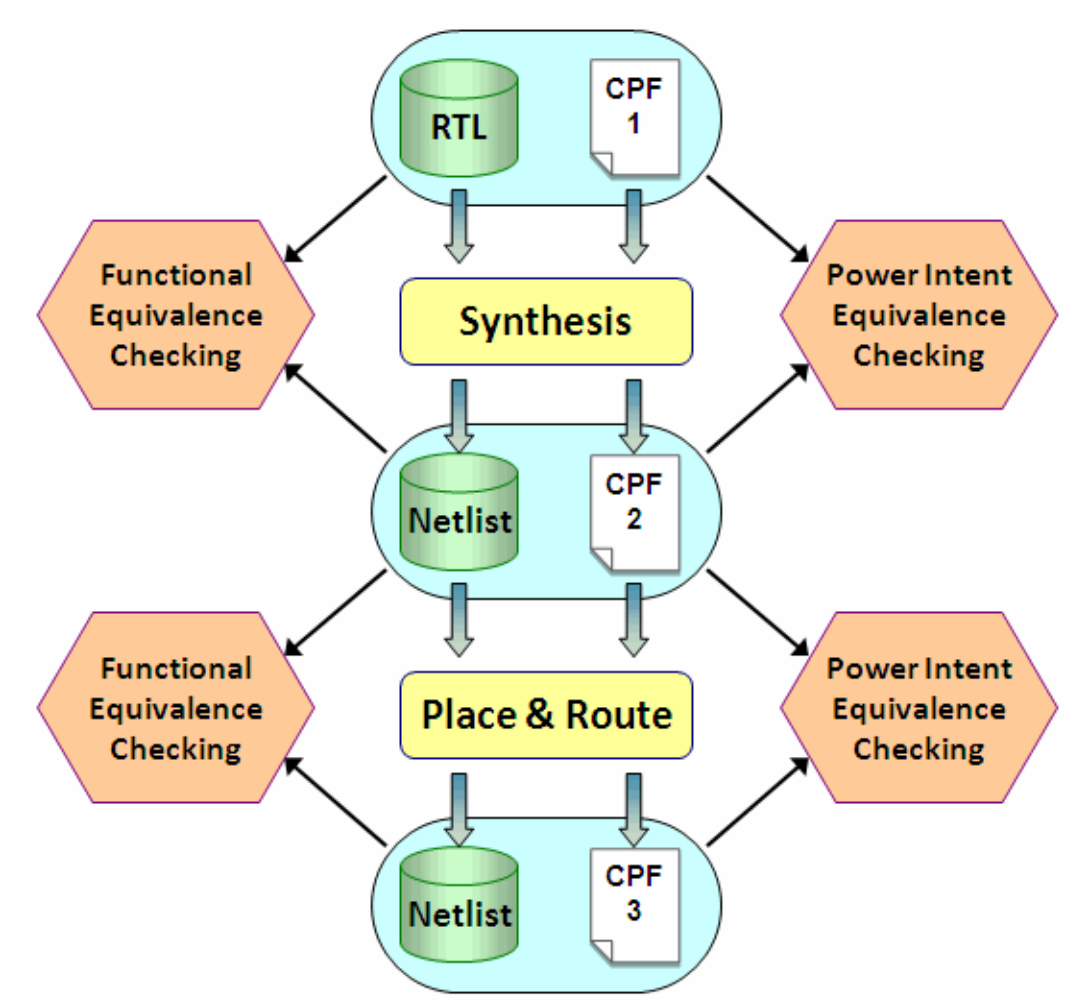
Read and Compare Power Intent

The power intent file defines the low power intent for the design.

Low Power EC compares two designs based on the power architecture of a single common power intent file (CPF). However, the power intent can change during synthesis and place-and-route, requiring Low Power EC to read in golden and revised versions of the power intent file. Low Power EC can compare the golden and revised versions of the power intent file and report the differences.

Read in the golden and revised versions of your power intent files using the `READ POWER INTENT` command and compare them using the `COMPARE POWER INTENT` command (these

commands are available only in setup mode). After comparing the power intent, report the differences using the `REPORT COMPARED INTENT` command.



Commit Power Intent

After parsing the power intent file, you can perform the isolation insertion with `COMMIT POWER INTENT` command with the `-insert_isolation` option.

Note: Power aware equivalency checking does not insert level shifters, retention cells, or switch cells.

Check Domain Consistency and Compare State Retention Strategies

After performing the equivalence checking, run the `CHECK LOWPOWER CELLS` command to perform the domain consistency checking for sequential elements. Low power equivalency checking also adds the ability to compare state retention strategies, which compares if the same retention strategy is applied to the mapped state elements.

The following illustrates the result of the `CHECK LOWPOWER CELLS` command:

```
// Command: check lowpower cells

COMPARING OBJECTS OF STATE RETENTION INSERTION SPECIFICATION ...
================================================================================
Status            DFF              DLAT
--------------------------------------------------------------------------------
Passed            0                0
Failed            2                0
Not Checked       0                0
--------------------------------------------------------------------------------
Total             2                0
================================================================================

STATE ELEMENT POWER DOMAIN EQUIVALENCE CHECKING ...
================================================================================
Status            DFF              DLAT
--------------------------------------------------------------------------------
Passed            2                0
Failed            0                0
Not Checked       0                0
--------------------------------------------------------------------------------
Total             2                0
================================================================================
```

In this example:

- Passed—Reports the number of sequential pairs that reside in the same power domain between the compared designs.
- Failed—Reports the number of sequential pairs that do not reside in the same power domain between the compared designs.
- Not checked—Specifies the number of sequential elements for which low power checks were not performed, or were interrupted.

For a verbose report, use the `REPORT LOWPOWER DATA` command.

Power Aware Equivalency Checking Sample Dofile

This section provides examples that illustrate the "Power Aware Equivalency Checking Flow" on page 25.

Example 2-1 RTL Versus Logical Netlist

Note: There is one CPF file and neither netlist has low-power cells.

The following is an example of a Conformal dofile that performs power intent equivalency checking on an RTL versus a logical netlist.

```
set lowpower option -golden_netlist_style logical -revised_netlist_style logical
read library -liberty <file* ...> -lp -both
        // Using -lp for Liberty low power cell extraction
read design <design models> -golden
read design <gate netlist> -revised
read power intent <design power spec> -cpf -both
commit power intent -golden -insert_isolation
commit power intent -revised -insert_isolation

<Mapping, modeling, and compare commands>

check lowpower cells
report lowpower data
```

Example 2-2 Logical Netlist without Low Power Cells Versus Logical Netlist with Low Power Cells

Note: In this example, the netlists have different power intent files.

The following is an example of a Conformal dofile that performs power intent equivalency checking on a logical netlist without low power cells versus a logical netlist with low power cells.

```
set lowpower option -golden_netlist_style logical -revised_netlsit_style logical

read library -liberty <file* ...> -lp -both
read library -cpf <cpf library cells> -both -append

read design <netlist1> -golden
read design <netlist2> -revised

read power intent <design power spec 1> -cpf -golden
read power intent <design power spec 2> -cpf -revised
commit power intent -golden -insert_isolation
commit power intent -revised
compare power intent // compare golden and revised power intent settings
report compared intent

< Mapping, modeling, and compare commands>

check lowpower cells
report lowpower data
```

Example 2-3 Logical Netlist with Low Power Cells Versus Physical Netlist

```
// Reading LEF is not required when Liberty files has pg_pin defined
// read lef file <file*...> -rev

read library -liberty <file* ...> -lp -both

read design <gate netlist> -golden
read design <P&R netlist> -revised

read power intent <design power spec 1> -cpf -golden
read power intent <design power spec 2> -cpf -revised
commit power intent -golden
commit power intent -revised
compare power intent // compare golden and revised power intent settings
report compared intent

< Mapping, modeling and compare options and commands>
check lowpower cells
report lowpower data
```

3

Low Power Extended Checks

Starting Conformal Low Power (Extended Checks)

To start the software in non-graphical mode, run the following command:

```
lec -verify -NOGui -lp
```

To start the software in graphical mode, run the following command:

```
lec -verify [-Gui] -lp
```

Logical Netlist Checking

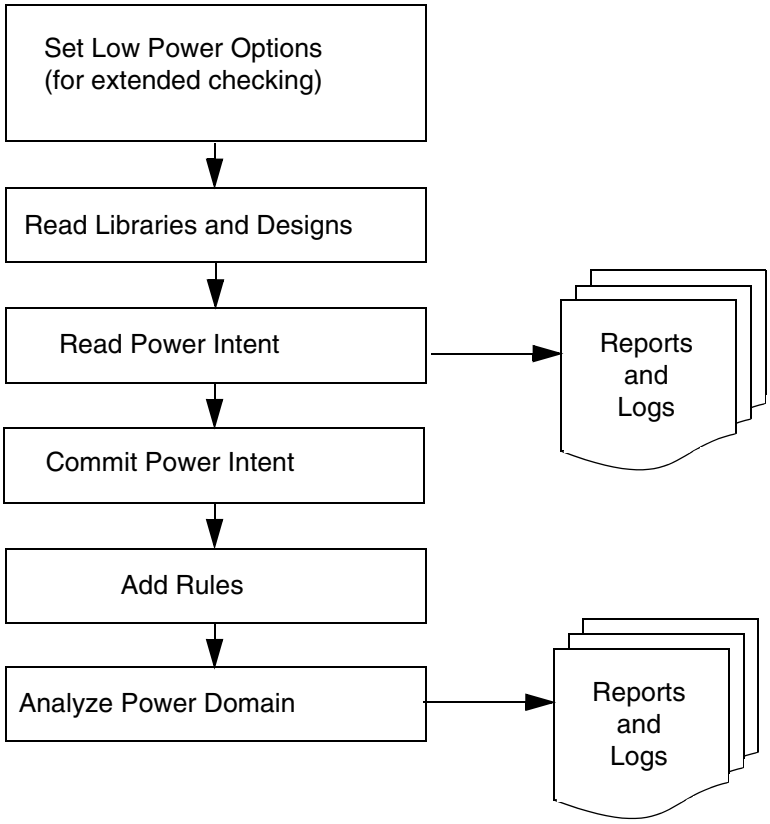
This section describes how Conformal Low Power checks can be performed on logical netlists.

In netlists without power and ground nets, low power checks verify that the syntax and quality of the power specification are correct before moving on to simulation and synthesis.

For gate netlists that have low-power cell insertion, the low power rule checks verify that the required low-power cells (level shifter, isolation, and retention) are inserted correctly. This does not include power-intent syntax checking.

Logical Netlist Checking Flow

The following figure illustrates the logical netlist checking flow:



Design Step 1: Developing Power Specifications

The following illustrates a behavioral RTL model and its power intent specification:

Behavioral RTL Model

```
always @() begin
 if (Rst) St <= 2'b00;
 else
  case ({St,En})
  3'b001 : St <= 2'b01; // Initiate
  3'b010 : St <= 2'b00; // Interrupt
  3'b011 : St <= 2'b10; // Resume
  3'b100 : St <= 2'b01; // Interrupt
  3'b101 : St <= 2'b11; // Active
  3'b110 : St <= 2'b10; // Initiate
  endcase
  end
```

+

Power Specification (CPF)

```
set_design TopChip
  create_power_domain -name top
  create_power_domain -name pcm
  create_nominal_condition -name off
  create_nominal_condition -name low
  create_nominal_condition -name high
  create_power_mode -name sw
  create_power_mode -name low
  create_isolation_rule -name Iso1 -from pcm -to top
```

You can use the Conformal Low Power graphical user interface (GUI) to help develop your power specifications. The Conformal Low Power GUI offers the following features:

- Power specification forms that are design data aware
- Interactive checking and diagnostics
- Does not require advanced CPF language skills

For more information on the Conformal Low Power GUI, refer to Chapter 5, “Low Power User Interface Overview”.

Verification Step 1: RTL and Power Specification Checking

During RTL verification, low power checks verify that the syntax and quality of the power specification are correct before moving on to simulation and synthesis. Specifically, the power

specification is checked for correct power specification syntax and missing isolation or level-shifter rules.

Command example:

```
set lowpower option -netlist_style logical
read library -liberty -lp <file* ...>
analyze library -lowpower

read design <design models>
read power intent <power spec> -cpf
```

Note: The `COMMIT POWER INTENT` and `ANALYZE POWER DOMAIN` commands are not required for RTL and power specification checking.

Design Step 2: Gate Synthesis

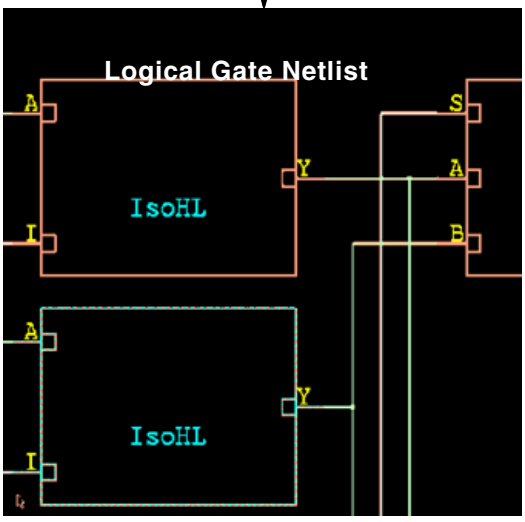
In this design step, the behavioral RTL model is synthesized and the low power cells are inserted according to the power specification.

