Si2 Common Power Format Specification™

Version 1.0

02 January 2007

Published by
Silicon Integration Initiative, Inc. (Si2™)
9111 Jollyville Road, Suite 250
Austin TX 78759

ISBN: 1-882750-20-9

TO THE SI2 COMMON POWER FORMAT, THE SI2 CPF SPECIFICATION OR ANY MATERIALS OR INFORMATION THEREIN, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGES.

Attention is called to the possibility that implementation of this specificatin may require use of subject matter covered by patent rights under which a license may be required . Si2 shall not be responsible for identifying patents or patent applications for which a license may be required to implement an Si2 specification or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention. By publication of this document, Si2 takes no position with respect to the existence or validity of any patent rights.

Si2 makes no representation as to the reasonableness or nondiscriminatory nature of the terms and conditions of the license agreements offered by any such holder(s). Further information may be obtained from Si2 upon request .

No assurances are provided that the CPF Speification will be compatible with subsequent versions or with any version that may be implemented in any products or technology.

Silicon Integration Initiative, 9111 Jollyville Rd., #250, Austin, TX 78759

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Si2 or any LPC member company. Except as may be explicitly set forth in such agreement, neither Si2 nor the LPC member companies make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Neither Si2 nor the LPC member companies warrant that use of such information will not infringe any third party rights, neither does Si2 nor the LPC member companies assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Document Status

Released version: si2_cpf_v1.0_01-02-2007.pdf

This section reflects the status of this document at the time of its publication. Other documents may supersede this document. Please contact Si2 for a complete list of current Si2 publications produced by the LPC.

Formal comments and error reports on this document should be sent to Si2 Common Power Format Project on http://openeda.si2.org.

The Si2 CPF Specification Version 1.0 (2 January 2007) was approved by the LPC Technical Steering Group and Low Power Coalition.

Revision History

Version	Date	Comments
1.0	02-Jan-2007	Initial specification

Acknowledgements

This document has been produced by the LPC TSG as part of the Low Power Coalition.

Editor

Susan Carver	Si2

LPC Technical Steering Group

Srikanth Jadcheria	ArchPro
Steev Wilcox	Azuro
Gila Kamhi	Intel
Dan Shimzu	AMD
Qi Wang	Cadence
Dave Allen	Atrenta
Will Ruby	Golden Gate
Tom Miller	Sequence
Arvind Narayanan	Magma Design Automation
Gary Delp	LSI Logic
Herve Menager	NXP
Ming Chau	Texas Instruments

Contents

Preface

Documentation Conventions

To aid the readers understanding, a consistent formatting style has been used throughout this manual.

The list below describes the syntax conventions used for the CPF constraints.

`literal`	Nonitalic words indicate keywords that you must type literally. These keywords represent command or option names.
`arguments and options`	Words in italics indicate user-defined arguments or options for which you must substitute a name or a value.
|	Vertical bars (OR-bars) separate possible choices for a single argument.
`[ ]`	Brackets denote options. When used with OR-bars, they enclose a list of choices from which you can choose one.
`{ }`	Braces denote arguments and are used to indicate that a choice is required from the list of arguments separated by OR-bars. You must choose one from the list. `{ argument1 | argument2 | argument3 }`
`...`	Three dots (...) indicate that you can repeat the previous argument. If the three dots are used with brackets (that is, `[argument]...`), you can specify zero or more arguments. If the three dots are used without brackets (`argument...`), you must specify at least one argument, but can specify more.
#	The pound sign precedes comments.

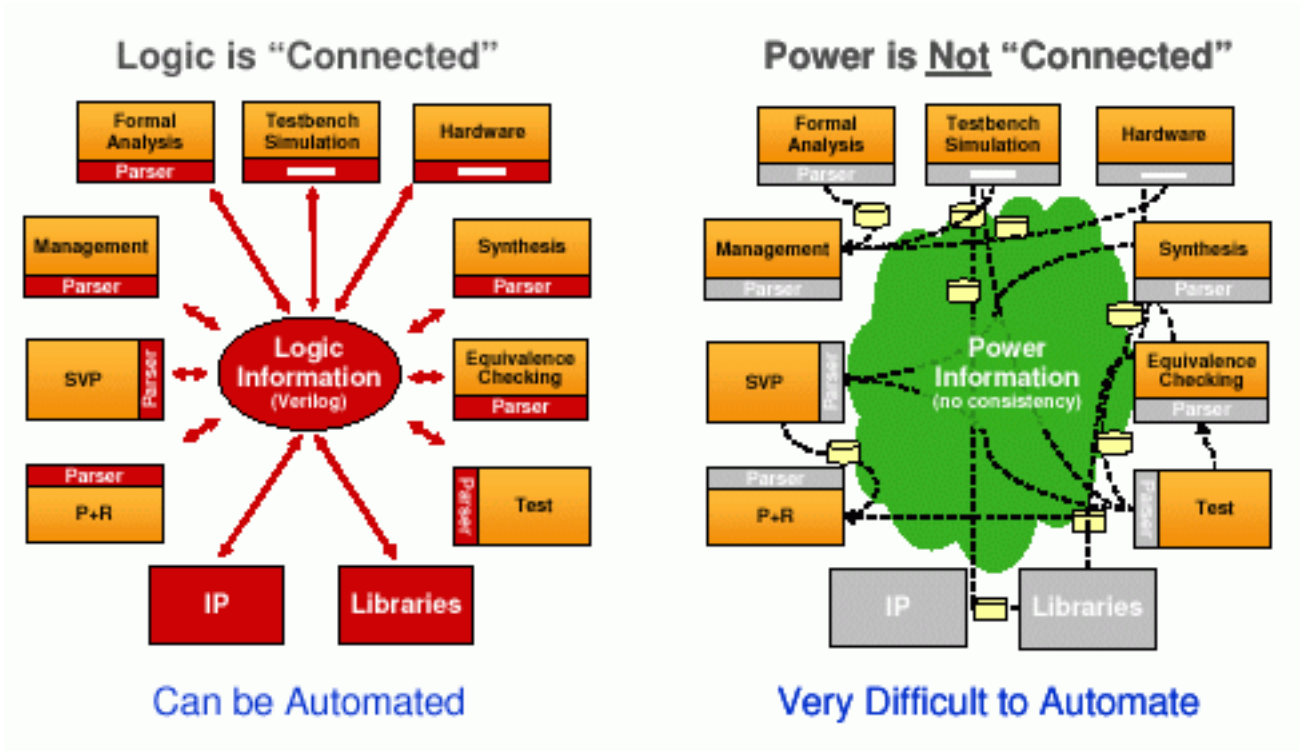
1

Introducing the Common Power Format

Introduction

The shift in the use of chips to consumer applications and the change in the latest process technologies have made power one of the primary design criteria for a majority of the chips worldwide. However, the industry's design infrastructure has not evolved at the same pace. Figure1-1 shows the mature state of the infrastructure for functional designs versus the chaotic state of the infrastructure for designs using advanced low power design techniques.

Figure 1-1 Comparison of State of Infrastructures for Functional Designs and Advanced Low Power Designs



To accomplish an industry-wide solution for this industry-wide problem, every effort was made to use an open and inclusive approach to create a complete and well architected solution.

The lack of support in the infrastructure for designs using advanced low power design techniques has resulted in a gap between the design techniques needed to control power dissipation and the ability of the design environment to support those techniques in a safe and efficient manner. The **C**ommon **P**ower **F**ormat has been architected to supply the infrastructure needed to support the state of the art in low power design styles and techniques.

The requirements for the Common Power Format were created using a wide range of viewpoints and with a broad range of applications in mind:

■	Semiconductor manufacturing equipment	■	High-end graphics processing
■	Semiconductor manufacturing (foundry)	■	Cell phone design
■	Library provider	■	Processor design
■	IDM (system design through silicon manufacturing) consumer, computing, networking	■	Intellectual Property (core processors & peripherals)
■	EDA	■	Automotive

The broad participation in creating the requirements specification ensured the architecture of a comprehensive solution that would be complete in nature. Some primary requirements are:

- **Easy to adopt**—to overcome cost, time and risk deployment issues.
- **Incremental** to existing infrastructure—overlay on top of methods in place.
- **Non-intrusive** to existing practices, methodologies and flows
- **Serves IP/re-use** methodologies with a minimal incremental effort
- **Consolidated** view of the power strategy for a design into a single entity
- **Comprehensive** in capabilities to support the most advanced existing low power design techniques, across the entire continuum of design automation.
- **Extensible** to new low power design techniques and to broader design flow scope (up to system-level and into analog mixed signal in particular).