Behavioral RTL Model

```
always @() begin
 if (Rst) St <= 2'b00;
 else
  case ({St,En})
  3'b001 : St <= 2'b01; // Initiate
  3'b010 : St <= 2'b00; // Interrupt
  3'b011 : St <= 2'b10; // Resume
  3'b100 : St <= 2'b01; // Interrupt
  3'b101 : St <= 2'b11; // Active
  3'b110 : St <= 2'b10; // Initiate
  endcase
  end
```

Power Specification (CPF)

```
set_design TopChip
  create_power_domain -name top
  create_power_domain -name pcm
  create_nominal_condition -name off
  create_nominal_condition -name low
  create_nominal_condition -name high
  create_power_mode -name sw
  create_power_mode -name low
  create_isolation_rule -name Iso1 -from pcm -to top
```

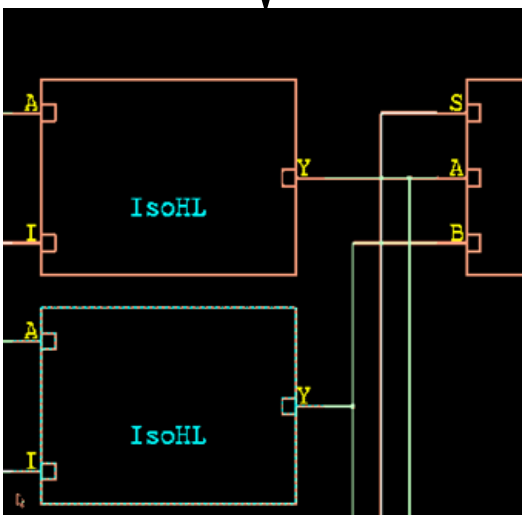


Verification Step 2: Power Specification and Gate Netlist Checking

During gate-level netlist verification, low power checks verify that the required low-power cells (level shifter, isolation, and retention) are inserted as specified by the power specification. During this step, low power checks look for electrical problems and issues with retention control and isolation control connectivity.

Power Specification (CPF)

```
set_design TopChip
  create_power_domain -name top
  create_power_domain -name pcm
  create_nominal_condition -name off
  create_nominal_condition -name low
  create_nominal_condition -name high
  create_power_mode -name sw
  create_power_mode -name low
  create_isolation_rule -name Iso1 -from pcm -to top
```



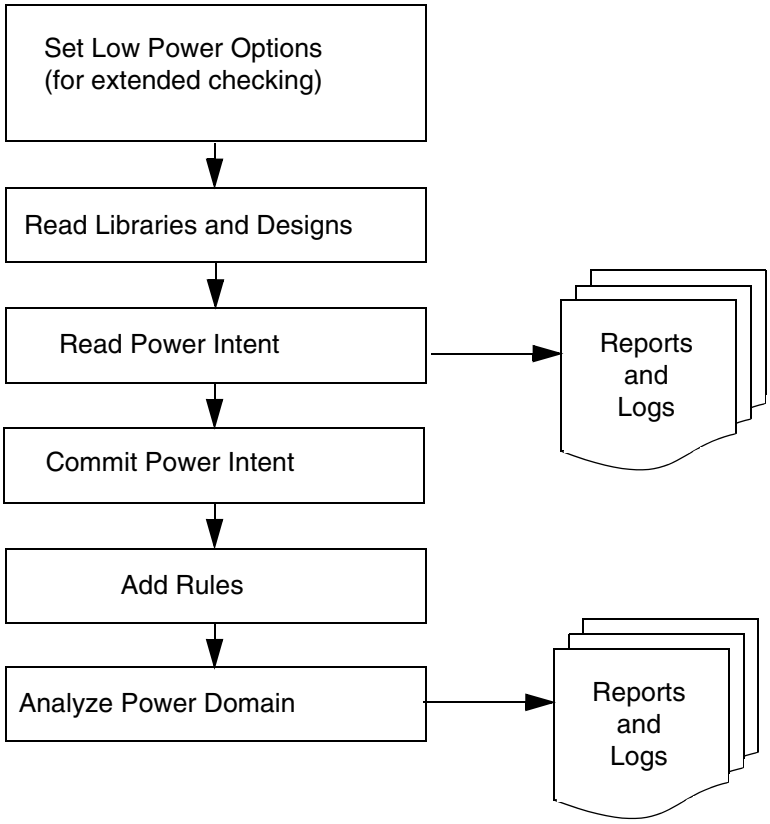
Command Template for Logical Netlist Checking

```
set lowpower option -netlist_style logical
read library -liberty -lp <file* ...>
analyze library -lowpower

read design <design models>
read power intent <power spec> -cpf

commit power intent

analyze power domain
```

Physical Netlist Checking

Physical Netlist Low Power Checking Flow

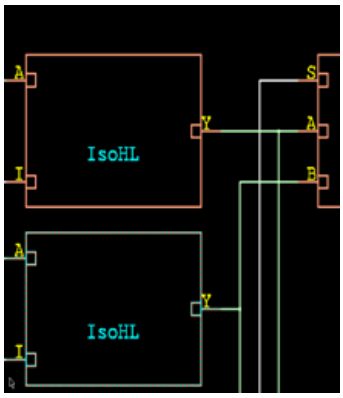
The Conformal Low Power physical netlist checking works on netlists with power and ground net. It completes checks for low power cells.



Design Step 3: Place and Route

The following illustrates a gate netlist and its power specification.

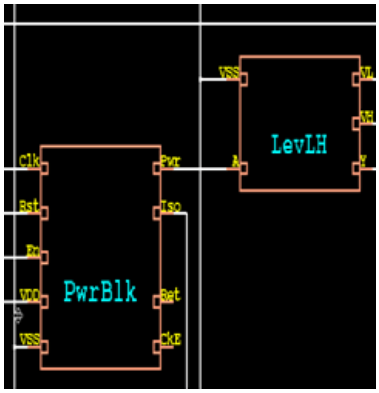
Gate Netlist



Power Specification

```
set_design TopChip
  create_power_domain -name top
  create_power_domain -name pcm
  create_nominal_condition -name off
  create_nominal_condition -name low
  create_nominal_condition -name high
  create_power_mode -name sw
  create_power_mode -name low1
  create_isolation_rule -name Off2On
  create_level_shifter_rule -name L2H
```

Place and Route Netlist



In this design step, the following steps occur:

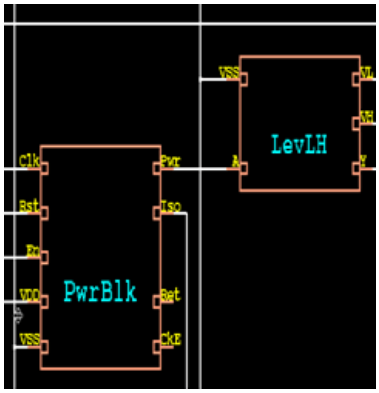
- Perform place and route on the gate netlist.
- Insert power switch cells
- Route power and ground nets
- Integrate the core and padframe

Verification Step 3: Conformal Low Power Extended Checks

Power Specification

```
set_design TopChip
  create_power_domain -name top
  create_power_domain -name pcm
  create_nominal_condition -name off
  create_nominal_condition -name low
  create_nominal_condition -name high
  create_power_mode -name sw
  create_power_mode -name low1
  create_isolation_rule -name Off2On
  create_level_shifter_rule -name L2H
```

Place and Route Netlist



In this verification step, the following Conformal Low Power extended checks are performed:

- Checks to ensure that insertion is in line with the power specification
- Structural checks to find electrical problems
- Low power cell structural checks
- Low power cell control signal checks

Command Template for Physical Netlist Checking

The following example illustrates reading power and ground pins from LEF, and reading low power cells from CPF declarations:

```
set lowpower option –netlist_style physical

read lef file <lef file* …>
read library <functional library file* ...>

read design <gate netlist>
read power intent <cell and design power spec*> -cpf

commit power intent
analyze power domain
```

The following example illustrates reading in a liberty file with low power attributes:

```
set lowpower option –netlist_style physical

read library -liberty -lp <file* ...>
analyze library –lowpower
```

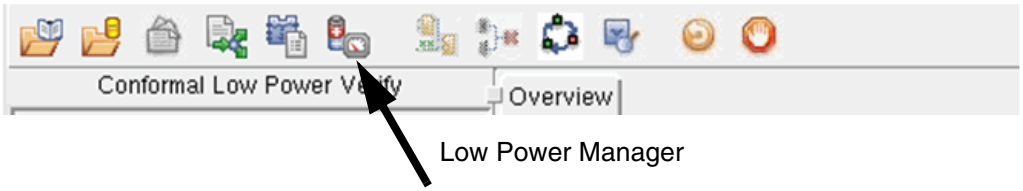
```
read design <gate netlist>
read power intent <power spec> -cpf

commit power intent
analyze power domain
```

4

Low Power Diagnosis

Interactive Diagnostics

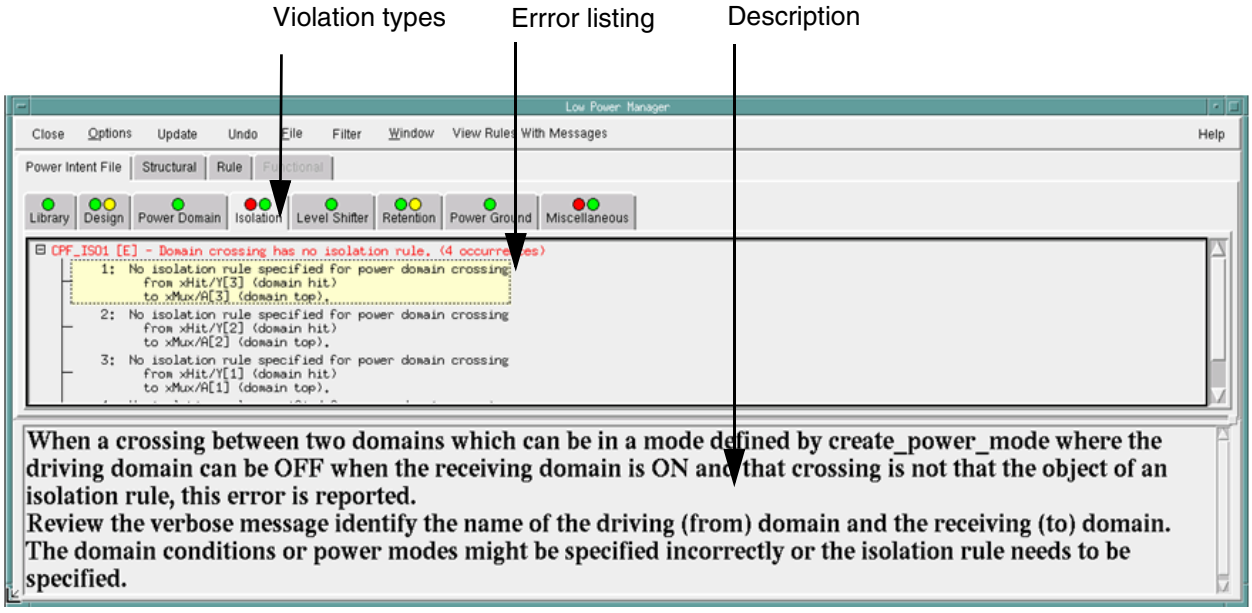
To access the Low Power feature, click on the *Low Power Manager* icon located on the icon bar in the main window, or select *Low Power Manager* from the *Tools* drop-down menu.



The Low Power Manager consists of tabs for each of the rule categories:

- Power intent file—Displays power intent quality checks. The power intent tab is only active if power intent files have been read in.
- Structural—Displays the structural rule checks.
- Rule—Displays the structural rule checks
- Functional—Displays the functional rule checks.

By default, only rules with violations are displayed. For the description of a rule message, see type ‘`man <rule> -verbose`’ in the command line.



Running Low Power Functional Checks

For each cell or crossing, you can click the row to highlight the name, then click the right mouse button to bring up the pull-down menu where you can select from the following options:

- *Add Power Check*—applies the power check to the selected low power object.
- *Delete Power Check*—removes the power check from the selected low power object.
- *Report Power Check*—reports the power check information on the selected low power object.
- *Diagnose*—diagnoses failures reported after doing functional checks on the selected low power object.
- *Sequential Explorer*—opens the Sequential Exploration Manager, which you can use to sequentially explore designs.
- *Module Schematics*—opens the Module Schematic Viewer, displaying the top-level design with the object selected.
- *Flattened Schematics*—opens the Flattened Schematic Viewer, displaying an isolated view of the selected object.
- *Report Validated Data*—displays information about validation results of added power checks.

Low Power Manager Fields and Options

Options	Click the *View* pull-down menu and choose *Rule with messages only* (the default), *All* to display a complete list of rules and the messages (violations) for each page, or *Hidden Rules* to display the hidden rules and the messages. Click the *Page Size* option to open the Page Size form to specify the page limits to control the number of rules that are displayed. The default is 25.
Update	Forces a refresh of the message display.
Undo	Unmarks the previous waive occurrence. This will only undo the last waiver mark. To undo previous waiver marks, select *Options – Hidden Rules* and select *Unwaive Occurrence*.

File	Opens the Write Rule Check form where you write the low power violations into a file. For more information, see Writing Low Power Violations to a File on page 48.
Filter	Opens the Filter Rule form where you can add or delete rule filters. For more information, see Filtering Rules on page 49.

Status Icons

 or 	Indicates that the rules in this category have passed.
 or 	Indicates that the design triggered one or more rule failures in the applicable category.
	Indicates that the checks have been run but the status is unresolved.
	Indicates that the checks have not yet been validated.

Changing Severity in the Low Power Manager

For each message, you can click to highlight the message, then right click to bring up the pull-down menu where you can select from the following options:

- *Severity*—(in Setup mode) changes the message severity level. You can choose *Warning*, *Error*, *Ignore*, or *Note*.
- *Report*—displays the status of the rules. You can select one of the following:
  - *Summary*—displays the rule description and occurrences of the violation.
  - *Verbose*—displays detailed descriptions of each violation.

Diagnosis in the Low Power Manager

After expanding the message by clicking the *+* symbol, you can click to highlight the cell row, then right click to bring up the pull-down menu where you can select from the following options:

- *Waive Occurrence*—marks the specified occurrences as waived.

  For more information, see Marking Rule Occurrences as Waived on page 47.

- *Source Code*—opens the Source Code Viewer on the selected object.
- *Schematics* (for rules with violations)—opens the Module Schematic Viewer. You can select one of the following:
  - *Show Module Schematics*—displays the entire module where the selected object resides.
  - *Show Instance View*—displays a partial schematic of the module that is relevant to the selected object.
- *Analyze Instance Connectivity*—shows the connectivity of the selected instance along with its low power information to help you debug low power violations. If the instance is a special low power cell, this also shows the low power cell specific information to help debug low power structural errors.

Marking Rule Occurrences as Waived

To mark the specified occurrences as waived, you can use the `ADD RULE WAIVER` command or the Rule Manager:

1. From a Rule Manager, expand a rule to view the list of occurrences.
2. Right-click on an occurrence and choose *Waive Occurrence* from the pop-up menu.

Note: This mark is lost when the rule is re-checked.