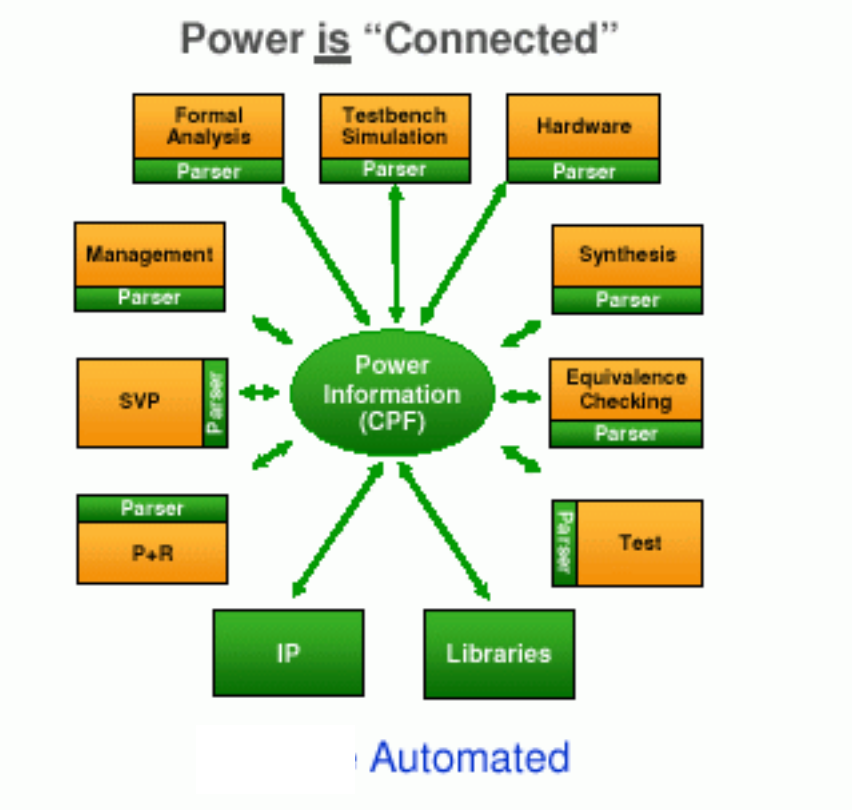
A bottom-up analysis has led to support for a digital RTL to sign-off solution. Although limited in scope, the solution is broad in terms of design automation technology inclusion:

■	RTL/gate simulation	■	Physical synthesis / placement
■	Hardware simulation acceleration	■	Clock tree synthesis
■	Hardware emulation	■	Power grid design
■	Formal analysis	■	Power integrity analysis
■	Design analysis & rule checking	■	Design for Test
■	Formal verification	■	Automatic test pattern generation
■	Synthesis & optimization	■	Constraint generation
■	Floorplanning	■	Constraint verification
■	Silicon virtual prototyping	■	Design project management
■	Power analysis	■	Design IP

Adopting the Common Power Format into standard design flows will have fundamental benefits to those that use it along with industry leading tool solutions. It

- Enables RTL functional verification to validate power related operation
- Guarantees higher design quality with fewer functional failures
- Reduces risk in applying state-of-the-art low power design techniques
- Increases productivity and reduced cost of using those power saving methods

Figure 1-2 Benefit of the Common Power Format on the Design Flow



Format Specifics

The Common Power Format (CPF) is a strictly Tcl-based format.

The CPF file is a power specification file. This implies that the functionality of the design does not change when sourcing a CPF file. The CPF file complements the HDL description of the design and can be used throughout the design creation, design implementation, and design verification flow.

The CPF file contains two categories of objects:

- Design Objects are objects that already exists in the description of the design.
- CPF Objects are objects that are created in the CPF file.

Object Names

Design object names must specify the path to the objects. An object name is specified with respect to the module name specified in the last `set_design` command.

The CPF file for a hierarchical design can contain multiple `set_design` commands.

The first `set_design` command specifies the top module of the design. The top module is at the root of the design hierarchy and is referred to as the *top* design.

Subsequent `set_design` commands must each be preceded by a `set_instance` command. A `set_instance` command specifies the name of a hierarchical instance in the top design. The `set_design` that follows this `set_instance` command specifies the corresponding module name of this instance. This module becomes the *current* design and design objects in the hierarchy of this module can be specified with respect to this current design.

Referencing Design Objects shows how object names are interpreted.

CPF objects are created with a unique name for each type of object.The name cannot contain the hierarchy delimiter character.

See Referencing CPF Objects for more information on referencing CPF objects inside and outside the current scope.

Note: In this document, **scope** refers to either the current design or the top design.

Object names must follow the following naming style:

- Names can contain any sequence of letters, digits, dollar signs ($), and the underscore (_).

Important

If the object name contains a $, you need to escape the dollar sign.

- Object names that contain the escape character must be enclosed in braces ({}).
- Escaped object names must start with the backslash character (\) and end with a white space (blank, tab, newline).

List of Objects

Lists of objects must be enclosed in braces.

You can also specify multiple objects (instances, pins, nets and modules) by including wildcards:

- * matches zero or more characters
- ? matches a single character

Important

Wildcard characters do *not* represent the hierarchical separator.

Hierarchy Delimiter

The default hierarchy delimiter character is the period (.). Other supported characters are

- slash (/)
- caret (^)

The hierarchical delimiter can be specified using the `set_hierarchy_separator` command. See Information Inheritance for more information on the scope sensitivity of this command.

This character only has this special meaning in object names. An escaped hierarchy delimiter character loses its meaning as a hierarchy delimiter.

Bus Delimiters

The default bus delimiters are the square brackets ([ ]). However, because the square brackets represent command substitution in the Tcl language, you need to enclose the bus name in curly braces.

These characters only have this special meaning in object names. When the object name is escaped, the square brackets lose their meaning as bus delimiters.

Range Specification

To specify a range (multiple bits of a bus or of a register array), use the bus delimiters and the colon (:). For example:

```
a[2:7]
b[6:3]
c_reg[4:2]
```

Individual Registers Names

A register or latch instance name is based on

- A base name
- (optional) A suffix appended to the base name

The format of a name in RTL and in the netlist can be different. When you want to use the RTL names in the CPF file, but you are reading a gate-level netlist, you need to specify how the base name and bit information are represented in the netlist.

Specifying the Representation of the Base Name

- To specify the suffix that is appended to the base name of a register or latch instance in the netlist, use the `set_register_naming_style` command.

  The `set_register_naming_style` command expects a string with the following format:

  `string%s`

  The default format is: `_reg%s`

  See Information Inheritance for more information on the scope sensitivity of this command.

The following rules apply:

- An instance name is always started with the base name.
- The suffix is appended to the base name to form the instance name, according to the format specified in the string.
- If the register is an array, `%s` represents the bit information (see also Specifying the Representation of the Bits).

Specifying the Representation of the Bits

- To specify how the bit information of a register or latch instance is represented in the netlist, use the `set_array_naming_style` command.

  The `set_array_naming_style` command expects a string with the following format:

  [*character*]%d[*character*]

  The default format is: `\[%d\]`

  See Information Inheritance for more information on the scope sensitivity of this command.

For example, this option can have values such as:

```
<%d>,\[%d\], _%d_, _%d
```

The following rules apply:

- A suffix is generated for each dimension, according to the format specified in this string.
- The %d represents an index of a certain dimension.

All pieces of the suffix are concatenated, from the highest dimension to the lowest dimension, to form a combined suffix for multi-dimensional arrays.

Expressions

In this document, all expressions refer to Boolean expressions.

The current version only supports the following operators for Boolean expressions:

Operator	Description
!	invert following expression
&	logical AND
|	logical OR

All operators associate left to right. When operators differ in precedence, the operators with higher precedence apply first. In the table above, the operators are shown in order of precedence.

You can use parentheses () to change the operator precedence.

Signal names in expressions cannot represent buses.

Units

To specify the power unit, use the `set_power_unit` command. The default power unit is `mW`.

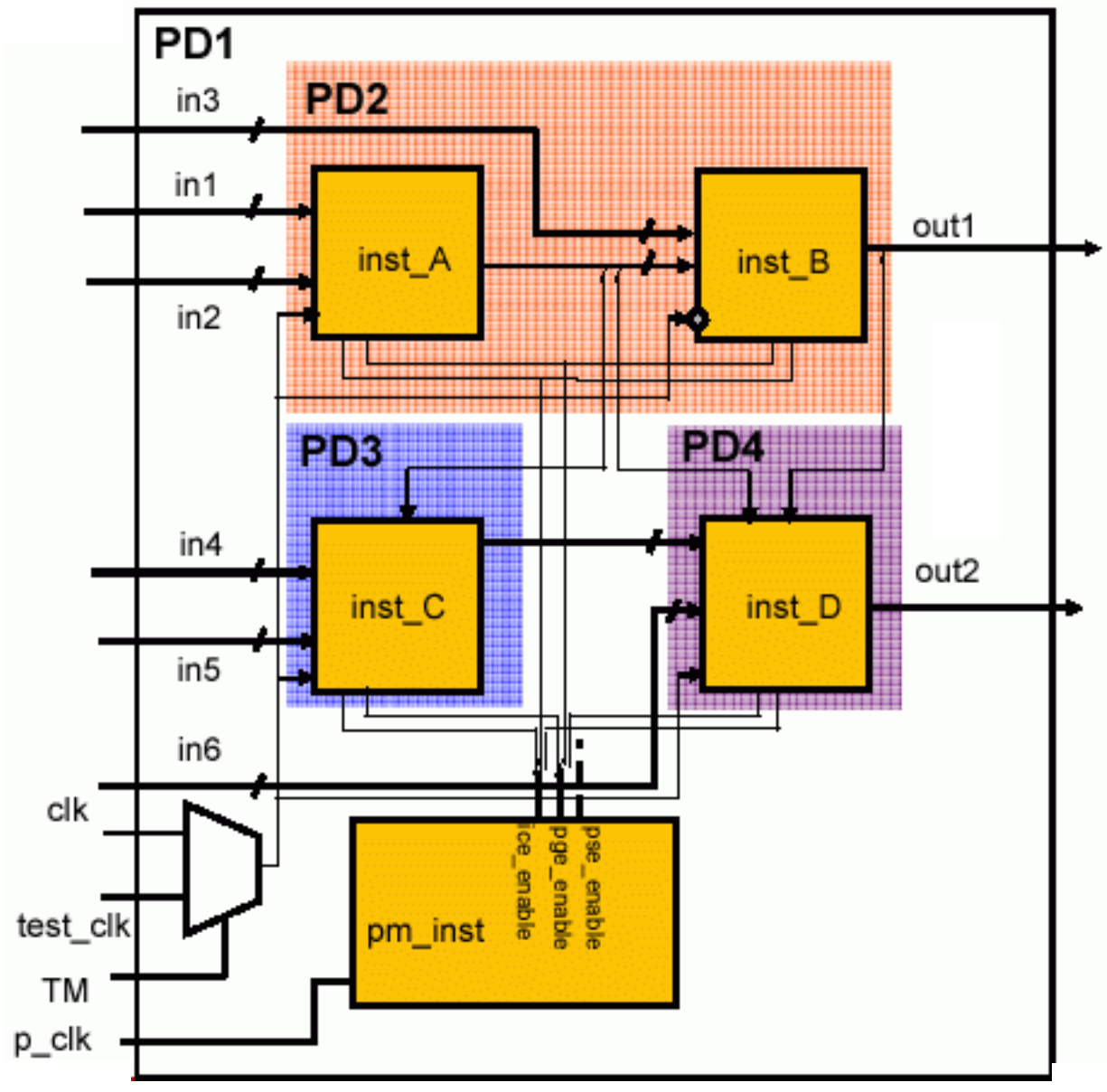
To specify the time unit, use the `set_time_unit` command. The default time unit is `ns`.

All voltage values must be specified in volt (V).

Example

Consider the example design shown in Figure 1-3 on page 14.

Figure 1-3 Example Design for CPF



The design has four domains:

- The top-level of the design and hiearchical instance `pm_inst` belong to the default domain `PD1`

- Hierarchical instances `inst_A` and `inst_B` belong to the power domain `PD2`
- Hierarchical instance `inst_C` belongs to power domain `PD3`
- Hierarchical instance `inst_D` belongs to power domain `PD4`

Table 1-1 on page 15 shows the static behavior (voltage) for each power domain in each of the modes.

Note: A voltage of 0.0V indicates that the power domain is off.

Table 1-1 Static Behavior

Power Mode	Power Domain			
	PD1	PD2	PD3	PD4
PM1	1.2V	1.1V	1.2V	1.0V
PM2	1.2V	0.0V	1.2V	1.0V
PM3	1.2V	0.0V	0.0V	1.0V
PM4	1.0V	0.0V	0.0V	0.0V

The power manager (`pm_inst`) generates three sets of control signals to control each power domain.

Table 1-2 Signals Controlling the Power Domains

Power Domain	Control Signals		
	power switch	isolation cell	state retention cell
PD1	no control signal	no control signal	no control signal
PD2	pse_enable[0]	ice_enable[0]	pge_enable[0]
PD3	pse_enable[1]	ice_enable[1]	pge_enable[1]
PD4	pse_enable[2]	ice_enable[2]	pge_enable[2]

CPF File of Top Design

```
# Define top design
#------------------
set_design top
# Set up logic structure for all power domains
#---------------------------------------------
create_power_domain -name PD1 -default
create_power_domain -name PD2 -instances {inst_A inst_B} \
-shutoff_condition {!pm_inst.pse_enable[0]}
create_power_domain -name PD3 -instances inst_C \
-shutoff_condition {!pm_inst.pse_enable[1]}
create_power_domain -name PD4 -instances inst_D \
-shutoff_condition {!pm_inst.pse_enable[2]}
# Define static behavior of all power domains and specify timing constraints
#---------------------------------------------------------------------------
create_nominal_condition -name high -voltage 1.2
create_nominal_condition -name medium -voltage 1.1
create_nominal_condition -name low -voltage 1.0
create_power_mode -name PM1 -domain_conditions {PD1@high PD2@medium PD3@high \
PD4@low}
create_power_mode -name PM2 -domain_conditions {PD1@high PD3@high PD4@low}
create_power_mode -name PM3 -domain_conditions {PD1@high PD4@low}
create_power_mode -name PM4 -domain_conditions {PD1@low}
# Set up required isolation and state retention rules for all domains
#--------------------------------------------------------------------
create_state_retention_rule -name sr1 -domain PD2 \
-restore_edge {!pm_inst.pge_enable[0]}
create_state_retention_rule -name sr2 -domain PD3 \
-restore_edge {!pm_inst.pge_enable[1]}
create_state_retention_rule -name sr3 -domain PD4 \
-restore_edge {!pm_inst.pge_enable[2]}
create_isolation_rule -name ir1 -from PD2 \
-isolation_condition {pm_inst.ice_enable[0]} -isolation_output high
create_isolation_rule -name ir2 -from PD3 \
-isolation_condition {pm_inst.ice_enable[1]}
create_isolation_rule -name ir3 -from PD4 \
-isolation_condition {pm_inst.ice_enable[2]}
create_level_shifter_rule -name lsr1 -to {PD1 PD3}
end_design
```

2

Terminology

Design Objects

Design objects are objects that are being named in the description of the design which can be in the form of RTL files or a netlist. Design objects can be referenced by the CPF commands.

Design

The top-level module.

Instance

An instantiation of a module or library cell.

- Hierarchical instances are instantiations of modules.
- Leaf instances are instantiations of library cells.

Module

A logic block in the design.

Net

A connection between instance pins and ports.

Pad

An instance of an I/O cell.

Pin

An entry point to or exit point from an instance or library cell.

Port

An entry point to or exit point from the design or a module.

CPF Objects

CPF objects are objects that are being defined (named) in the CPF constraint file. CPF objects can be referenced by the CPF commands.

Analysis View

A view that associates an operating corner with a power mode for which SDC constraints were specified.

The set of active views represent the different design variations (MMMC, that is, multi-mode multi-corner) that will be timed and optimized.

Isolation Rule

Defines the location and type of isolation logic to be added and the condition for when to enable the logic.

Level Shifter Rule

Defines the location and type of level shifter logic to be added.

Library Set

A set (collection) of libraries that was characterized for the same set of operating conditions. By giving the set a name it is easy to reference the set when defining operating corners.

Nominal Operating Condition

A typical operating condition under which the design or blocks perform.

Mode Transition

Defines when the design transitions between the specified power modes.

Operating Corner

A specific set of process, voltage, and temperature values under which the design must be able to perform.

Power Domain

A collection of instances that use the same power supply during normal operation and that can be switched on or off at the same time. You can also associate boundary ports with a power domain to indicate that the drivers for these ports belong to the same power domain.

The only leaf instances allowed are IP blocks and I/O pads.

A power domain can be nested within another power domain.

At the physical level a power domain contains

- A set of (regular) physical gates with a single power and a single ground rail connecting to the same pair of power and ground nets
- The nets driven by these physical gates
- A set of special gates such as level shifter cells, state retention cells, isolation cells, power switches, always-on cells, or multi-rail hard macros (such as, I/Os, memories, and so on) with multiple power and ground rails. At least one pair of the power or ground rails in these special gates or macros must be connecting to the same pair of power and ground nets as the (regular) physical gates connect to.

At the logic level a power domain contains

- A set of logic gates that correspond to the (regular) physical gates of this power domain
- The nets driven by these logic gates
- A set of special gates such as level shifter cells, state retention cells, isolation cells, power switches, always-on cells, or multi-rail hard macros (such as, I/Os, memories, and so on) that correspond to the physical implementation of these gates in this power domain.

At RTL a power domain contains

- The computational elements (operators, process, function and conditional statements) that correspond to the logic gates in this power domain
- The signals that correspond to the nets driven by the corresponding logic gates.

Power Mode

A static state of a design in which each power domain operates on a specific nominal condition.

Power Switch Rule

Defines the location and type of power switches to be added and the condition for when to enable the power switch.

State Retention Rule

Defines the instances to be replaced with state retention flip-flops and the conditions for when to save and restore their states.