After marking an occurrence as waived, you can use the `DELETE RULE WAIVER` command to unmark the waived occurrence. From the Rule Manager, or you can by select the *Undo* button. This will only undo the last waiver mark. To undo previous waiver marks, select *Options – Hidden Rules* and select *Unwaive Occurrence*.

Tip

You can report waived rule occurrences with the `REPORT RULE CHECK -waived` command.

Writing Low Power Violations to a File

To write the low power violations into a file, use the following procedure:

1. Click *File – Write Rule Check*.

   This opens the Write Rule Check form.

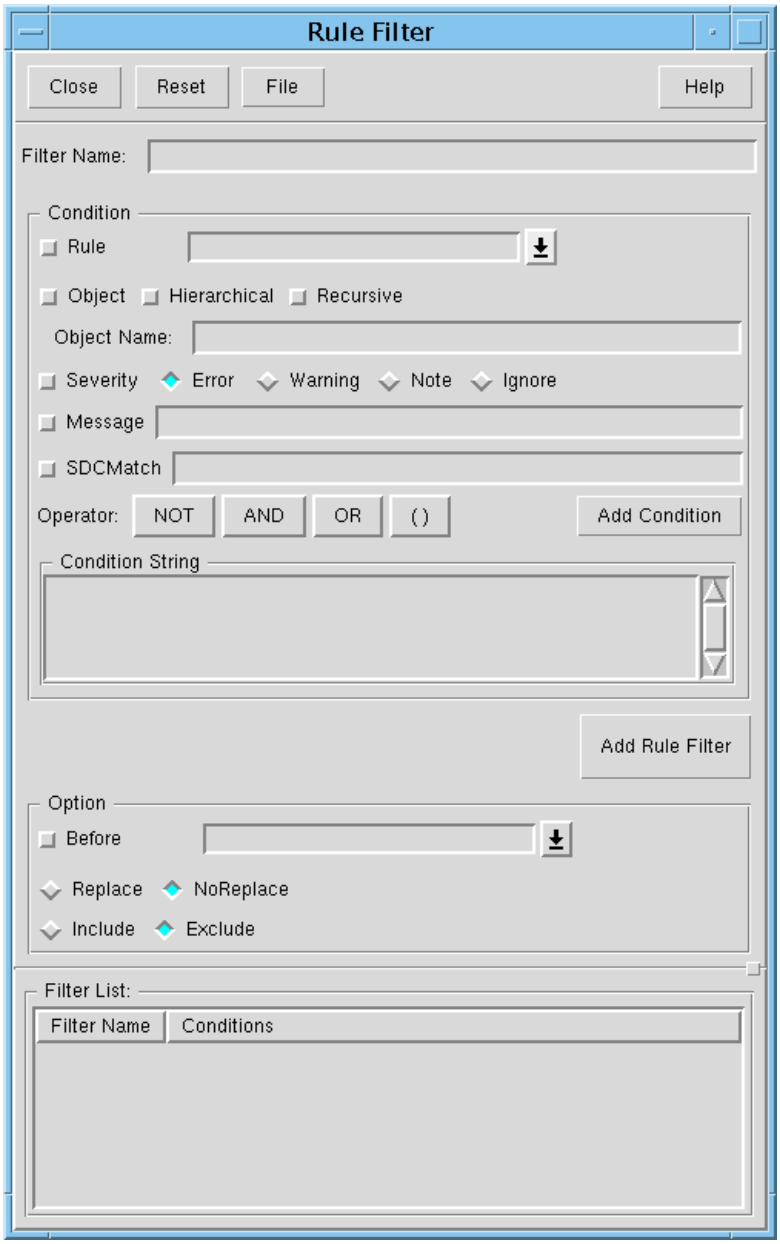
Write Rule Check Form Fields and Options

Filename	Type the file of the rule file, or click *Browse* to choose a file from the Write Rule Check browser window.
Occurrence Type	*Complete* (the default) reports all occurrences regardless whether they are hidden or not. *Hidden* reports only occurrences that are hidden. *Waived* reports only occurrences that are marked as waived.
Open Mode	*Replace* (the default) overwrites the contents of an existing file. *Append* appends the contents to an existing file.

Filtering Rules

You can add or delete rule filters with the Rule Filter form.

➤ From the HDL or Modeling Rule Manager, click the *Filter* button.



Adding Rule Filters

To add a rule filter:

1. Specify the name of the filter in the *Filter Name* field.

   Note: If you do not enter a name in this field, a unique name is automatically generated.

2. Enter the Conditions.

   - *Rule* — Filters out all occurrences of specified rule(s). Type the name of the rule or use the pull-down menu.
   - *Object* — Matches rule occurrences related to objects that match the specified pattern.
     - *Hierarchical*—matches any instance in the design hierarchy against the specified pattern (analogous to the SDC command '`get_pins -hier <pinname>`').
     - *Recursive*—matches any object under an instance that matches the specified pattern (analogous to the Unix command '`grep -r ... <dirname>`').
   - *Severity* — Filters out all occurrences of the selected severity level(s).
   - *Message* — Matches rule occurrences whose verbose message matches the specified pattern.
   - *SDC Match* — Matches rule occurrences related to SDC statements matching the specified pattern. The string representing the SDC statement is the one displayed in the SDC command browser, not the one from the SDC file.

     Note: This condition is only available for SDC rules.
   - *Operator* — Specifies that the operator.

3. Enter the Options.

   - *Before* — Specifies that the new or replaced filter is inserted before the first filter whose name matches the provided pattern. This can be the same name as the one being replaced. In this case, the replaced filter stays in the same position in the list of filters. Type the name of the filter or use the pull-down menu.

- *Replace* — Specifies that the filter name can be that of an existing filter, which is then modified.
- *NoReplace* — Specifies that the filter name cannot be that of an existing filter.
- *Include* — Specifies that the filter will cause any matching occurrence not to be filtered out, unless this is reversed by another filter down the list.
- Exclude — Specifies that the filter will cause any matching occurrence to be filtered out, unless this is reversed by another filter down the list.

4. Click the *Add Rule Filter* button.

Deleting Rule Filters

To delete a rule filter:

1. Select a filter in the *Filter List*.
2. Right-click and choose *Delete* or *Delete All* from the pop-up menu.

Categorizing Messages

You can categorize rule messages using a redirect with the `REPORT RULE CHECK` command:

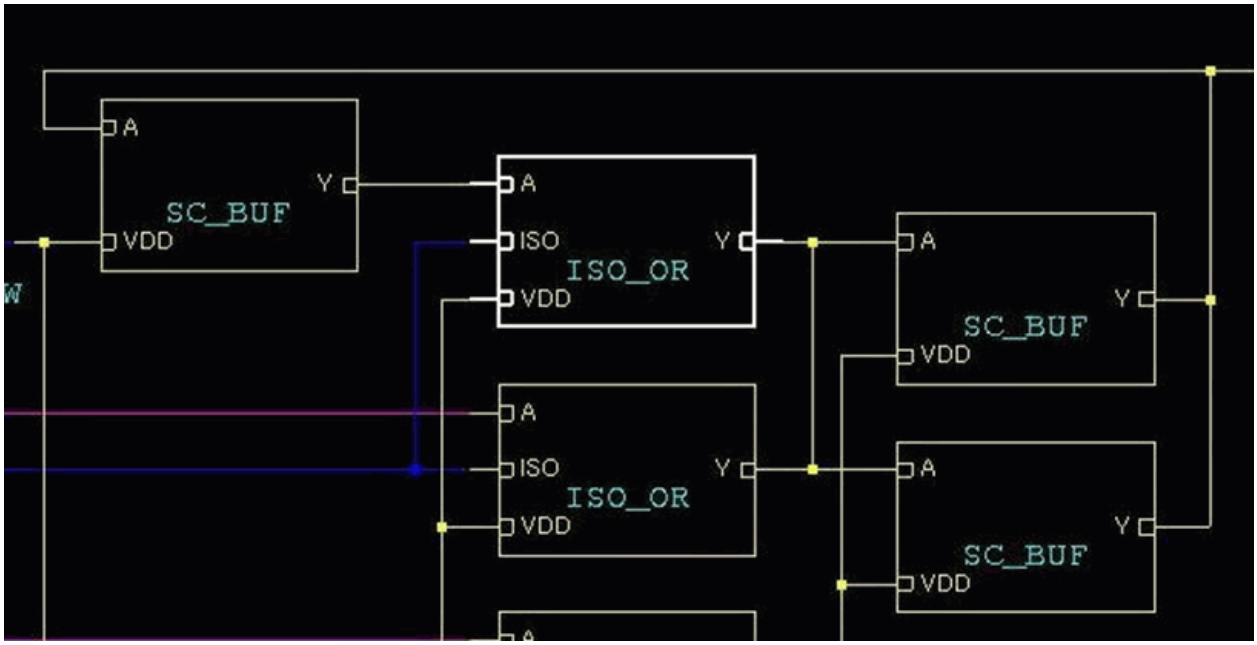
```
% report rule check CPF_* -verbose > CPF.report
%   report rule check LSH*   -verbose > LSH.report
%   report rule check ISO*   -verbose > ISO.report
%   report rule check PDM* -verbose > PDM.report
```

Debugging Combinational Loops

Combinational loops (`STRC1.1`, `STRC1.2`, `CLP_STRC1.1`, `CLP_STRC1.2`) can be difficult to debug. The following is a two-step approach to understand the complete debugging process of combinational loops.

1. In the Conformal main window, click the *Low Power Manager* icon to open the Low Power Manager.
2. In the Low Power Manager, click the *Logical Check* tab.
3. Click the *Structural* tab.

4. Select any of the violations (for example, `CLP_STRC1.1`), and expand the list to select one of the violating instances.
5. Right click on the instance name and choose *Schematic – Show Module Schematic* to open the schematic as shown in the following figure.



The command to report this in the text window is:

```
report rule check CLP_STRC1.1 -verbose.
```

The result of the command is:

```
CLP_STRC1.1: Low power cell in strong combinational loop
    Severity: Error      Occurrence: 1
    1: Pin 'A' of 'm2/xicomb' (ISO_OR) is in a strong combinational loop
```

From the schematic, it is apparent that the `ISO_OR` cell is involved in a combinational loop, and because this is a small path, you can also traverse the path manually to see the combinational loop. But in a real design, tracing this path is not as easy.

In order to do this, use the `REPORT PATH -loop` command as follows:

```
report path -loop m2/xicomb/x2 -strong
```

where `x2` is the instance beneath the `ISO_OR` module.

When you run this command, the software gives a complete path of the combinational loop. In this example, there are two paths to the loop, and both are listed here:

```
Strong combinational loop from m2/xicomb/x2 (18):
```

```
OR      : m2/xicomb/x2 (18) ? ORIGIN
BUS     : m2/xicomb/a1$BUS (10)
BUF     : $ID32 (32)
BUF     : m2/xb1/x1 (19)    ? PATH 1
BUS     : m2/xcombLoop/X$BUS (11) ? PATH 2
BUF     : $ID31 (31)
BUF     : m2/xcombLoop2/x1 (17)
OR      : m2/xicomb/x2 (18)  ? ORIGIN
```

5

Low Power User Interface Overview

When you launch Conformal Low Power (with a CLP-XL or CLP-GXL license), the new Low Power graphical user interface (GUI) invokes by default.

The tool launches in GUI mode when you use:

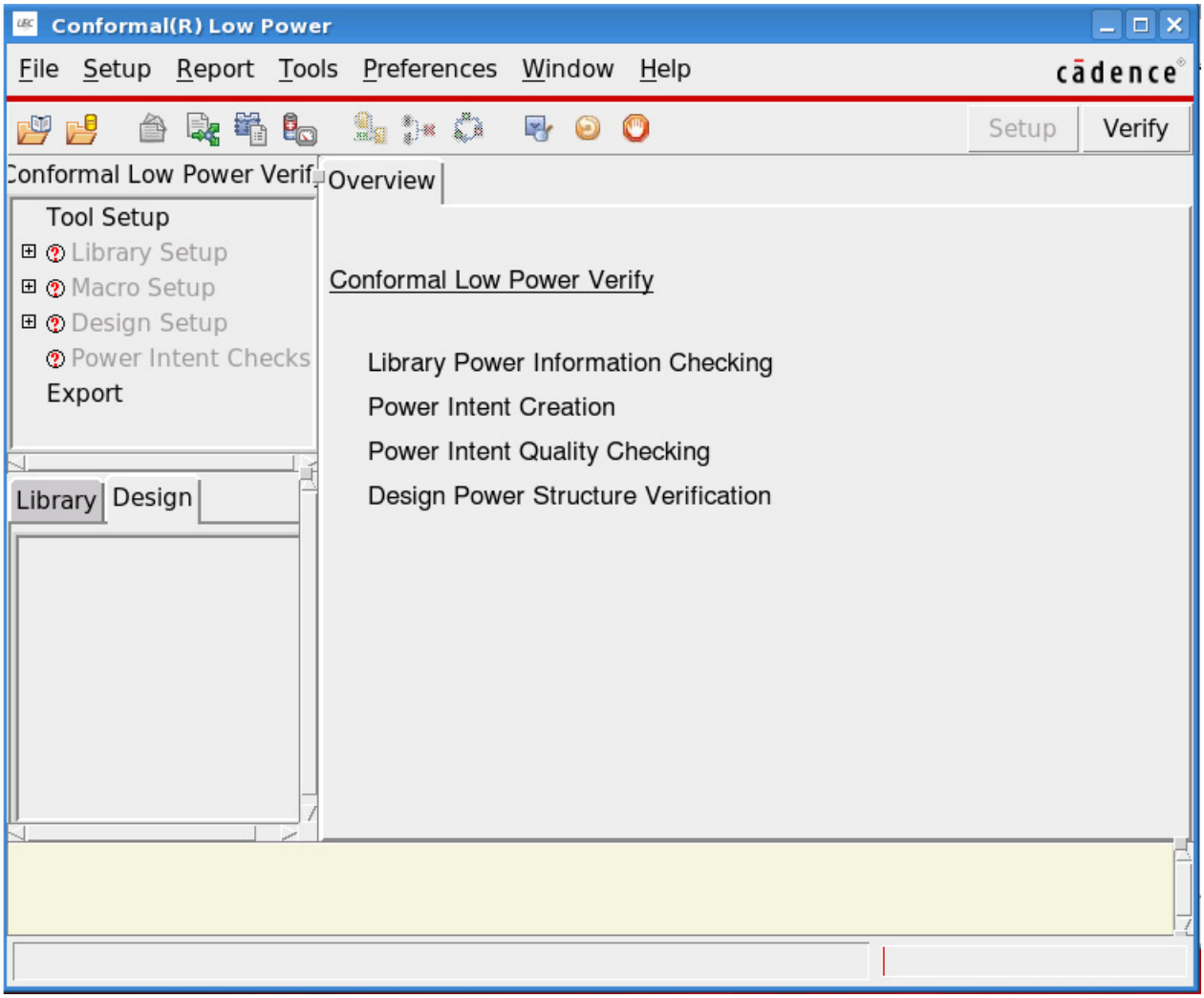
```
lec -lp -verify
```

Low Power GUI Features

This GUI offers the following features:

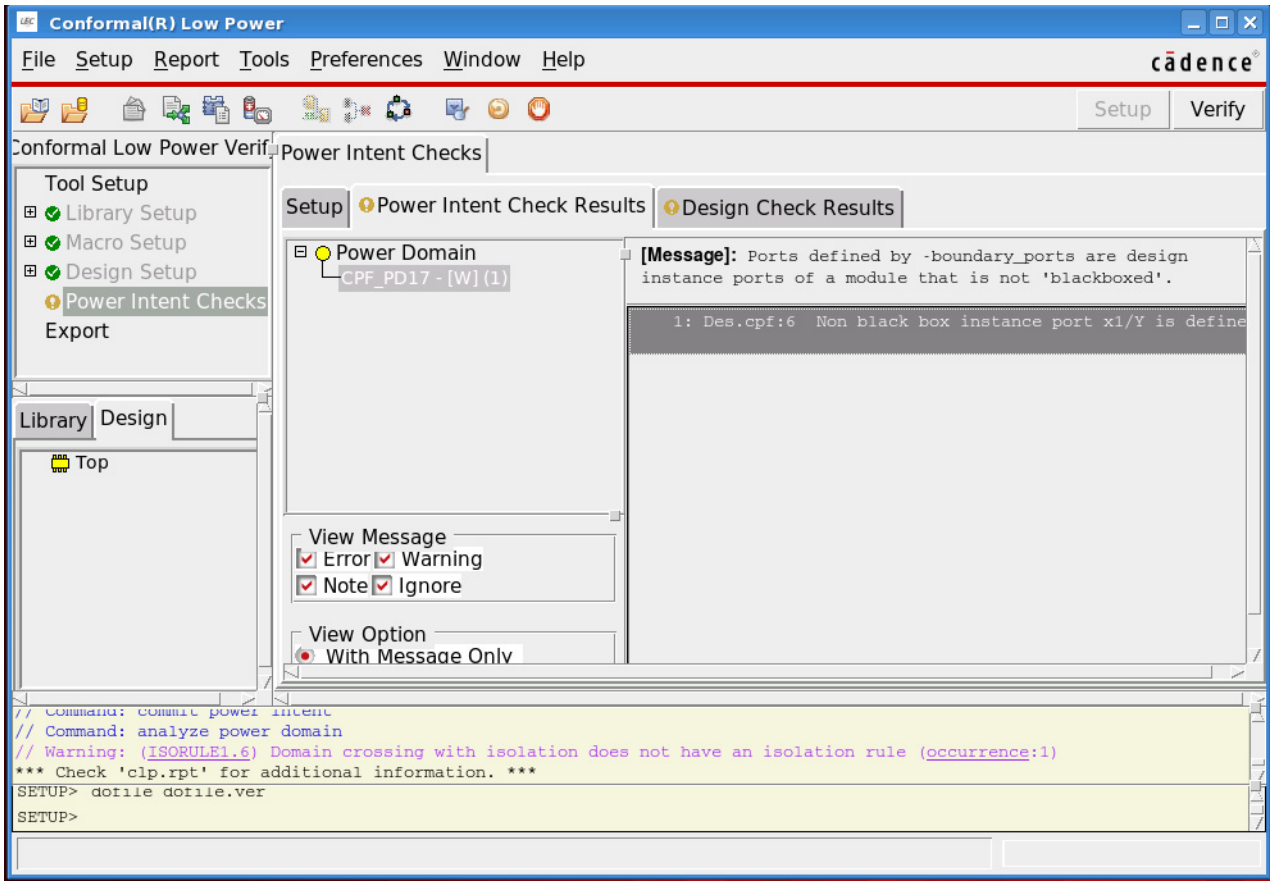
- Library Power Information Checking—Checks the consistency between library views for information such as ports, cell names, and power/ground port attributes.
- Power Intent Creation—You can intuitively define power intent specifications in CPF by filling out table templates without advanced knowledge of their language syntax or semantics.
- Power Intent Quality Checking—Guides you through the checks required to verify that the power intent specifications (in CPF format) are correct and complete, before handing them off to synthesis or place and route.
- Design Power Structure Verification—Guides you through the checks required to verify that designs after synthesis or place and route are correctly implemented based on the the low power specifications.

Figure 5-1 Low Power GUI

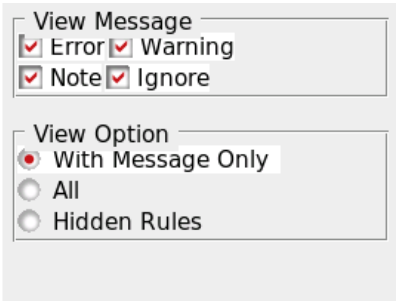


Viewing Rule Checks

During a debug session, click on the Power Intent Checks item under the Tool Setup window on the and the right-hand pane changes the display accordingly for rule check occurrences, categorized by the rule check type.

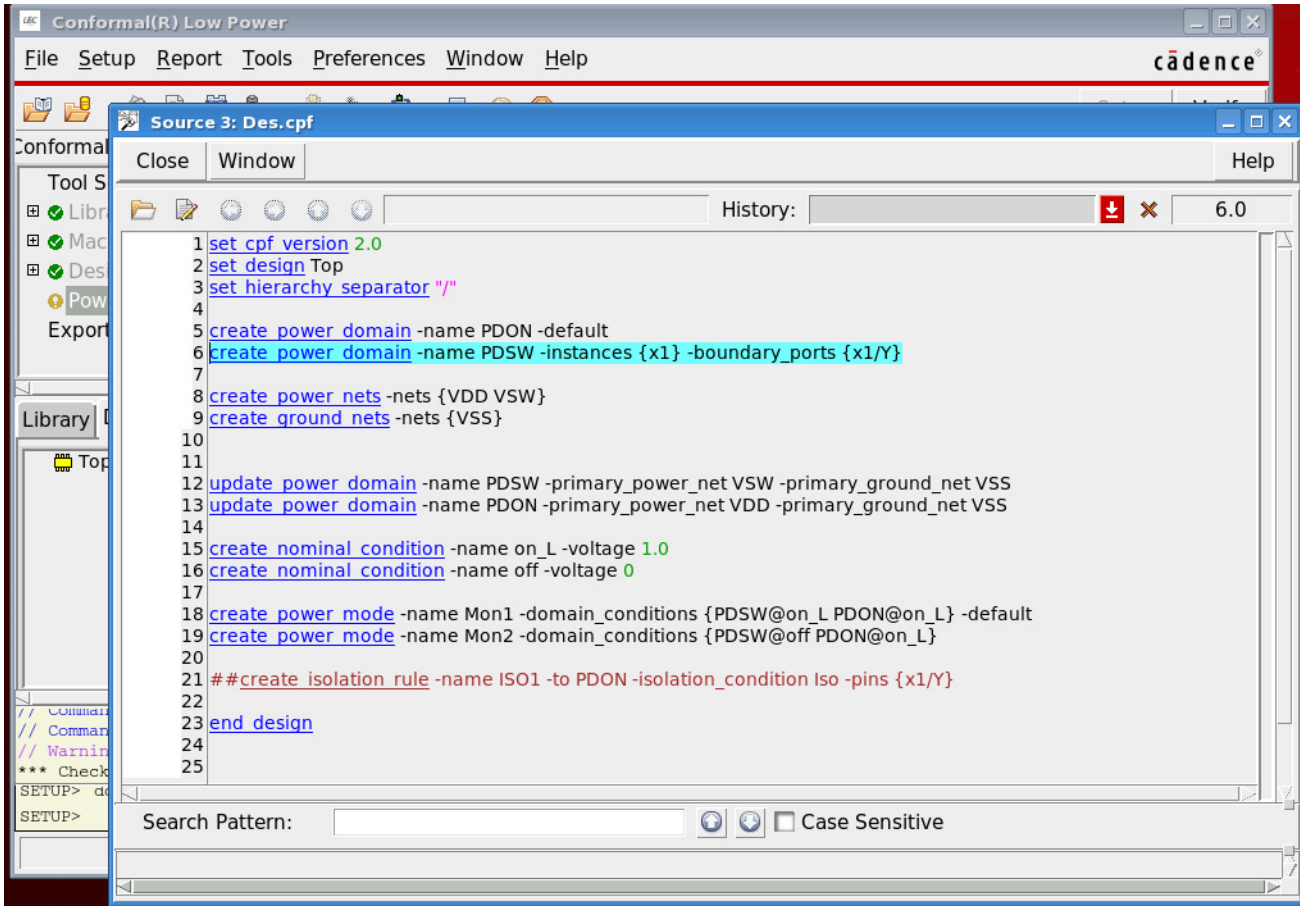


You can filter what is displayed by message type (Error, Warning, Note, or Ignore). You can also filter the amount of checks displayed (view only violated rule occurrences with messages, view all rule checks, or hide all rules).

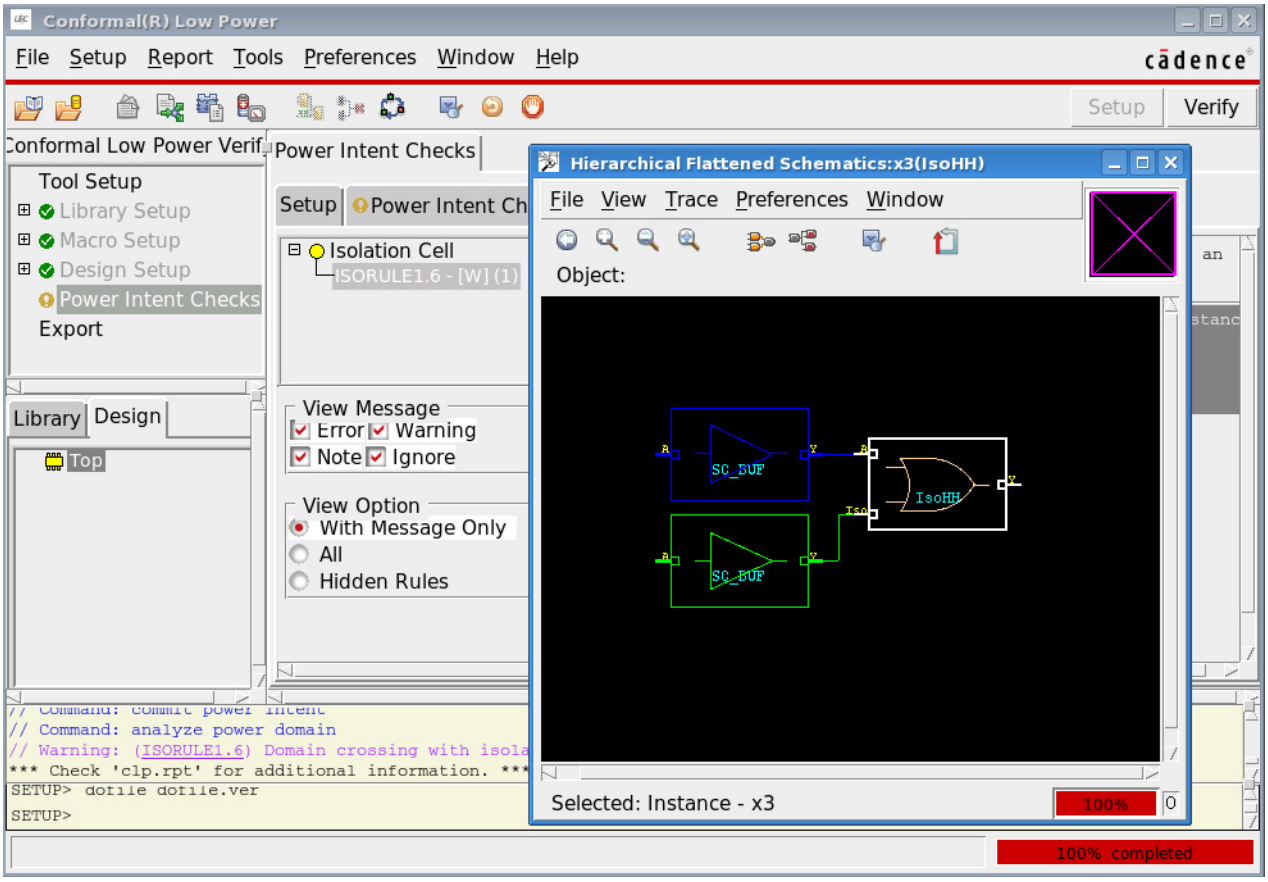


When you click on an occurrence of a rule check type, the rule message will be shown as well as additional information which is helpful for debugging.

For power intent quality checks, a window will pop up which includes the source code related to the message.



For structural rule checks, a window containing the related schematic view showing where the error occurs will be displaying.



Notes:

- Debugging can also be achieved through the Low Power Manager (go to *Tools - Low Power Manager*) from the applicable tab. This is especially useful for larger designs.
- The check form can be used to diagnose low power errors, but you cannot invoke the ANALYZE LIBRARY command or the ANALYZE POWER DOMAIN command from this form. Those commands must be invoked by command line.
- By default, the *Preferences - Launch Schematics/Source Code Viewer Standalone* menu item is checked to launch the schematics browser in a stand-alone window. Users can disable this feature to make the schematics embedded in the main GUI window.

For more information:

- Structural rule checks are described in the "Low Power Design Rule Checks" chapter of the *Low Power Reference*.

- Power intent quality checks are described in the "Common Power Format Rule Checks" chapter of the *Low Power Reference*.