Special Library Cells for Power Management

Always On Cell

A special cell located in a switched-off domain, and whose power supply is continuous on even when the power supply for the rest of the logic in the power domain is off.

Isolation Cell

Logic used to isolate signals between two power domains where one is switched on and one is switched off.

The most common usage of such cell is to isolate signals originating in a power domain that is being switched off, from the power domain that receives these signals and that remains switched on.

Level Shifter Cell

Logic to pass data signals between power domains operating at different voltages.

Power Clamp Cell

A special diode cell to clamp a signal to a particular voltage.

Power Switch Cell

Logic used to connect and disconnect the power supply from the gates in a power domain.

State Retention Cell

Special flop or latch used to retain the state of the cell when its main power supply is shut off.

3

CPF File

Command Categories

The following table shows how the CPF commands can be categorized.
.

Category	CPF Command
version command	set_cpf_version
scope commands	set_design
	set_instance
	end_design
general purpose commands	set_array_naming_style
	set_hierarchy_separator
	set_power_unit
	set_register_naming_style
	set_time_unit
design specifications	create_analysis_view
	create_bias_net
	create_global_connection
	create_ground_nets
	create_isolation_rule
	create_level_shifter_rule
	create_mode_transition
	create_nominal_condition
	create_operating_corner
	create_power_domain
	create_power_mode
	create_power_nets
	create_power_switch_rule
	create_state_retention_rule
	define_library_set
	identify_always_on_driver
	identify_power_logic
	set_power_target
	set_switching_activity
	update_isolation_rules
	update_level_shifter_rules
	update_nominal_condition
	update_power_domain

Category	CPF Command
	update_power_mode
	update_power_switch_rule
	update_state_retention_rules
library-related commands	define_always_on_cell
	define_isolation_cell
	define_level_shifter_cell
	define_open_source_input_pin
	define_power_clamp_cell
	define_power_switch_cell
	define_state_retention_cell

Information Precedence

- If you define a CPF object in a specific scope multiple times with the same name, the last definition takes precedence.

  You can add implementation details for CPF objects using multiple update commands as long as each command specifies unique information. If the same information is specified, the information specified in the last command takes precedence.

- If information defined in the CPF file conflicts with information in the referenced library, the information in the CPF file takes precedence.

Information Inheritance

The following commands are scope sensitive:

```
set_array_naming_style
set_cpf_version
set_hierarchy_separator
set_register_naming_style
set_time_unit
set_power_unit
```

By default, the scope inherits the values of the previous scope.

You can change the values for the current scope, but these values only apply as long as you are within the scope.

Referencing Design Objects

When you reference an object by name, the result of the search depends on the format of the object name.

1. `<hierarchy_separator><name>`

   Uses an absolute path to search object `<name>` starting from the root-level hierarchy.

2. `<name>`

   Uses an absolute path to search object <name> in the current design.

   If object <name> is not found in the current design, consider `<hierarchy_separator><name>`.

Referencing CPF Objects

- You can only reference a CPF object that was already created.
- To reference CPF object created *inside* the current scope, you can use the same name.
- To reference a CPF object created *outside* of the current scope, use the hierarchical name of the CPF object. This is the defined name of the CPF object prefixed with the hierarchical name of the scope in which the CPF object is created with respect to the current scope.
- All CPF objects except for the library set are scope sensitive.

Support for Hierarchical CPF

Many design teams can contribute to different blocks in the design. These blocks, whether they are soft blocks or hard blocks (such as IP instances, where the internal details of the block are unknown) can each have their own CPF file.

You can either

- Source these CPF files in the CPF file of the top design
- Use the commands in these CPF files directly in the CPF file of the top design

4

General CPF Commands

create_analysis_view

```
create_analysis_view
     -name string
     -mode mode
     -domain_corners domain_corner_list
```

Creates an analysis view. Associates a list of operating corners with a given mode.

Options and Arguments

`-domain_corners` *`domain_corner_list`*

Specifies the operating corner of the power domain to be considered in the specified mode.

Use the following format to specify a domain corner:

`domain_name@corner_name`

Specify a corner for each domain that you listed when you defined the specified power mode and that is not switched off in that mode.

`-mode` *`mode`*

Specifies a mode.

`-name` *`string`*

Specifies the name of the analysis view.

Note: The specified string cannot contain wildcards.

Note: The specified string cannot contain the hierarchy delimiter character.

create_bias_net

```
create_bias_net
     -net net
     [-driver pin]
     [-user_attributes string_list]
     [-peak_ir_drop_limit float]
     [-average_ir_drop_limit float]
```

Specifies or creates a bias net to be used as a power supply to either forward or backward bias a transistor.

Note: Even if this net exists in the RTL or the netlist, it still must be declared through this command if the net is referenced in other CPF commands.

Options and Arguments

`-average_ir_drop_limit` *`float`*

Specifies the maximum allowed average voltage change on a bias net due to resistive effects in volt (V) for any mode.

Default: `0`

`-driver` *`pin`*

Specifies the driver pin of the net.

`-net` *`net`*

Declares a bias net.

Note: The specified net name cannot contain the hierarchy delimiter character.

`-peak_ir_drop_limit` *`float`*

Specifies the maximum allowed peak voltage change on a bias net due to resistive effects in volt (V) for any mode.

Default: `0`

`-user_attributes` *`string_list`*

Attaches a list of user-defined attributes to the net. Specify a list of strings.

create_global_connection

```
create_global_connection
     -net net
     -pins pin_list
     [-domain domain | -instances instance_list]
```

Specifies how to connect a global net to the specified pins. A global net can be a data net, bias net, power net or ground net.

Given a list of pins, if a specified pin is already connected, that pin is ignored for connection while the remaining pins are connected to the specified global net.

This command allows to specify which pins must be connected. You can

- Specify all pins to be connected with the `-pins` option

  If you omit the -domain or -instances option, the global connection applies to the specified pins of the entire design.

- Combine options to filter the set of pins:
  - Combine `-pins` and `-domain` options—only connects those pins in the specified list that also belong to the specified power domain
  - Combine `-pins` and `-instances` options—only connects those pins in the specified list that also belong to the specified instances.

Options and Arguments

`-instances` *`instance_list`*

Limits the pins to which the specified global net should be connected to pins that belong to the specified instances. Specify the name with respect to the current design or top design.

You can use wildcards (*) to specify a pattern of instance names.

`-net` *`net`*

Specifies the name of the global net for which you specify the global connection.

If the specified net does not exist in the design, you must have defined it with a `create_bias_net`, `create_power_nets` or `create_ground_nets` command.

`-pins` *`pin_list`*	Specifies the name of LEF pin to connect to the specified global net. If several pins of the same instance have names that match the specified names, all those pins will be connected to the specified global net. You can use wildcards (*) to specify the pin names.
`-domain` *`domain`*	Limits the pins to be connected to pins that belong to the specified power domain.

create_ground_nets

```
create_ground_nets
     -nets net_list
     [-voltage string]
     [-internal]
     [-user_attributes string_list]
     [-peak_ir_drop_limit float]
     [-average_ir_drop_limit float]
```

Specifies or creates a list of ground nets.

Note: Even if this net exists in the RTL or the netlist, it still must be declared through this command if the net is referenced in other CPF commands.

The ground nets are created within the current scope.

Options and Arguments

`-average_ir_drop_limit` *`float`*

Specifies the maximum allowed average ground bounce on the specified ground nets due to resistive effects in volt (V) for any mode.

Default: `0`

`-internal` — Specifies that the nets have no connection to any I/O ports or pads.

`-nets` *`net_list`* — Declares a list of ground nets.

Note: The specified net names cannot contain the hierarchy delimiter character.

`-peak_ir_drop_limit` *`float`*

Specifies the maximum allowed peak ground bounce on the specified grounds net due to resistive effects in volt (V) for any mode.

Default: `0`

`-user_attributes` *`string_list`*

Attaches a list of user-defined attributes to the net. Specify a list of strings.

`-voltage` *`string`* — Identifies the voltage applied to the specified nets.

create_isolation_rule

```
create_isolation_rule
     -name string
     -isolation_condition expression
     {-pins pin_list | -from power_domain_list | -to power_domain_list}...
     [-isolation_target {from|to}] [-isolation_output {high|low|hold}]
     [-exclude pin_list]
```

Defines a rule for adding isolation cells.

This command allows to specify which pins must be isolated. You can

- Specify all pins to be isolated with the `-pins` option
- Select only output pins in the power domains listed with the `-from` option
- Select only input pins in the power domains listed with the `-to` option
- Combine options to filter the set of pins:
  - Combine `-pins` and `-from` options—only isolates those pins in the specified list that are also output pins in a power domain listed with the `-from` option
  - Combine `-pins` and `-to` options—only isolates those pins in the specified list that are also input pins in a power domain listed with the `-to` option
  - Combine `-from` and `-to` options—only isolates input pins that belong to a power domain listed with the `-to` option but that are also driven by a net coming from a power domain listed with the `-from` option
  - Combine `-pins`, `-from` and `-to` options—only isolates those input pins in the specified list that belong to a power domain listed with the `-to` option but that are also driven by a net coming from a power domain listed with the `-from` option
  - Exclude specific pins through the `-exclude` option

Options and Arguments

`-exclude` *`pin_list`* Specifies a list of pins that do not require isolation logic.

`-from` *`power_domain_list`*

Limits the pins to be considered for isolation to output pins in the specified power domains.

If specified with `-to` option, all input pins in the `-to` domains that are receiving signals from the `-from` domains will be isolated.

`-isolation_condition` *`expression`*

Specifies the condition when the specified pins should be isolated. The condition is a function of pins.

`-isolation_output {high|low|hold)`

Controls the output value at the output of the isolation gates when the isolation condition is true. The output can be high, low, or held to the value it had right before the isolation condition is activated.

Default: `low`

`-isolation_target {from|to)`

Specifies when this rule applies.

- `from` indicates that the rule applies when the power domain of the *drivers* of the specified pins is switched off.
- `to` indicates that the rule applies when the power domain of the *loads* of the specified pins is switched off.

Default: `from`

Tip

If you intend to use the isolation rule to isolate cells with open source input pins or to isolate power clamp cells, the isolation target must be `to`.

`-name` *`string`*

Specifies the name of the isolation rule.

Note: The specified string cannot contain wildcards.

Note: The specified string cannot contain the hierarchy delimiter character.

`-pins` *`pin_list`*

Specifies a list of pins to be isolated. You can list input pins and output pins of power domains.

You can further limit the pins to be isolated using the `-from`, `-to`, and `-exclude` options.

`-to` *`power_domain_list`*

Limits the pins to be considered for isolation to input pins in the specified power domains.

create_level_shifter_rule

```
create_level_shifter_rule
     -name string
     {-pins pin_list | -from power_domain_list | -to power_domain_list}...
     [-exclude pin_list]
```

Defines a rule for adding level shifters.

This command allows to specify on which pins to insert level shifters. You can

- Specify all pins on which to insert level shifters with the `-pins` option
- Select only output pins in the power domains listed with the `-from` option
- Select only input pins in the power domains listed with the `-to` option
- Combine options to filter the set of pins:
  - Combine `-pins` and `-from` options—only adds level shifters to those pins in the specified list that are also output pins in a power domain listed with the `-from` option
  - Combine `-pins` and `-to` options—only adds level shifters to those pins in the specified list that are also input pins in a power domain listed with the `-to` option
  - Combine `-from` and `-to` options—only adds level shifters to input pins that belong to a power domain listed with the `-to` option but that are also driven by a net coming from a power domain listed with the `-from` option
  - Combine `-pins`, `-from` and `-to` options—only adds level shifters to those input pins in the specified list that belong to a power domain listed with the `-to` option but that are also driven by a net coming from a power domain listed with the `-from` option
  - Exclude specific pins through the `-exclude` option

Options and Arguments

`-exclude` *`pin_list`* Specifies a list of pins that do not require level shifters.

`-from` *`power_domain_list`*

Specifies the name of the originating (driving) power domains.

`-name` *`string`*

Specifies the name of the level shifter rule.

Note: The specified string cannot contain wildcards.

Note: The specified string cannot contain the hierarchy delimiter character.

`-pins` *`pin_list`*

Specifies a list of pins to be isolated. You can list input pins and output pins of power domains.

You can further limit the pins to be isolated using the `-from`, `-to`, and `-exclude` options.

`-to` *`power_domain_list`*

Specifies the names of the destination (receiving) power domains.

create_mode_transition

```
create_mode_transition
     -name string
     -from_mode power_mode -to_mode power_mode
     -start_condition expression [-end_condition expression]
     [-clock_pin clock_pin [-cycles number | -latency float]]
```

Defines how the transition between two power modes is controlled.

Options and Arguments

`-clock_pin` *clock_pin*

Specifies the name of the clock pin that controls the transition.

`-end_condition` *expression*

Specifies the condition that acknowledges when the power mode transition is finished.

`-cycles` *number* — Specifies an integer of number of clock cycles needed to complete the power mode transition.

`-from_mode` *power_mode*

Specifies the power mode from which to transition.

`-latency` *float* — Specifies the time needed to complete the power mode transition. Specify the time in the units specified by the `set_time_unit` command.

`-name` *string* — Specifies the name of the power mode transition.

Note: The specified string cannot contain wildcards.

Note: The specified string cannot contain the hierarchy delimiter character.

`-start_condition` *expression*

Specifies the condition that triggers the power mode transition.

`-to_mode` *power_mode*

Specifies the power mode to which to transition.

create_nominal_condition

```
create_nominal_condition
     -name string
     -voltage float
     [-pmos_bias_voltage float] [-nmos_bias_voltage float]
```

Creates a nominal operating condition with the specified voltage.

Note: A power domain is switched off if it is associated with a nominal condition whose voltage is 0.

Options and Arguments

`-nmos_bias_voltage` *float*

Specifies the bias voltage of the n-type transistors in the domain that uses this condition. The voltage must be specified in volt (V).

`-name` *string*

Specifies the name of the nominal operating condition.

Note: The specified string cannot contain wildcards.

Note: The specified string cannot contain the hierarchy delimiter character.

`-pmos_bias_voltage` *float*

Specifies the bias voltage of the p-type transistors in the domain that uses this condition. The voltage must be specified in volt (V).

`-voltage` *float*

Specifies the voltage of the nominal operating condition in volt (V).

create_operating_corner

```
create_operating_corner
     -name string
     -voltage float
     [-process float]
     [-temperature float]
     -library_set library_set
```

Defines an operating corner and associates it with a library set.

Options and Arguments

`-library_set` *`library_set`*

References the library set to be associated with the specified corner.

`-name` *`corner`*

Specifies the name of the operating corner you want to create.

Note: The specified string cannot contain wildcards.

Note: The specified string cannot contain the hierarchy delimiter character.

`-process` *`float`*

Specifies the process value of the corner. This value depends on the used technology process and is provided by the library vendor.

If this option is not specified, the value defaults to the value specified in the first library of the specified library set.

`-temperature` *`float`*

Specifies the temperature of the operating condition in degrees Celsius.

If this option is not specified, the value defaults to the value specified in the first library of the specified library set.

`-voltage` *`float`*

Specifies the voltage of the operating condition in volt.

create_power_domain

```
create_power_domain
     -name power_domain
     { -default [-instances instance_list] [-boundary_ports pin_list]
     | -instances instance_list [-boundary_ports pin_list]
     | -boundary_ports pin_list }
     [ -shutoff_condition expression ]
     [ -default_restore_edge expression ]
     [ -default_save_edge expression ]
     [ -power_up_states {high|low|random} ]
```

Creates a power domain and specifies the instances and boundary ports and pins that belong to this power domain.

By default, an instance inherits the power domain setting from its parent hierarchical instance or the design, unless that instance was associated with a specific power domain. In addition, all top-level boundary ports are considered to belong to the default power domain, unless they have been associated with a specific domain.

In CPF, power domains are associated with the design objects based on the order of the logical hierarchy. The order in which you create the power domains is irrelevant.

You must define at least one power domain for a design, and one (and only one) power domain must be specified as the default power domain.

The top design, identified by the first `set_design` command, belongs to the default power domain.

Options and Arguments

`-boundary_ports` *`pin_list`*

Specifies the list of inputs and outputs that are considered part of this domain.

- For inputs and outputs of the top-level design, specify ports.
- For inputs and outputs of IP instances, specify a list of the instance pins that are part of the domain.

`-default`

Identifies the specified domain as the default power domain.

All instances of the design that were *not* associated with a specific power domain belong to the default power domain.

`-default_restore_edge` *`expression`*

Specifies the default condition when the states of the sequential elements need to be restored for all state retention rules created for sequential instances in this power domain.

If no state retention rules were created for this power domain, this option is ignored.

The expression is a function of pins. When the expression changes from `false` to `true`, the states are restored.

`-default_save_edge` *`expression`*

Specifies the default condition when the states of the sequential elements need to be saved for all state retention rules created for sequential instances in this power domain.

If no state retention rules were created for this power domain, this option is ignored.

The condition is a function of pins. When the expression changes from `false` to `true`, the states are saved.

`-instances` *`instance_list`*

Specifies the names of all instances that belong to the specified power domain.

If this option is specified together with the `-boundary_ports` option, it indicates that for any connection between a specified port and any instance inside the power domain, no special interface logic for power management is required.

`-name` *`power_domain`* Specifies the name of a power domain.

Note: The specified string cannot contain wildcards.

Note: The specified string cannot contain the hierarchy delimiter character.

`-power_up_states {high|low|random}`

Specifies the state to which the non-state-retention cells in this power domain must be initialized after powering up the power domain.

- `high`: all non state-retention registers are initialized to1 after power-up
- `low`: all non state-retention registers are initialized to 0 after power-up
- `random`: all non state-retention registers are randomly initialized to 0 or 1 after power-up

If this option is omitted, the state to which the non-state-retention cells in this power domain must be initialized is unknown (X).