6

CPF Integrator

Note: This requires a Conformal Low Power XL license and is only available with the `-verify` executable option (`lec -lp -verify`)

Hierarchical CPF is used for bottom-up implementation and IP integration. The integrated CPF is the starting point for all design-level verification and implementation, RTL simulation, logic synthesis, and place and route.

The Conformal Low Power CPF Integrator reads in hierarchical CPF files and creates flat CPF files for verification of integrated scope or chip-level implementation. This supports lower scope power domain reconfiguration, low power rule creation, and automatically resolves block-level and top-level low power rule precedence.

Using the CPF Integrator

Use the `WRITE POWER INTENT` command's `-integrated` option to specify that when a hierarchical CPF is read, it generates integrated flattened CPF files. During the integration, each design scope is checked independently for design correctness. Although this step is expected to have already been performed when the CPF for that design scope was developed, it is verified at this phase to avoid integration issues due to incorrect or incomplete lower scope CPF. Checks include domain mapping checks and port consistency checks

By default, the library related information is written to a `<filename>_lib_<version>` file. The integrated CPF `<filename>` will source these related library CPF file(s). To specify that the library information is not required to be integrated, use the following command:

```
WRITE POWER INTENT <filename> -integrated -nolibrary
```

The CPF Integrator writes the integrated CPF file for each macro model declaration once and each macro instance using the following format to link to the macro model:

```
set_instance <instance name> -model <macro model name>
```

Alternatively, you can specify that the definitions of a macro model are expanded immediately after each `set_instance` of that macro model with the following command:

```
WRITE POWER INTENT <filename> -integrated -expand_macro
```

When the integrated CPF is written out and read in to the Conformal software, a quality check is performed on the integrated CPF. Any top-scope CPF errors, or any errors related to integration, are reported.

Important

To continue in Conformal Low Power, you must read the integrated CPF file back in using the `READ POWER INTENT <file> -replace` command followed by the `COMMIT POWER INTENT` command.

Creating Flat CPF Files

To create a flat CPF file that is integrated from hierarchical CPF files, do the following:

1. Read in the hierarchical CPF files with the `READ POWER INTENT` command. For example:

   ```
   read power intent hier1.cpf hier2.cpf hier3.cpf
   ```

2. Run the `WRITE POWER INTENT` command. For example:

   ```
   write power intent output.cpf -integrated
   ```

Verifying Hierarchical CPF Files

To verify the hierarchical CPF files, do the following:

1. Read in the hierarchical CPF files with the `READ POWER INTENT` command. For example:

   ```
   read power intent hier1.cpf hier2.cpf hier3.cpf
   ```

2. Create a flat output CPF file with the `WRITE POWER INTENT` command. For example:

   ```
   write power intent output.cpf -integrated
   ```

   To continue in Conformal Low Power, you can read the integrated CPF file back in using the `READ POWER INTENT` command followed by the `COMMIT POWER INTENT` command.

3. Read in the flat output CPF file. For example:

   ```
   read power intent output.cpf -replace
   ```

4. Apply the CPF low power cell information.

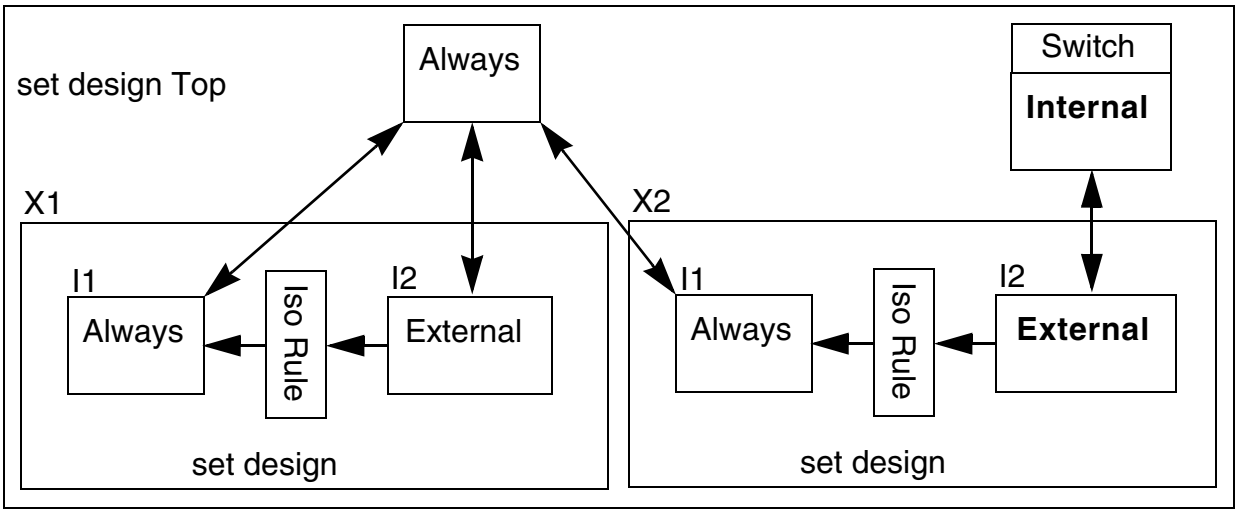
   ```
   commit power intent
   ```

5. Run power domain analysis, structural checks, and rule checks:

   ```
   analyze power domain
   ```

Before Integration

The following example, note the isolation rule between the always-on and external switchable cells for instance `X1`:

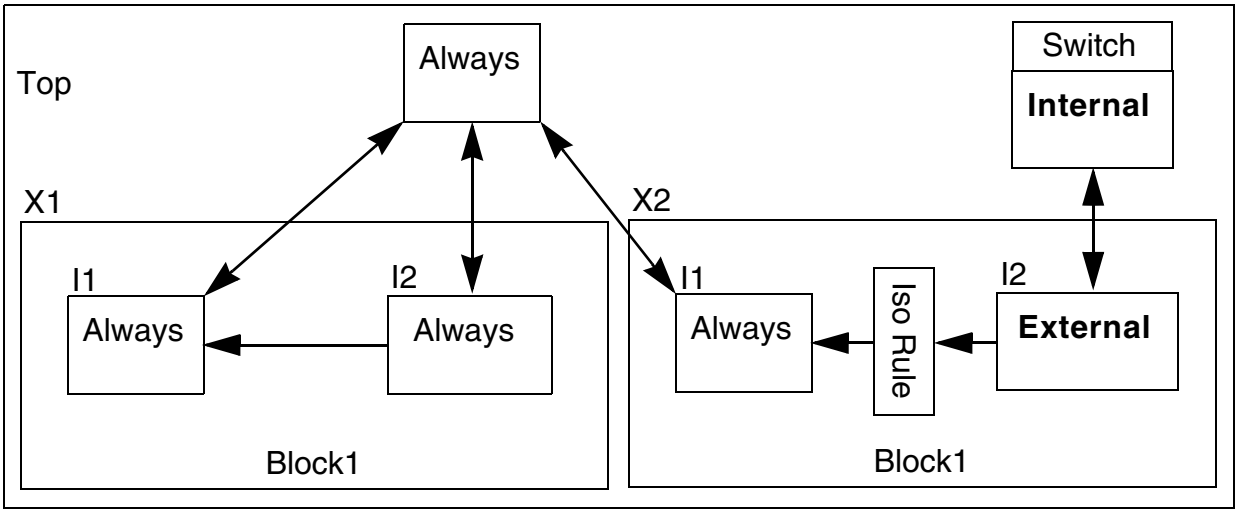


Isolation rule `Iso1` covers the crossing from `X1/I1` to `X1/I2`:

```
set_design Block1 -ports  { IsoExt }
    create_power_domain -name On -instances { I1 }
    create_power_domain -name Ext -external_controlled_shutoff -instances { I2 }
    create_isolation_rule -name Iso1 -from Ext -to On -isolation_condition IsoExt
end_design
set_design Top -ports { Iso Pwr }
    create_power_domain -name AON
    create_power_domain -name Int -shutoff_condition { !Pwr }
    set_instance X1 -domain_mapping { { On AON } { Ext AON } } -design Block1
    set_instance X2 -domain_mapping { { On AON } { Ext Int } } -design Block1 \
         -port_mapping { {IsoExt Iso} }
end_design
```

After Integration

After integration, because of domain mapping, the isolation rule crossing from `X1/I1` to `X1/I2` is removed. The crossing isolation rule crossing from `X2/I1` to `X2/I2` remains and is covered by `X2_Iso1`.



In the following, low-level blocks `X2/I1` and `X2/I2` are integrated to the domain definition at the top, and the low level isolation rule (`-name X2_Iso1`) is promoted to the top.

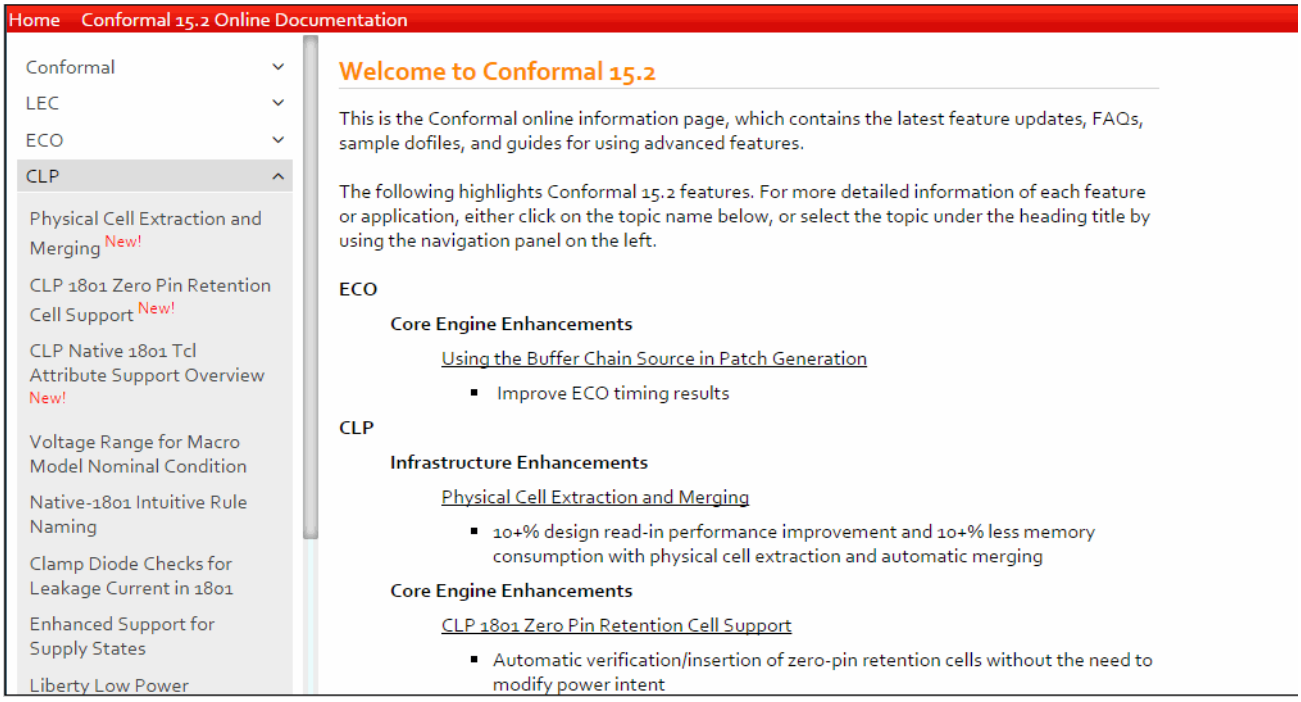
```
set_design Top -ports { Iso Pwr }
    create_power_domain -name AON -instances { X1 X2/I1 }
    create_power_domain -name Int -shutoff_condition { !Pwr } -instances { X2/I2 }
    create_isolation_rule -name X2_Iso1 -from Int -to AON -isolation_condition Iso
end_design
```

7

Running the Conformal Low Power 1801 Flow

To access the latest information regarding the native 1801 flow (such as features, guides, FAQs, and dofile scripts), launch the web interface using the `SET WEB_INTERFACE ON` command.

```
SETUP> set web_interface on
// Web Interface URL is http://server:8090 or http://server.company:8090
or http:ip_address:8090
// A modern browser supporting HTML5 is required
// File browsing is enabled. To limit access to documentation only,
use the -DOCOnly option.
```



The left-hand side lists all of the Conformal products. There is also a section called *Sample Dofiles* that provides sample dofiles (dofiles) for running CLP in different scenarios.

A

CPF Support in Conformal

General CPF Command Support

Note: Status can have the following values:

- S: supported
- U: unsupported
- NA: not applicable

Unless otherwise noted, the support status indicates the support status for Low Power Equivalency Checking (LP-EC) and Low Power Verify (LP-Verify, which includes CPF quality checks and Low Power design rule checks).