`-shutoff_condition` *`expression`*

Specifies the condition when a power domain is shut off. The condition is a boolean function of pins.

If this option is omitted, the power domain is considered to be always on.

create_power_mode

```
create_power_mode
     -name string
     -domain_conditions domain_condition_list
     [-default]
```

Defines a power mode.

If your design has more than one power domain, you must define at least one power mode.

If you define any power mode, you must define one (and only one) power mode as the default mode.

Options and Arguments

`-default` — Labels the specified mode as the default mode. The default mode is the mode that corresponds to the initial state of the design.

`-name` *`string`* — Specifies the name of the mode.

Note: The specified string cannot contain wildcards.

Note: The specified string cannot contain the hierarchy delimiter character.

`-domain_conditions` *`domain_condition_list`*

Specifies the nominal condition of each power domain to be considered in the specified power mode.

Use the following format to specify a domain condition:

`domain_name``@`*`nominal_condition_name`*

A domain is considered switched off in the specified mode if

- It is associated with a nominal condition whose voltage is 0
- It is not specified in the list of domain conditions

The voltage of a switched-off domain corresponds to 0.0V.

Note: You can associate each power domain with only one nominal condition for a given power mode.

create_power_nets

```
create_power_nets
     -nets net_list
     [-voltage string]
     [-external_shutoff_condition expression | -internal]
     [-user_attributes string_list]
     [-peak_ir_drop_limit float]
     [-average_ir_drop_limit float]
```

Specifies or creates a list of power nets.

Note: Even if this net exists in the RTL or the netlist, it still must be declared through this command if the net is referenced in other CPF commands.

The power nets are created within the current scope.

Options and Arguments

`-average_ir_drop_limit` *`float`*

Specifies the maximum allowed average IR drop on the specified power nets due to resistive effects in volt (V) for any mode.

Default: `0`

`-external_shutoff_condition` *`expression`*

When the specified power nets are powered by an external power source, you can use an expression to specify under which condition the power source can be switched off.

If this option is not specified, the power source is assumed to be an always-on power supply.

`-internal`

Specifies that the nets have no connection to any I/O ports or pads.

`-nets` *`net_list`*

Declares a list of power nets.

Note: The specified net name cannot contain the hierarchy delimiter character.

`-peak_ir_drop_limit` *`float`*

Specifies the maximum allowed peak IR drop on the specified power nets due to resistive effects in volt (V) for any mode.

Default: `0`

`-user_attributes` *`string_list`*

Attaches a list of user-defined attributes to the net. Specify a list of strings.

`-voltage` *`string`* Identifies the voltage applied to the specified nets.

create_power_switch_rule

```
create_power_switch_rule
     -name string
     -domain power_domain
     {-external_power_net net | -external_ground_net net}
```

Specifies how a single power switch must connect the external and internal power or ground nets for the specified power domain.

You can specify one or more commands for a power domain depending on whether you want to control the switchable power domain by a single switch or multiple switches.

By default, the proper power switch cell will be selected from the cells specified through the `define_power_switch_cell` command. To use a specific cell, use the `update_power_switch_rule` command.

By default, the inversion of the expression specified for the shutoff condition of the power domain is used as the driver for the enable pin of the power switch cell. For complicated cells with multiple enable pins, or if you want to use a different signal to drive the enable pins, use the `update_power_switch_rule` command.

Options and Arguments

`-domain` *`power_domain`*

Specifies the name of a power domain.

`-external_ground_net` *`net`*

Specifies the external ground net to which the source pin of the power switch must be connected. The drain pin must be connected to the internal ground net associated with the specified power domain.

Note: You can only specify this option when you use a footer cell.

`-external_power_net` *`net`*

Specifies the external power net to which the source pin of the power switch must be connected. The drain pin must be connected to the internal power net associated with the specified power domain.

Note: You can only specify this option when you use a header cell.

`-name` *`string`*

Specifies the name of the power switch rule.

Note: The specified string cannot contain wildcards.

Note: The specified string cannot contain the hierarchy delimiter character.

create_state_retention_rule

```
create_state_retention_rule
     -name string
     { -domain power_domain | -instances instance_list }
     [-restore_edge expression [ -save_edge expression ]]
```

Defines the rule for replacing selected registers or all registers in the specified power domain with state retention registers.

Options and Arguments

`-instances` *instance_list*

Specifies the instances that you want to replace with a state retention register.

An instance can be a

- Leaf or hiearchical instance name in a gate-level netlist
- Register variable or hiearchical instance in RTL

If you specify the name of a hierarchical instance, all registers in this instance and its children that belong to the same power domain will be replaced.

Note: The specified instances can belong to several power domains. If they belong to different power domains, the same conditions will be applied to all of them.

`-domain` *power_domain*

Specifies the name of a power domain containing the target registers to be replaced.

All registers in this power domain will be replaced.

`-name` *string*

Specifies the name of the state retention rule.

Note: The specified string cannot contain wildcards.

Note: The specified string cannot contain the hierarchy delimiter character.

`-restore_edge` *`expression`*

Specifies the condition when the states of the registers need to be restored. The expression is a function of pins. When the expression changes from `false` to `true`, the states are restored.

If this option is omitted, but if you specified the `-default_restore_edge` option with the `create_power_domain` command for the corresponding power domain(s), that condition will be used.

If you omit this option and the `-default_restore_edge` option for the corresponding power domain was not specified, the state retention rule will be ignored.

If you specify this option with the `create_state_retention_rule` and the `-default_restore_edge` option with the `create_power_domain` command, the option specified with this command takes precedence.

`-save_edge` *`expression`*

Specifies the condition when the states of the registers need to be saved. The condition is a function of pins. When the expression changes from `false` to `true`, the states are saved.

If this option is omitted, but a `-default_save_edge` option was specified for the corresponding power domain, that condition will be used, otherwise, if both are omitted, the inversion of the expression specified for the `-restore_edge` option will be used.

If you specify this option with the `create_state_retention_rule` and the `-default_save_edge` option with the `create_power_domain` command, the option specified with this command takes precedence.

define_library_set

```
define_library_set
     -name library_set
     -libraries library_list
```

Creates a library set.

Options and Arguments

`-libraries` *`library_list`*

Specifies a list of library files (.lib files).

`-name` *`library_set`* Specifies the name of a library set.

Note: The specified string cannot contain wildcards.

end_design

`end_design`

Used with a `set_design` command groups a number of CPF commands that apply to the current design or top design.

identify_always_on_driver

```
identify_always_on_driver
     -pins pin_list [-no_propagation]
```

Specifies a list of driving pins in the design that are considered as always on-drivers.

A net connected to an always-on driver is not switched off (even if its parent power domain is switched off) as long as the parent of the driver is not switched off.

For any always-on driver, none of the logical nets physically connected to this driver are switched off as long as the parent of the driver is not switched off.

If the specified pin is a driving pin of a multiple driven net, the net is considered to be always on.

Note: Outputs of cells that are always on, are always-on drivers.

Options and Arguments

`-no_propagation`	Considers only the logical net directly connected to each driving pin to be not switched off.
`-pins` *`pin_list`*	Specifies the names of the driving pins. Specify the full hierarchical path of the pin. **Note:** A bidirectional pin can also be considered as a driving pin.

identify_power_logic

```
identify_power_logic
     -type isolation
     -instances instance_list
```

Identifies any isolation logic instantiated in RTL or the gate-level netlist that is implemented through regular cells that do not have the required Liberty attributes and are not defined through the `define_isolation_cell` command.

Note: Any instances of special low power cells (such as level shifter cells, isolation cells, and so on) instantiated in RTL or the gate-level netlist that have the required Liberty attributes or that are defined through any of the library cell-related CPF commands are automatically identified.

Options and Arguments

`-instances` *`instance_list`*

Specifies the names of all instances of the power logic selected through the `-type` option.

`-type`

Specifies the type of power logic to be identified.

Currently, the only valid option is `isolation`.

set_array_naming_style

```
set_array_naming_style
     [string]
```

Specifies the format used to name the design objects in the netlist starting from multi-bit arrays in the RTL description. For sequential elements, the bit information is appended to the instance name which is determined by the `set_register_naming_style` command.

The command returns the new setting or the current setting in case the command was specified without an argument.

Options and Arguments

string	Specifies the format of the bit information. The string must have the following format: [*character*]%d[*character*] You can use angle brackets, square brackets, or underscores. *Default:* `\[%d\]`

set_cpf_version

```
set_cpf_version
     [value]
```

Specifies the version of the format.

The command returns the new setting or the current setting in case the command was specified without an argument.

If specified, this command must be the first CPF command in a CPF file.

Options and Arguments

value	Specifies the version. Use a string. *Default*: `1.0`

set_design

```
set_design
     module [-ports port_list]
```

Specifies the name of the module to which the power information in the CPF file applies.

Note: If this command appears multiple times, the first one applies to the top design, while the next ones must follow a scope change using the `set_instance` command.