Command	Options	Valid in CPF Version		Comment	Support Status
assert_illegal_domain_configurations		1.1	2.0		NA
	-domain_conditions	1.1	2.0		NA
	-group_modes	1.1	2.0		NA
	-name	1.1	2.0		NA
create_analysis_view		1.1	2.0		NA
	-default	--	2.0		NA
	-domain_corners	1.1	2.0		NA
	-group_views	1.1	2.0		NA
	-mode	1.1	2.0		NA
	-name	1.1	2.0		NA
	-user_attributes	1.1	2.0		NA
create_assertion_control		1.1	2.0		NA
	-assertions	1.1	2.0		NA
	-domains	1.1	2.0		NA
	-exclude	1.1	2.0		NA
	-name	1.1	2.0		NA
	-shutoff_condition	1.1	2.0		NA
	-type	1.1	2.0		NA
create_bias_net		1.1	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
	-average_ir_drop_limit	1.1	2.0		NA
	-driver	1.1	2.0		U
	-net	1.1	2.0		S
	-peak_ir_drop_limit	1.1	2.0		NA
	-user_attributes	1.1	2.0		NA
create_global_connection		1.1	2.0		S
	-domain	1.1	2.0		S
	-instances	1.1	2.0		S
	-net	1.1	2.0		S
	-pg_type	--	2.0		U
	-pins	1.1	2.0		S
	-ports	--	2.0		NA
create_ground_nets		1.1	2.0		S
	-average_ir_drop_limit	1.1	2.0		NA
	-external_shutoff_condition	1.1	2.0		S
	-internal	1.1	2.0		S
	-nets	1.1	2.0		S
	-peak_ir_drop_limit	1.1	2.0		NA
	-user_attributes	1.1	2.0		NA
	-voltage	1.1	2.0		S
create_isolation_rule		1.1	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
	-exclude	1.1	2.0		S
	-force	--	2.0	S in LP-EC U in LP-Verify	S, U
	-from	1.1	2.0		S
	-isolation_condition	1.1	2.0		S
	-isolation_control	--	2.0		S
	-isolation_output	1.1	2.0		S
	-isolation_target	1.1	2.0		S
	-name	1.1	2.0		S
	-no_condition	1.1	2.0		S
	-pins	1.1	2.0		S
	-secondary_domain	1.1	2.0		S
	-to	1.1	2.0		S
create_level_shifter_rule		1.1	2.0		S
	-bypass_condition	--	2.0		U
	-exclude	1.1	2.0		S
	-force	--	2.0		U
	-from	1.1	2.0		S
	-input_domain	--	2.0		U
	-name	1.1	2.0		S
	-output_domain	--	2.0		U
	-pins	1.1	2.0		S
	-to	1.1	2.0		S
create_mode		--	2.0		U
	-condition	--	2.0		U
	-illegal	--	2.0		U
	-name	--	2.0		U
	-probability	--	2.0		U
create_mode_transition		1.1	2.0		NA

Command	Options	Valid in CPF Version		Comment	Support Status
	-assertions	--	2.0		NA
	-clock_pin	1.1	2.0		NA
	-cycles	1.1	2.0		NA
	-end_condition	1.1	2.0		NA
	-from	1.1	2.0		NA
	-illegal	--	2.0		NA
	-latency	1.1	2.0		NA
	-name	1.1	2.0		NA
	-start_condition	1.1	2.0		NA
	-to	1.1	2.0		NA
create_nominal_condition		1.1	2.0		S
	-deep_nwell_voltage	--	2.0		U
	-deep_pwell_voltage	--	2.0		U
	-ground_voltage	1.1	2.0		S
	-nmos_bias_voltage	1.1	2.0		S
	-name	1.1	2.0		S
	-pmos_bias_voltage	1.1	2.0		S
	-state	1.1	2.0		S
	-voltage	1.1	2.0		S
create_operating_corner		1.1	2.0		NA

Command	Options	Valid in CPF Version		Comment	Support Status
	-ground_voltage	1.1	2.0		NA
	-library_set	1.1	2.0		NA
	-name	1.1	2.0		NA
	-nmos_bias_voltage	1.1	2.0		NA
	-pmos_bias_voltage	1.1	2.0		NA
	-power_library_set	--	2.0		NA
	-process	1.1	2.0		NA
	-temperature	1.1	2.0		NA
	-voltage	1.1	2.0		NA
create_pad_rule		--	2.0		U
	-instances	--	2.0		U
	-mapping	--	2.0		U
	-name	--	2.0		U
	-of_bond_ports	--	2.0		U
create_power_domain		1.1	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
	-active_state_conditions	1.1	2.0		NA
	-base_domains	1.1	2.0		S
	-boundary_ports	1.1	2.0		S
	-default	1.1	2.0		S
	-default_isolation_condition	1.1	2.0	Supported in hierarchical CPF	S
	-default_restore_edge	1.1	2.0		S
	-default_restore_level	1.1	2.0		S
	-default_save_edge	1.1	2.0		S
	-default_save_level	1.1	2.0		S
	-exclude_instances	--	2.0		S
	-exclude_ports	--	2.0		S
	-external_controlled_shutoff	1.1	2.0		S
	-instances	1.1	2.0		S
	-name	1.1	2.0		S
	-power_down_states	--	2.0		NA
	-power_source	--	2.0		S
	-power_up_states	1.1	2.0		NA
	-shutoff_condition	1.1	2.0		S
create_power_mode		1.1	2.0		S
	-condition	--	2.0		U
	-default	1.1	2.0		S
	-domain_conditions	1.1	2.0		S
	-group_modes	1.1	2.0		S
	-name	1.1	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
create_power_nets		1.1	2.0		S
	-average_ir_drop_limit	1.1	2.0		NA
	-external_shutoff_condition	1.1	2.0		S
	-internal	1.1	2.0		S
	-nets	1.1	2.0		S
	-peak_ir_drop_limit	1.1	2.0		NA
	-user_attributes	1.1	2.0		NA
	-voltage	1.1	2.0		S
create_power_switch_rule		1.1	2.0		S
	-domain	1.1	2.0		S
	-external_ground_net	1.1	2.0		S
	-external_power_net	1.1	2.0		S
	-name	1.1	2.0		S
create_state_retention_rule		1.1	2.0		S
	-domain	1.1	2.0		S
	-exclude	1.1	2.0		S
	-instances	1.1	2.0		S
	-name	1.1	2.0		S
	-required	--	2.0		U
	-restore_edge	1.1	2.0		S
	-restore_level	1.1	2.0		S
	-restore_precondition	1.1	2.0		NA
	-retention_precondition	--	2.0		NA
	-save_edge	1.1	2.0		S
	-save_level	1.1	2.0		S
	-save_precondition	1.1	2.0		NA
	-secondary_domain	1.1	2.0		S
	-target_type	1.1	2.0		S
	-use_secondary_for_output	--	2.0		U

Command	Options	Valid in CPF Version		Comment	Support Status
define_library_set		1.1	2.0		S
	-libraries	1.1	2.0		S
	-name	1.1	2.0		S
	-user_attributes	1.1	2.0		NA
end_design		1.1	2.0		
	module	1.1	--	replaced with *power_design*	S
	power_design	--	2.0		S
end_macro_model		1.1	2.0		S
	macro_cell	1.1	2.0		S
end_power_mode_control_group		1.1	2.0		S
find_design_objects		--	2.0		S
	-direction	--	2.0		S
	-exact	--	2.0		S
	-hierarchical	--	2.0		S
	-ignore_case	--	2.0		S
	-leaf_only	--	2.0		S
	-non_leaf_only	--	2.0		S
	-object	--	2.0		S
	pattern	--	2.0	Note: Ports are not supported	S
	-pattern_type	--	2.0		S
	-regexp	--	2.0		U
	-scope	--	2.0		S
get_parameter		1.1	2.0		S
	parameter_name	1.1	2.0		S
identify_always_on_driver		1.1	2.0		S
	-pins	1.1	2.0		S
identify_power_logic		1.1	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
	-instances	1.1	2.0		S
	-module	1.1	2.0		U
	-type	1.1	2.0		S
identify_secondary_domain		1.1	2.0		S
	-cells	1.1	2.0		S
	-domain	1.1	2.0		S
	-from	1.1	2.0		S
	-instances	1.1	2.0		S
	-secondary_domain	1.1	2.0		S
	-to	1.1	2.0		S
include		1.1	2.0		S
	file	1.1	2.0		S
set_analog_ports		--	2.0		S
	port_list	--	2.0		S
	-user_attributes	--	2.0		S
set_array_naming_style		1.1	2.0		S
	string	1.1	2.0		S
set_cpf_version		1.1	2.0		S
	value	1.1	2.0		S
set_design		1.1	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
	-honor_boundary_port_domain	1.1	2.0		S
	-inout_ports	--	2.0		S
	-input_ports	--	2.0		S
	module	1.1	--	replaced with *power_design*	S
	-modules	--	2.0		U
	-output_ports	--	2.0		S
	-parameters	1.1	2.0		S
	-ports	1.1	2.0		S
	power_design	--	2.0		S
	-testbench	--	2.0		NA
set_diode_ports		--	2.0		S
	-negative	--	2.0		S
	-positive	--	2.0		S
set_equivalent_control_pins		1.1	2.0		S
	-domain	1.1	2.0		S
	-master	1.1	2.0		S
	-pins	1.1	2.0		S
	-rules	1.1	2.0		S
set_floating_ports		1.1	2.0		S
	port_list	1.1	2.0		S
set_hierarchy_separator		1.1	2.0		S
	character	1.1	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
set_input_voltage_tolerance		1.1	2.0		S
	-bias	1.1	--	replaced with -ground and -power	S
	-domain	--	2.0		S
	-ground	--	2.0		S
	-pins	--	2.0		S
	-ports	1.1	--	replaced with -pins	S
	-power	--	2.0		S
set_instance		1.1	2.0		S
	-design	1.1	2.0		S
	-domain_mapping	1.1	2.0		S
	instance	1.1	2.0		S
	-model	1.1	2.0		S
	-parameter_mapping	1.1	2.0		S
	-port_mapping	1.1	2.0		S
set_macro_model		1.1	2.0		S
	-cells	--	2.0		U
	macro_cell	1.1	2.0		S
set_pad_ports		--	2.0		S
	pin_list	--	2.0		S
set_power_mode_control_group		1.1	2.0		U
	-domains	1.1	2.0		U
	-groups	1.1	2.0		U
	-name	1.1	2.0		U
set_power_source_reference_pin		--	2.0		U

Command	Options	Valid in CPF Version		Comment	Support Status
	-domain	--	2.0		U
	pin	--	2.0		U
	-voltage_range	--	2.0		U
set_power_target		1.1	2.0		NA
	-dynamic	1.1	2.0		NA
	-leakage	1.1	2.0		NA
set_power_unit		1.1	2.0		NA
	{mW | nW | pW | uW |W}	1.1	2.0		NA
set_register_naming_style		1.1	2.0		S
	string%s	1.1	2.0		S
set_sim_control		--	2.0		NA
	-action	--	2.0		NA
	-controlling_domain	--	2.0		NA
	-domains	--	2.0		NA
	-exclude	--	2.0		NA
	-instances	--	2.0		NA
	-lib_cells	--	2.0		NA
	-modules	--	2.0		NA
	-targets	--	2.0		NA
	-type	--	2.0		NA
set_switching_activity		1.1	2.0		NA

Command	Options	Valid in CPF Version		Comment	Support Status
	-all	1.1	2.0		NA
	-clock_pins	1.1	2.0		NA
	-hierarchical	1.1	2.0		NA
	-instances	1.1	2.0		NA
	-mode	1.1	2.0		NA
	-pins	1.1	2.0		NA
	-probability	1.1	2.0		NA
	-toggle_percentage	1.1	2.0		NA
	-toggle_rate	1.1	2.0		NA
set_time_unit		1.1	2.0		NA
	{ms | ns | us}	1.1	2.0		NA
set_wire_feedthrough_ports		1.1	2.0		S
	port_list	1.1	2.0		S
update_design		--	2.0		U
	-name	--	2.0		U
update_isolation_rules		1.1	2.0		S
	-cells	1.1	2.0		S
	-domain_mapping	--	2.0		U
	-location	1.1	2.0		S
	-names	1.1	2.0		S
	-open_source_pins_only	1.1	2.0		U
	-pin_mapping	--	2.0		U
	-prefix	1.1	2.0		NA
	-suffix	--	2.0		NA
	-use_model	--	2.0		NA
	-within_hierarchy	1.1	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
update_level_shifter_rules		1.1	2.0		S
	-cells	1.1	2.0		S
	-location	1.1	2.0		S
	-names	1.1	2.0		S
	-prefix	1.1	2.0		NA
	-suffix	--	2.0		NA
	-through	--	2.0		U
	-within_hierarchy	1.1	2.0		S
update_nominal_condition		1.1	2.0		NA
	-library_set	1.1	2.0		NA
	-name	1.1	2.0		NA
	-power_library_set	--	2.0		NA
update_power_domain		1.1	2.0		S
	-boundary_ports	--	2.0		U
	-deep_nwell_net	--	2.0		U
	-deep_pwell_net	--	2.0		U
	-equivalent_ground_nets	1.1	2.0		S
	-equivalent_power_nets	1.1	2.0		S
	-instances	--	2.0		U
	-name	1.1	2.0		S
	-nmos_bias_net	1.1	2.0		S
	-pmos_bias_net	1.1	2.0		S
	-primary_ground_net	1.1	2.0		S
	-primary_power_net	1.1	2.0		S
	-transition_cycles	1.1	2.0		NA
	-transition_latency	1.1	2.0		NA
	-transition_slope	1.1	2.0		NA
	-user_attributes	1.1	2.0		NA

Command	Options	Valid in CPF Version		Comment	Support Status
update_power_mode		1.1	2.0		NA
	-activity_file	1.1	2.0		NA
	-activity_file_weight	1.1	2.0		NA
	-average_ir_drop_limit	1.1	2.0		NA
	-dynamic_power_limit	1.1	2.0		NA
	-hold_sdc_files	1.1	2.0		NA
	-leakage_power_limit	1.1	2.0		NA
	-name	1.1	2.0		NA
	-peak_ir_drop_limit	1.1	2.0		NA
	-sdc_files	1.1	2.0		NA
	-setup_sdc_files	1.1	2.0		NA
update_power_switch_rule		1.1	2.0		S
	-acknowledge_receiver_x	1.1	2.0		S
	-average_ir_drop_limit	1.1	2.0		NA
	-cells	1.1	2.0		S
	-enable_condition_x	1.1	2.0		S
	-gate_bias_net	1.1	2.0		U
	-name	1.1	2.0		S
	-peak_ir_drop_limit	1.1	2.0		NA
	-prefix	1.1	2.0		S
update_state_retention_rules		1.1	2.0		S
	-cells	1.1	2.0		S
	-cell_type	1.1	2.0		S
	-domain_mapping	--	2.0		U
	-names	1.1	2.0		S
	-pin_mapping	--	2.0		U
	-set_reset_control	1.1	2.0		U
	-use_model	--	2.0		S

Library Cell-Related CPF Command Support

Note: Status can have the following values:

- S: supported
- U: unsupported
- NA: not applicable to Conformal Low Power

Command	Options	Valid in CPF Version		Comment	Support Status
define_always_on_cell		1.1	2.0		S
	-cells	1.1	2.0		S
	-ground	1.1	2.0		S
	-ground_switchable	1.1	2.0		S
	-library_set	1.1	2.0		NA
	-power	1.1	2.0		S
	-power_switchable	1.1	2.0		S
define_global_cell		--	2.0		U
	-cells	--	2.0		U
	-enable	--	2.0		U
	-global_ground	--	2.0		U
	-global_power	--	2.0		U
	-isolated_pins	--	2.0		U
	-library_set	--	2.0		U
	-local_ground	--	2.0		U
	-local_power	--	2.0		U
	-power_off_function	--	2.0		U

Command	Options	Valid in CPF Version		Comment	Support Status
define_isolation_cell		1.1	2.0		S
	-always_on_pins	1.1	2.0		S
	-aux_enables	--	2.0		S
	-cells	1.1	2.0		S
	-clamp	--	2.0		S
	-enable	1.1	2.0		S
	-ground	1.1	2.0		S
	-ground_switchable	1.1	2.0		S
	-library_set	1.1	2.0		NA
	-no_enable	1.1	2.0		S
	-non_dedicated	1.1	2.0		S
	-pin_groups	--	2.0		U
	-power	1.1	2.0		S
	-power_switchable	1.1	2.0		S
	-valid_location	1.1	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
define_level_shifter_cell		1.1	2.0		S
	-always_on_pins	1.1	2.0		S
	-bypass_enable	--	2.0		U
	-cells	1.1	2.0		S
	-direction	1.1	2.0		S
	-enable	1.1	2.0		S
	-ground	1.1	2.0		S
	-ground_input_voltage_range	1.1	2.0		S
	-ground_output_voltage_range	1.1	2.0		S
	-input_ground_pin	1.1	2.0		S
	-input_power_pin	1.1	2.0		S
	-input_voltage_range	1.1	2.0		S
	-library_set	1.1	2.0		NA
	-multi_stage	--	2.0		U
	-output_ground_pin	1.1	2.0		NA
	-output_power_pin	1.1	2.0		NA
	-output_voltage_range	1.1	2.0		S
	-pin_groups	--	2.0		U
	-power	1.1	2.0		S
	-valid_location	1.1	2.0		S
define_open_source_input_pin		1.1	2.0		S
	-cells	1.1	2.0		S
	-library_set	1.1	2.0		NA
	-pin	1.1	2.0		S
	-type	--	2.0		U

Command	Options	Valid in CPF Version		Comment	Support Status
define_pad_cell		--	2.0		U
	-analog_pins	--	2.0		U
	-cells	--	2.0		U
	-enable	--	2.0		U
	-isolated_pins	--	2.0		U
	-pad_pins	--	2.0		U
	-pin_groups	--	2.0		U
define_power_clamp_cell		1.1	2.0		S
	-cells	1.1	2.0		S
	-data	1.1	2.0		S
	-ground	1.1	2.0		S
	-library_set	1.1	2.0		NA
	-power	1.1	2.0		S
define_power_clamp_pins		--	2.0		S
	-cells	--	2.0		S
	-data_pin	--	2.0		S
	-ground	--	2.0		S
	-library_set	--	2.0		NA
	-power	--	2.0		S
	-type	--	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
define_power_switch_cell		1.1	2.0		S
	-cells	1.1	2.0		S
	-enable_pin_bias	1.1	2.0		S
	-gate_bias_pin	1.1	2.0		U
	-ground	1.1	2.0		S
	-ground_switchable	1.1	2.0		S
	-leakage_current	1.1	2.0		NA
	-library_set	1.1	2.0		NA
	-power	1.1	2.0		S
	-power_switchable	1.1	2.0		S
	-stage_x_enable	1.1	2.0		S
	-stage_x_on_resistance	1.1	2.0		NA
	-stage_x_output	1.1	2.0		S
	-stage_x_saturation_current	1.1	2.0		NA
	-type	1.1	2.0		S
define_related_power_pins		1.1	2.0		S
	-cells	1.1	2.0		S
	-data_pins	1.1	2.0		S
	-ground	1.1	2.0		S
	-library_set	1.1	2.0		NA
	-power	1.1	2.0		S

Command	Options	Valid in CPF Version		Comment	Support Status
define_state_retention_cell		1.1	2.0		S
	-always_on_components	1.1	2.0		NA
	-always_on_pins	1.1	2.0		S
	-cells	1.1	2.0		S
	-cell_type	1.1	2.0		S
	-clock_pin	1.1	2.0		S
	-ground	1.1	2.0		S
	-ground_switchable	1.1	2.0		S
	-library_set	1.1	2.0		NA
	-power	1.1	2.0		S
	-power_switchable	1.1	2.0		S
	-restore_check	1.1	2.0		NA
	-restore_function	1.1	2.0		S
	-retention_check	--	2.0		NA
	-save_check	1.1	2.0		NA
	-save_function	1.1	2.0		S

B

1801 Support

The following table summarizes the 1801 support status for Low Power Equivalency Checking (LP-EC) and Low Power Verify (LP-Ver, which includes CPF quality checks and Low Power design rule checks). Status can have the following values:

Abbreviation	Meaning	
NA	Not Applicable	
NS	Not supported	Currently not supported but may be supported in the future
PS	Partially supported	
S	Supported	
*	Beta quality, not to be used for production	
LA	Limited access, production quality	
V	UPF or IEEE 1801 Version	The earliest version that this command/ option exits

Commands	Options	V	LP-EC	LP-Ver
General Commands				
add_parameter		3.0	NA	NA
	-type	3.0	NA	NA
	-default	3.0	NA	NA

Commands	Options	V	LP-EC	LP-Ver
	-description	3.0	NA	NA
add_domain_elements (deprecated)		1.0	NS	NS
	-elements	1.0	NS	NS
add_port_state (legacy)		1.0	S	S
	-state	1.0	S	S
add_power_state		2.0	S	S
	-state	2.0	S	S
	-supply_expr	2.0	S	S
	-logic_expr	2.0	S	S
	-simstate	2.0	S	S
	-legal | -illegal	2.0	NS	NS
	-update	2.0	S	S
	-complete	2.1	NS	NS
	-supply | domain	2.1	S	S
	-group	3.0	S	S
	-model	3.0		
	-instance	3.0		
add_pst_state (legacy)		1.0	S	S
	-pst	1.0	S	S
	-state	1.0	S	S
add_state_transition		3.0	NA	NA
	-transition	3.0	NA	NA
	-supply | -domain | -group | -model	3.0	NA	NA
	-instance	3.0	NA	NA
	-update	3.0	NA	NA
	-from	3.0	NA	NA
	-paired	3.0	NA	NA

Commands	Options	V	LP-EC	LP-Ver
	-legal | -illegal	3.0	NA	NA
	-complete	3.0	NA	NA
add_supply_state		3.0	NA	NA
	-state	3.0	NA	NA
apply_power_model		2.1	S	S
	-elements	2.1	S	S
	-supply_map	2.1	S	S
	-parameters	3.0	NS	NS
	-port_map	3.1	S	S
associate_supply_set		2.0	S	S
	-handle	2.0	S	S
begin_power_model (legacy)		2.1	S	S
	-for	2.1	S	S
bind_checker (linter only)		1.0	NA	NA
	-module (linter only)	1.0	NA	NA
	-elements (linter only)	1.0	NA	NA
	-ports (linter only)	1.0	NA	NA
	-bind_to (linter only)	2.0	NA	NA
	-parameters	3.0	NA	NA
connect_logic_net		1.0	S	S
	-ports	1.0	S	S
	-reconnect	2.1	S	S
connect_supply_net		1.0	S	S
	-ports	1.0	S	S
	-pg_type	1.0	NS	NS
	-vct	1.0	NA	NA
	-domain	1.0	NS	NS

Commands	Options	V	LP-EC	LP-Ver
	-cells	1.0	NS	NS
	-pins (deprecated in IEEE 1801-2013)	1.0	NS	NS
	-rail_connection (deprecated in IEEE 1801-2013)	1.0	NS	NS
	-elements	3.0	NS	NS
connect_supply_set		2.0	NS	NS
	-connect	2.0	NS	NS
	-elements	2.0	NS	NS
	-exclude_elements	2.0	NS	NS
	-transitive	2.0	NS	NS
create_composite_domain		2.0	NS	NS
	-subdomains	2.0	NS	NS
	-supply	2.0	NS	NS
	-update	2.0	NS	NS
create_hdl2upf_vct		1.0	NA	NA
	-hdl_type	1.0	NA	NA
	-table	1.0	NA	NA
create_logic_net		2.0	S	S
create_logic_port		2.0	S	S
	-direction	2.0	S	S
create_power_domain		1.0	S	S
	-simulation_only (deprecated in IEEE 1801-2015)	1.0	NA	NA
	-elements	2.0	S	S
	-elements {.}	2.1	S	S
	-exclude_elements	2.0	S	S
	-include_scope (deprecated in IEEE 1801-2013)	1.0	S	S

Commands	Options	V	LP-EC	LP-Ver
	-supply	2.0	S	S
	-supply {extra_supplies_n}		S	S
	-scope (deprecated in IEEE 1801-2013)	1.0	NA	NA
	-define_func_type	2.0	NA	NA
	-update	2.0	S	S
	-available_supplies	2.1	S	S
	-atomic	2.1	NA	NA
	-subdomains	3.0	NS	NS
	-power_up_states	C	NA	NA
	-power_down_states	C	NA	NA
create_power_state_group		3.0	S	S
create_power_switch		1.0	S	S
	-output_supply_port	1.0	S	S
	-input_supply_port	1.0	S	S
	-control_port	1.0	S	S
	-on_state	1.0	S	S
	-off_state	1.0	S	S
	-supply_set	2.0	S	S
	-on_partial_state	1.0	NS	NS
	-ack_port	1.0	S	S
	-ack_delay	1.0	NA	NA
	-error_state	1.0	NS	NS
	-domain	1.0	S	S
	-instance (changed to -instances in IEEE 1801-2013)	2.0	S	S
	-switch_type	3.1	NS	NS
	-output_voltage	C	NA	NA
	-update	2.0	S	S

Commands	Options	V	LP-EC	LP-Ver
create_pst (legacy)		1.0	S	S
	-supplies	1.0	S	S
create_supply_net		1.0	S	S
	-domain	1.0	S	S
	-reuse	1.0	S	S
	-resolve	1.0	NA	NA
create_supply_port		1.0	S	S
	-domain	1.0	S	S
	-direction	1.0	S	S
create_supply_set		2.0	S	S
	-function	2.0	S	S
	-reference_gnd (deprecated in IEEE 1801-2015)	2.0	NA	NA
	-update	2.0	S	S
create_upf2hdl_vct		1.0	NA	NA
	-hdl_type	1.0	NA	NA
	-table	1.0	NA	NA
define_power_model		3.1	S	S
	-for	3.1	S	S
describe_state_transition (deprecated)		2.0	NA	NA
	-object	2.0	NA	NA
	-from -to	2.0	NA	NA
	-paired	2.0	NA	NA
	-legal | -illegal	2.0	NA	NA
end_power_model (legacy)		2.1	S	S
find_objects		2.0	S	S

Commands	Options	V	LP-EC	LP-Ver
	scope	2.0	S	S
	-pattern	2.0	S	S
	-object_type	2.0	S	S
	-direction	2.0	S	S
	-transitive	2.0	S	S
	-regexp | -exact	2.0	S	S
	-ignore_case	2.0	S	S
	-non-leaf | -leaf_only	2.0	NA	NA
	-traverse_macros	3.1		
load_simstate_behavior		2.0	NA	NA
	-file	2.0	NA	NA
load_upf		1.0	S	S
	-scope	1.0	S	S
	-version (deprecated in IEEE 1801-2015)	1.0	S	S
	-hide_globals	3.0		
	-parameters	3.0		
load_upf_protected (deprecated)		2.0	S	S
	-hide_globals	2.0	NS	NS
	-version	2.0	S	S
	-scope	2.0	S	S
	-params	2.0	NS	NS
map_isolation_cell (deprecated)		1.0	S	S
	-domain	1.0	S	S
	-elements	1.0	S	S
	-lib_cells	1.0	S	S

Commands	Options	V	LP-EC	LP-Ver
	-lib_cell_type	1.0	NS	NS
	-lib_model	1.0	NS	NS
	-port	1.0	NA	NA
	-lib_cells	1.0	S	S
map_level_shifter_cell (deprecated)		1.0	S	S
	-domain	1.0	S	S
	-lib_cells	1.0	S	S
	-elements	1.0	NA	NA
map_power_switch		1.0	S	S
	-domain (deprecated in IEEE 1801-2013)	1.0	S	S
	-lib_cells	1.0	S	S
	-port_map	2.0	S	S
map_repeater_cell		3.0	S	S
	-domain	3.0	S	S
	-elements	3.0	NS	NS
	-exclude_elements	3.0	NS	NS
	-lib_cells	3.0	S	S
map_retention_cell		1.0	S	S
	-domain	1.0	S	S
	-elements	1.0	S	S
	-exclude_elements	1.0	S	S
	-lib_cells	1.0	S	S
	-lib_cells_type	1.0	S	S
	-lib_model_name	2.1	NS	NS
	-port	2.1	NS	NS

Commands	Options	V	LP-EC	LP-Ver
merge_power_domains (deprecated)		1.0	NS	NS
	-power_domains	1.0	NS	NS
	-scope	1.0	NS	NS
	-all_equivalent	1.0	NS	NS
name_format		1.0	S	S
	-isolation_prefix	1.0	S	S
	-isolation_suffix	1.0	S	S
	-level_shift_prefix	1.0	S	S
	-level_shift_suffix	1.0	S	S
	-implicit_suppot_suffix	2.0	NS	NS
	-implicit_logic_prefix	2.0	S	S
	-implicit_logic_suffix	2.0	S	S
save_upf		1.0	NA	NA
	-scope	1.0	NA	NA
	-version (deprecated in IEEE 1801-2013)	1.0	NA	NA
set_correlated		3.0	S	S
	-nets	3.0	S	S
	-sets	3.0	S	S
set_design_attributes		2.0	S	S
	-elements	2.0	S	S
	-exclude_elements	2.0	NA	NA
	-attribute	2.0	S	S
	-models	2.0	S	S
	-is_leaf_cell	2.1	NS	NS
	-is_macro_cell	2.1	NS	NS
	-is_soft_macro	3.0	S	S