Options and Arguments

module	Specifies the name of the module to which the power information in the current CPF file applies.
-ports *port_list*	Specifies a list of *virtual* ports in the specified module. Virtual ports do not exist in the RTL of this module but will be needed for the control signals of the low power logic such as isolation logic, state-retention logic, and so on.

set_hierarchy_separator

```
set_hierarchy_separator
     [character]
```

Specifies the hierarchy delimiter character used in the CPF file.

The command returns the new setting or the current setting in case the command was specified without an argument.

Options and Arguments

character	Specifies the hierarchy delimiter character. *Default:* .

set_instance

```
set_instance
     [hier_instance [-merge_default_domains]
     [-port_mapping port_mapping_list]]
```

Changes the scope to the specified hierarchical instance.

The command returns the current scope in case the command was specified without an argument. If the current scope is the top design, the hierarchy separator is returned.

If the command is specified with any argument, it must be followed by a `set_design` command. The hierarchical instance specified with the `set_instance` command must be an instantiation of the module name specified with the `set_design` command.

The scope is used for naming resolution and affects

- All design objects
- All the expressions in the CPF design-related constraints

All CPF objects referred to in the library cell-related CPF commands are scope *insensitive*.

Important

Any rule created in the module-level CPF file that references a virtual port that is declared using the `-ports` option of the `set_design` command, but whose mapping is not specified through the `-port_mapping` option, will be ignored.

Options and Arguments

`hier_instance` — Changes the scope to the specified hierarchical instance. The instance must be a valid hierarchical instance in the current scope.

`-merge_default_domains`

Specifies whether to merge the default power domain of the current design (scope) with the default power domain of the parent scope.

Note: You can only merge two default domains if they are both always on.

`-port_mapping` *`port_mapping_list`*

Specifies the mapping of the virtual ports specified in the `set_design` command to the parent-level drivers.

Use the following format to specify a port mapping:

{*`virtual_port parent_level_driver`*}

set_power_target

```
set_power_target
     { -leakage float | -dynamic float
     | -leakage float -dynamic float}
```

Specifies the targets for the average leakage and dynamic power of the current design across all the power modes. All power targets must be specified in the units specified by the `set_power_unit` command.

Options and Arguments

`-dynamic` *`float`*	Specifies the target for the average dynamic power. *Default*: 0 `mW`
`-leakage` *`float`*	Specifies the target for the average leakage power.

set_power_unit

```
set_power_unit
     [pW|nW|uW|mW|W]
```

Specifies the unit for all power values in the CPF file.

The command returns the new setting or the current setting in case the command was specified without an argument.

Options and Arguments

`[pW|nW|uW|mW|W]`	Specifies the power unit. You can specify any of these five values. *Default*: `mW`

set_register_naming_style

```
set_register_naming_style
     [string%s]
```

Specifies the format used to name flip-flops and latches in the netlist starting from the register names in the RTL description.

The command returns the new setting or the current setting in case the command was specified without an argument.

Options and Arguments

string	Specifies the suffix to be appended to the base name of a register. The %s represents the bit information. *Default*: `_reg%s`

set_switching_activity

```
set_switching_activity
     { {-all | -pins pin_list | -instances instance_list [-hierarchical]}
        -probability float -toggle_rate float }
     | [-clock_pins pin_list] -toggle_percentage float }
     [-mode mode]
```

Specifies activity values (toggle rate and probability) for the specified pins.

The toggle rate is the average number of toggle counts per time unit of a net during a given simulation time.

The probability is the probability of a net being high during a given simulation time.

Options and Arguments

`-all`	Indicates to apply the specified activity values to all pins.
`-clock_pins` *`pin_list`*	Indicates to apply the specified activity values only to data signals associated with the specified clock pins.
`-hierarchical`	Indicates to traverse the hierarchy of all specified hierarchical instances to apply the specified activity values to the outputs of all leaf instances in the hierarchy.
`-instances` *`instance_list`*	Indicates to apply the specified activity values to all outputs if the specified instances are non-hierarchical instances. For hierarchical instances, it indicates to apply the specified activity values to the outputs of the leaf instances in the specified hierarchical instances (without traversing the hierarchy).
`-mode` *`mode`*	Specifies the mode to which these values apply. If this option is not specified, the specified value applies to all modes for which no specific values were specified.
`-pins` *`pin_list`*	Indicates to apply the specified activity values to the specified pins.

`-probability` *`float`*

Specifies the probability value.

The probability is a floating value between `0` and `1`.

`-toggle_percentage` *`float`*

Specifies to compute the toggle rate as the multiplication of the specified value and the toggle rate of the related clock. If multiple clocks are related to the specified data pin, the clock with the worst frequency is used. If no clock is related to the data pin, the worst clock of the design is used.

The value must be a float between 0 and 100.

If you specify clock pins through the `-clock_pins` option, the computed toggle rate is only applied to the data pins related to those clock pins.

If you did not specify any clock pins, a computed toggle rate is applied to all data pins and the value for each data pin will be based on its related clock pin.

`-toggle_rate` *`float`*

Specifies the number of toggles per time unit.

set_time_unit

```
set_time_unit
     [ns|us|ms]
```

Specifies the unit for all time values in the CPF file.

The command returns the new setting or the current setting in case the command was specified without an argument.

Options and Arguments

`[ns|us|ms]`	Specifies the time unit. You can specify any of these three values. *Default*: `ns`

update_isolation_rules

```
update_isolation_rules -names rule_list
     { -location {from | to}
     | -cells cell_list -library_set library_set
     | -prefix string
     | -combine_level_shifting
     | -open_source_pins_only}...
```

Appends the specified isolation rules with implementation information.

Note: You must specify at least one of the options besides `-name`, but you can also combine several options.

Options and Arguments

`-cells` *`cell_list`*
: Specifies the names of the library cells that must be used as isolation cells for the selected pins.

  By default, the appropriate isolation cells are chosen from the isolation cells defined with the `define_isolation_cell` command or from the library cells with isolation related Liberty attributes.

`-combine_level_shifting`
: Specifies to apply the specified rules when level shifter logic is needed.

  Whenever isolation logic must be inserted for a given pin and a level shifter is also required for that pin, the tool must use a cell that has both functions.

`-library_set` *`library_set`*
: References the library set to be used to search for the specified cells. Specify the library set name. All matching cells will be used.

`-location {from|to}`

Specifies the power domain to which the isolation logic must be added.

- `from` stores the isolation logic with the instances of the originating power domain
- `to` stores the isolation logic with the instances of the destination power domain

Default: `to`

`-names` *`rule_list`* Specifies the names of the rules to be updated.

The name can contain wildcards.

`-open_source_pins_only`

Limits the pins to be isolated to the open source pins that belong to a power domain that is switched off while the driver domain remains powered on.

This implies that only those rules that were created with the `-isolation_target` option set to `to` can be updated.

`-prefix` *`string`* Specifies the prefix to be used when creating the isolation logic.

Default: `CPF_ISO_`

update_level_shifter_rules

```
update_level_shifter_rules
     -names rule_list
     { -location {from | to}
     | -cells cell_list -library_set library_set
     | -prefix string }...
```

Appends the specified level shifter rule with implementation information.

Note: You must specify at least one of the options besides `-name`, but you can also combine several options.

Options and Arguments

`-cells` *`cell_list`*
: Specifies the names of the library cells to be used to bridge the specified power domains.

  By default, the appropriate level shifter cells are chosen from the level shifter cells defined with the `define_level_shifter_cell` command or from the library cells with level-shifter related Liberty attributes.

`-library_set` *`library_set`*
: References the library set to be used to search for the specified cells. Specify the library set name. All matching cells will be used.

`-location {from|to}`
: Specifies where the level shifters must be stored:

  - `from` stores the level shifters with the instances of the originating power domain
  - `to` stores the level shifters with the instances of the destination power domain

  Default: `to`

`-names` *`rule_list`*
: Specifies the names of the level shifter rules to be updated.

  The name can contain wildcards.

`-prefix` *`string`*
: Specifies the prefix to be used when creating this logic.

  Default: `CPF_LS_`

update_nominal_condition

```
update_nominal_condition
     -name condition
     -library_set library_set
```

Associates a library set with the specified nominal operating condition.

Options and Arguments

`-library_set` *`library_set`*

References the library set to be associated with the specified condition.

`-name` *`condition`* Specifies the name of the nominal operating condition.

Note: The specified string cannot contain wildcards.

update_power_domain

```
update_power_domain
     -name domain
     { -internal_power_net net | -internal_ground_net net
     | -min_power_up_time float | -max_power_up_time float
     | -pmos_bias_net net | -nmos_bias_net net | -user_attributes string_list
     | -rail_mapping rail_mapping_list -library_set library_set} ...
```

Specifies implementation aspects of the specified power domain.

Note: You must specify at least one of the options besides `-name`, but you can also combine several options.

Options and Arguments

`-internal_ground_net` *net*

Specifies the main ground net for all functional gates in the specified power domain.