Commands	Options	V	LP-EC	LP-Ver
	-is_hard_macro	3.0	NS	NS
	-is_power_aware_model	3.1		
	-switch_cell_type	3.0	NS	NS
set_design_top		1.0	NA	NA
	-testbench	C	NA	NA
set_domain_supply_net (legacy)		1.0	S	S
	-primary_power_net	1.0	S	S
	-primary_ground_net	1.0	S	S
set_equivalent		2.1	S	S
	-function_only	2.1	NS	NS
	-nets	2.1	S	S
	-sets	2.1	S	S
	-interchangeable	3.1	NS	NS
set_isolation		1.0	S	S
	-domain	1.0	S	S
	-elements	1.0	S	S
	-exclude_elements	2.1	S	S
	-source	2.0	S	S
	-sink	2.0	S	S
	-applies_to	2.0	S	S
	-applies_to_boundary	3.0	S	S
	-applies_to_clamp	2.0	S	S
	-applies_to_sink_off_ clamp	2.0	NA	NA
	-applies_to_source _off_clamp	2.0	NA	NA
	-isolation_power_net (legacy)	1.0	S	S

Commands	Options	V	LP-EC	LP-Ver
	-isolation_ground_net (legacy)	1.0	S	S
	-no_isolation	1.0	S	S
	-isolation_supply_set (changed to -isolation_supply in IEEE 1801-2015)	2.0	S	S
	-isolation_supply	3.0	S	S
	-isolation_signal	2.0	S	S
	-isolation_sense	2.0	S	S
	-name_prefix	2.0	S	S
	-name_suffix	2.0	S	S
	-clamp_value	2.0	S	S
	-sink_off_clamp (deprecated in IEEE 1801-2013)	2.0	NA	NA
	-source_off_clamp (deprecated in IEEE 1801-2013)	2.0	NA	NA
	-location	2.0	S	S
	-force_isolation	2.0	S	S
	-instance	2.0	NS	NS
	-diff_supply_only	2.0	S	S
	-transitive (deprecated in IEEE 1801-2013)	2.0	NS	NS
	-update	2.0	S	S
	-use_equivalence (deprecated in IEEE 1801-2018)	2.1	NS	NS
	-use_functional_equivalence	3.1	NS	NS
set_isolation_control (deprecated)		1.0	S	S
	-domain	1.0	S	S
	-isolation_signal	1.0	S	S
	-isolation_sense	1.0	S	S

Commands	Options	V	LP-EC	LP-Ver
	-location	1.0	S	S
set_level_shifter		1.0	S	S
	-domain	1.0	S	S
	-elements	1.0	S	S
	-exclude_elements	2.1	NS	NS
	-no_shift	1.0	S	S
	-threshold	1.0	S	S
	-force_shift	2.0	S	S
	-source	2.0	S	S
	-sink	2.0	S	S
	-applies_to	1.0	S	S
	-rule	1.0	S	S
	-location	1.0	S	S
	-name_prefix	2.0	S	S
	-name_suffix	2.0	S	S
	-input_supply_set (change to -input_supply in IEEE 1801-2015)	2.0	S	S
	-output_supply_set (change to -output_supply in IEEE 1801-2015)	2.0	S	S
	-internal_supply_set (change to -internal_supply in IEEE 1801-2015)	2.0	S	S
	-instance	1.0	NA	NA
	-transitive (deprecated in IEEE 1801-2013)	1.0	NA	NA
	-update	1.0	S	S

Commands	Options	V	LP-EC	LP-Ver
	-use_equivalence (deprecated in IEEE 1801-2018)	2.1	NS	NS
	-use_functional_equivalence	3.1	NS	NS
set_partial_on_translation		2.0	NA	NA
	[OFF | FULL_ON]	2.0	NA	NA
	-full_on_tools (deprecated in IEEE 1801-2013)	2.0	NA	NA
	-off_tools (deprecated in IEEE 1801-2013)	2.0	NA	NA
set_pin_related_supply (deprecated)		1.0	S	S
	-pins	1.0	S	S
	-related_power_pin	1.0	S	S
	-related_ground_pin	1.0	S	S
set_port_attributes		2.0	S	S
	[-ports | -exclude_ports]	2.0	S	S
	-applies_to		S	S
	-domains (deprecated in IEEE 1801-2013)	2.0	NA	NA
	-exclude_domains (deprecated in IEEE 1801-2013)	2.0	NA	NA
	-elements	2.0	S	S
	-exclude_elements	2.0	S	S
	-model	2.0	S	S
	-attribute	2.0	S	S
	-clamp_value	2.0	S	S
	-sink_off_clamp	2.0	NA	NA
	-source_off_clamp	2.0	NA	NA
	-receiver_supply	2.0	S	S

Commands	Options	V	LP-EC	LP-Ver
	-driver_supply	2.0	S	S
	-related_power_port	2.0	S	S
	-related_ground_port	2.0	S	S
	-related_bias_port	2.0	S	S
	-repeater_supply (deprecated in IEEE 1801-2013)	2.0	S	S
	-pg_type	2.0	S	S
	-transitive (deprecated in IEEE 1801-2013)	2.0	NA	NA
	-feedthrough	2.1	S	S
	-unconnected	2.1	S	S
	-is_analog	3.0	S	S
	-is_isolated	3.0	S	S
	-literal_supply	3.0	NS	NS
set_repeater		2.1	S	S
	-domain	2.1	S	S
	-elements	2.1	S	S
	-exclude_elements	2.1	S	S
	-source	2.1	S	S
	-sink	2.1	S	S
	-use_equivalence (deprecated in IEEE 1801-2018)	2.1	NS	NS
	-use_functional_equivalence	3.1	NS	NS
	-applies_to	2.1	S	S
	-applies_to_boundary	3.0	NS	NS
	-repeater_supply_set (changed to -repeat_supply in UPF 3.0)	2.1	S	S
	-repeater_supply	3.0	S	S
	-name_prefix	2.1	S	S

Commands	Options	V	LP-EC	LP-Ver
	-name_suffix	2.1	S	S
	-instance	2.1	NA	NA
	-update	2.1	S	S
set_power_switch (deprecated)		1.0	NS	NS
	-output_supply_port	1.0	NS	NS
	-input_supply_port	1.0	NS	NS
	-control_port	1.0	NS	NS
	-on_state	1.0	NS	NS
	-supply_set	2.0	NS	NS
	-on_partial_state	1.0	NS	NS
	-off_state	1.0	NS	NS
	-error_state	1.0	NS	NS
set_repeater		2.1	S	S
	-domain	2.1		
	-elements	2.1	S	S
	-exclude_elements	2.1	S	S
	-source	2.1	S	S
	-sink	2.1	S	S
	-use_equivalence (deprecated in IEEE 1801-2018)	2.1	NA	NA
	-use_functional_equivalence	3.1		
	-applies_to	2.1	S	S
	-applies_to_boundary	3.0		
	-repeater_supply_set (changed to -repeater_supply in UPF 3.0)	2.1	S	S
	-repeater_supply	3.0		
	-instance	2.1	NA	NA

Commands	Options	V	LP-EC	LP-Ver
	-update	2.1	S	S
	-name_prefix	2.1	S	S
	-name_suffix	2.1	S	S
set_retention		1.0	S	S
	-domain	1.0	S	S
	-elements	1.0	S	S
	-exclude_elements	2.0	S	S
	-retention_power_net (legacy)	1.0	S	S
	-retention_ground_net (legacy)	1.0	S	S
	-retention_supply_set (changed to -retention_supply in IEEE 1801-2015)	2.0	S	S
	-retention_supply	3.0	S	S
	-no_retention	2.0	S	S
	-save_signal	2.0	S	S
	-restore_signal	2.0	S	S
	-save_condition	2.0	NA	NA
	-restore_condition	2.0	NA	NA
	-retention_condition	2.0	S	S
	-use_retention_as_ primary	2.0	S	S
	-parameters	2.0	NS	NS
	-instance	2.0	NS	NS
	-transitive	2.0	NS	NS
	-update	2.0	S	S
set_retention_control (deprecated)		1.0	S	S
	-domain	1.0	S	S
	-save_signal	1.0	S	S

Commands	Options	V	LP-EC	LP-Ver
	-restore_signal	1.0	S	S
	-assert_r_mutex	1.0	NA	NA
	-assert_s_mutex	1.0	NA	NA
	-assert_rs_mutex	1.0	NA	NA
set_retention_elements		2.0	S	S
	-elements	2.0	S	S
	-applies_to	2.0	NA	NA
	-exclude_elements	2.0	S	S
	-retention_purpose	2.0	NA	NA
	-transitive	2.0	NA	NA
	-expand (deprecated in IEEE 1801-2013)	2.0	NA	NA
set_scope		1.0	S	S
set_simstate_behavior		2.0	NA	NA
	-lib	2.0	NA	NA
	-model (changed to -models in IEEE 1801-2015)	2.0	NA	NA
	-elements	2.0	NA	NA
	-exclude_elements	2.0	NA	NA
set_variation		3.0	NS	NS
	-supply	3.0	NS	NS
	-range	3.0	NS	NS
sim_assertion_control		3.1		
	-control_expr	3.1		
	-controlling_domain	3.1		
	-domain	3.1		
	-elements	3.1		
	-exclude_elements	3.1		

Commands	Options	V	LP-EC	LP-Ver
	-model	3.1		
	-type	3.1		
	-transitive	3.1		
sim_corruption_control		3.1		
	-domain	3.1		
	-elements	3.1		
	-exclude_elements	3.1		
	-model	3.1		
	-type	3.1		
	-transitive	3.1		
sim_replay_control		3.1		
	-controlling_domain	3.1		
	-domain	3.1		
	-elements	3.1		
	-exclude_elements	3.1		
	-model	3.1		
	-transitive	3.1		
upf_version		2.0	S	S
use_interface_cell		2.0	S	S
	-strategy	2.0	S	S
	-domain	2.0	S	S
	-lib_cells	2.0	S	S
	-map	2.0	S	S
	-elements	2.0	S	S
	-exclude_elements	2.0	S	S
	-applies_to_clamp	2.0	NS	NS
	-update_any (deprecated in IEEE 1801-2018)	2.0	NA	NA

Commands	Options	V	LP-EC	LP-Ver
	-force_function	2.0	NS	NS
	-inverter_supply_set	2.0	NA	NA
query_design_attributes		2.0	NS	NS
query_port_state		2.0	NS	NS
query_power_domain		2.0	NS	NS
	-no_elements	2.0	NS	NS
	-non_leaf | -all	2.0	NS	NS
	-detailed	2.0	NS	NS
query_power_state	ALL	2.0	NS	NS
query_power_switch	ALL	2.0	NS	NS
query_pst	ALL	2.0	NS	NS
query_pst_state	ALL	2.0	NS	NS
query_retention	ALL	2.0	NS	NS
query_retention_control	ALL	2.0	NS	NS
Power Management Cell Commands				
define_always_on_cell		2.1	S	S
	-cells	2.1	S	S
	-power	2.1	S	S
	-ground	2.1	S	S
	-power_switchable	2.1	S	S
	-ground_switchable	2.1	S	S
	-isolated_pins	2.1	NS	NS
	-enable	2.1	NS	NS
define_diode_clamp		2.1	S	S
	-cells	2.1	S	S
	-data_pins	2.1	S	S

Commands	Options	V	LP-EC	LP-Ver
	-type	2.1	S	S
	-power	2.1	S	S
	-ground	2.1	S	S
define_isolation_cell		2.1	S	S
	-cells	2.1	S	S
	-power	2.1	S	S
	-ground	2.1	S	S
	-enable	2.1	S	S
	-always_on_pins	2.1	NS	NS
	-power_switchable	2.1	S	S
	-ground_switchable	2.1	S	S
	-non_dedicated	2.1	S	S
define_level_shifter_cell		2.1	S	S
	-cells	2.1	S	S
	-input_voltage_range	2.1	S	S
	-output_voltage_range	2.1	S	S
	-ground_input_ voltage_range	2.1	NS	NS
	-ground_output_ voltage_range	2.1	NS	NS
	-direction	2.1	S	S
	-input_power_pin	2.1	S	S
	-output_power_pin	2.1	S	S
	-input_ground_pin	2.1	S	S
	-output_ground_pin	2.1	S	S
	-ground	2.1	S	S
	-enable	2.1	S	S
	-valid_location	2.1	S	S

Commands	Options	V	LP-EC	LP-Ver
	-bypass_enable	2.1	NS	NS
	-multi_stage	2.1	NS	NS
define_power_switch_cell		2.1	S	S
	-cells	2.1	S	S
	-type	2.1	S	S
	-stage_1_enable	2.1	S	S
	-power_switchable	2.1	S	S
	-ground_switchable	2.1	S	S
	-stage_2_enable	2.1	S	S
	-always_on_pins	2.1	NS	NS
	-gate_bias_pin	2.1	S	S
define_retention_cell		2.1	S	S
	-cells	2.1	S	S
	-power	2.1	S	S
	-ground	2.1	S	S
	-cell_type	2.1	S	S
	-always_on_pins	2.1	S	S
	-restore_function	2.1	S	S
	-save_function	2.1	S	S
	-restore_check	2.1	NA	NA
	-save_check	2.1	NA	NA
	-retention_check	2.1	NA	NA
	-hold_check	2.1	NA	NA
	-always_on_ components	2.1	NA	NA
	-power_switchable	2.1	S	S
	-ground_switchable	2.1	S	S

Commands	Options	V	LP-EC	LP-Ver
Non-Standard Extensions				
wildcard*	connect_supply_net -ports		S	S
	set_isolation -elements -exclude_elements		S	S
	set_level_shifter -elements -exclude_elements		S	S
	set_repeater -elements -exclude_elements		S	S
	set_retention -elements -exclude_elements		S	S
	set_port_attributes -ports		S	S
set_related_supply_net	-object_list		S	S
	-ground		S	S
	-power		S	S
	-reset		NA	NA
create_power_domain	-supply {extra_supplies_n}		S	S
set_design_attributes	-attribute {iso_nor TRUE}		NS	NS
	-elements {strategy_name}		NS	NS
set_sim_control	-targets	C	NA	NA
	-action power_up_replay	C	NA	NA
	-action disable_corruption	C	NA	NA
	-action {disable_isolation | disable_retention}	C	NA	NA
	-domains	C	NA	NA
	-instances	C	NA	NA

Commands	Options	V	LP-EC	LP-Ver
	-modules	C	NA	NA
	-libcells	C	NA	NA
	-disable_timing_ warnings	C	NA	NA
create_assertion_control	-name	C	NA	NA
	-assertions	C	NA	NA
	-exclude	C	NA	NA
	-assertion_control	C	NA	NA
	-type	C	NA	NA
	-supply_set	C	NA	NA