This option is required when you use footer cells.

`-internal_power_net` *net*

Specifies the main power net for all functional gates in the specified power domain.

This option is required when you use header cells.

`-library_set` *library_set*

References the library set to be used to search for the specified power rails. Specify the library set name.

`-min_power_up_time (max_power_up_time)` *float*

Specifies the minimum (maximum) time allowed for the power domain to ramp up. Specify the time in the units defined by `set_time_unit`.

Default: `0`

`-nmos_bias_net` *net*

Specifies the net to be used to bias the n-type transistors of all functional gates in this power domain.

`-name` *domain* Specifies the name of the power domain.

`-pmos_bias_net` *net*

Specifies the net to be used to bias the p-type transistors of all functional gates in this power domain.

`-rail_mapping` *rail_mapping_list*

Specifies the mapping of the power rails specified in the library (.lib files) to the power and ground nets defined in the CPF file.

Use the following format to specify a rail mapping:

{*power_rail_name power_or_ground_net*}

`-user_attributes` *string_list*

Attaches a list of user-defined attributes to the domain. Specify a list of strings.

update_power_mode

```
update_power_mode
     -name mode
     { -activity_file file -activity_file_weight weight
     | -sdc_files sdc_file_list
     | -peak_ir_drop_limit domain_voltage_list
     | -average_ir_drop_limit domain_voltage_list
     | -leakage_power_limit float
     | -dynamic_power_limit float}...
```

Specifies the constraints for the power mode.

Note: You must specify at least one of the options besides `-name`, but you can also combine several options.

Options and Arguments

`-activity_file` *`file`*

Specifies the path to the activity file. Supported formats for the activity files are VCD, TCF, and SAIF.

`-activity_file_weight` *`weight`*

Specifies the relative weight of the activities in this file in percentage. Use a positive floating number between 0 and 100.

To estimate the total average chip power over all modes, the activity weights are used to adjust the relative weight of each power mode.

Note: If the weights specified for the activity files for the different power modes do not add up to 100, an adjusted weight is used.

`-average_ir_drop_limit` *`domain_voltage_list`*

Specifies the maximum allowed average voltage change on a power net due to resistive effects in volt (V) for the specified power mode. This net must be the internal power net of the power domain to be considered in the specified mode.

Use the following format to specify the maximum allowed average voltage change in the domain:

`domain_name@voltage`

Use a floating value for the voltage.

If a domain is omitted from this list, the value for the internal power net of this domain will be 0, unless you specified a value using the `-average_ir_drop_limit` option of the `create_power_nets` command.

`-dynamic_power_limit` *`float`*

Specifies the maximum allowed average dynamic power in the specified mode.

Default: 0 `mW`

`-leakage_power_limit` *`float`*

Specifies the maximum allowed average leakage power in the specified mode.

Default: 0 `mW`

`-name` *`mode`*

Specifies the name of the mode.

`-peak_ir_drop_limit` *`domain_voltage_list`*

Specifies the maximum allowed peak voltage change on a power net due to resistive effects in volt (V) for the specified power mode. This net must be the internal power net of the power domain to be considered in the specified mode.

Use the following format to specify the maximum allowed peak voltage change in the domain:

`domain_name@voltage`

Use a floating value for the voltage.

If a domain is omitted from this list, the value for the internal power net of this domain will be 0, unless you specified a value using the `-average_ir_drop_limit` option of the `create_power_nets` command.

`-sdc_files` *`sdc_file_list`*

Specifies a list of SDC files to be used for the specified mode.

update_power_switch_rule

```
update_power_switch_rule
     -name string
     { -enable_condition_1 expression [-enable_condition_2 expression]
     | -acknowledge_receiver pin
     | -cells cell_list -library_set library_set
     | -prefix string
     | -peak_ir_drop_limit float
     | -average_ir_drop_limit float }...
```

Appends the specified rules for power switch logic with implementation information.

Options and Arguments

`-acknowledge_receiver` *`pin`*

Specifies an input pin in the design which must be connected to an output pin of the power switch cell.

`-average_ir_drop_limit` *`float`*

Specifies the maximum allowed average voltage change across a power switch due to resistive effects in volt (V).

Default: `0`

`-cells` *`cell_list`*

Specifies the name of the library cells that can be used as power switch cells.

`-enable_condition_1 (-enable_condition_2)` *`expression`*

Specifies the condition when the power switch should be enabled. The condition is a Boolean expression of one or more pins.

If only `-enable_condition_1` is specified, the expression is used as the enable signal for all enable pins of the power switch cell.

If both options are specified, the expression of the `-enable_condition_1,-enable_condition_1` will be used respectively as enable signal for the enable pin of stage 1 and stage 2 of the power switch cell.

Note: If the specified power domain has a shutoff condition, the support set of this expression must be a subset of the support set of the shut-off condition.

Default: the inversion of the expression specified for the shutoff condition of the power domain is used as the enable signal driver for the enable pin(s) of the power switch cell

`-library_set` *`library_set`*

References the library set to be used to search for the specified cells. Specify the library set name. All matching cells will be used.

`-name` *`string`* Specifies the name of the power switch rule.

`-peak_ir_drop_limit` *`float`*

Specifies the maximum allowed peak voltage change across a power switch due to resistive effects in volt (V).

Default: `0`

`-prefix` *`string`* Specifies the prefix to be used when creating this logic.

Default: `CPF_PS_`

update_state_retention_rules

```
update_state_retention_rules
     -names rule_list
     {-cell_type string | -cell libcell}
     -library_set library_set
```

Appends the specified rules for state retention logic with implementation information.

By default, the appropriate state retention cells are chosen from the state retention cells defined with the `define_state_retention_cell` command or from the library based on the appropriate Liberty attributes.

Options and Arguments

`-cell` *`libcell`*
: Specifies the library cell to be used to map the flops.

`-cell_type` *`string`*
: Specifies the class of library cells that can be used to map the flops.

`-library_set` *`library_set`*
: References the library set to be used to search for the specified cell or specified cell type. Specify the library set name.

`-names` *`rule_list`*
: Specifies the names of the rules to be updated.

  The name can contain wildcards.

5

Library Cell-Related CPF Commands

define_always_on_cell

```
define_always_on_cell
     -cells cell_list [-library_set library_set]
     [ [-power_switchable LEF_power_pin | -ground_switchable LEF_ground_pin]
       -power LEF_power_pin  -ground LEF_ground_pin ]
```

Identifies the library cells in the .lib files that can be used as cells that are always on.

Note: Outputs of cells that are always on, are always-on drivers.

Options and Arguments

`-cells` *`cell_list`*

Identifies the specified cells as special cells that are always on.

`-ground` *`LEF_ground_pin`*

If this option is specified with the `-power_switchable` option, it indicates the `GROUND` pin of the specified cell.

If this option is specified with the `-ground_switchable` option, it indicates the `GROUND` pin in the corresponding LEF cell to which the ground that is on during power shut-off mode is applied.

`-ground_switchable` *`LEF_power_pin`*

Identifies the `GROUND` pin in the corresponding LEF cell to which the ground that is switched off during power shut-off mode is applied.

`-library_set` *`library_set`*

References the library set to be used to search for the specified cells. Specify the library set name. All matching cells will be used.

If you omit this option, all library sets are searched and all matching cells will be used.

`-power` *`LEF_power_pin`*

If this option is specified with the `-ground_switchable` option, it indicates the `POWER` pin of the specified cell.

If this option is specified with the `-power_switchable` option, it indicates the `POWER` pin in the corresponding LEF cell to which the power that is on during power shut-off mode is applied.

`-power_switchable` *`LEF_power_pin`*

Identifies the `POWER` pin in the corresponding LEF cell to which the power that is switched off during power shut-off mode is applied.

define_isolation_cell

```
define_isolation_cell
     -cells cell_list [-library_set library_set]
     [-always_on_pin pin_list]
     [ {-power_switchable LEF_power_pin | -ground_switchable LEF_ground_pin}
       -power LEF_power_pin  -ground LEF_ground_pin ]
     [-valid_location { from | to}]
     [-non_dedicated]
     -enable pin
```

Identifies the library cells in the .lib files that *can* be used as isolation cells.

Options and Arguments

`-always_on_pin` *`pin_list`*

Specifies a list of cell pins which must always be driven.

Note: A pin specified with this option, can be specified with other options as well.

`-cells` *`cell_list`*

Identifies the specified cells as isolation cells.

The libraries loaded will be searched and all cells found will be identified.

`-enable` *`pin`* Identifies the specified cell pin as the enable pin.

This pin must be an always-on pin.

`-ground` *`LEF_ground_pin`*

If this option is specified with the `-power_switchable` option, it indicates the `GROUND` pin of the specified cell.

If this option is specified with the `-ground_switchable` option, it indicates the `GROUND` pin in the corresponding LEF cell to which the ground that is on during power shut-off mode is applied.

`-ground_switchable` *`LEF_power_pin`*

Identifies the `GROUND` pin in the corresponding LEF cell to which the ground that is turned off during power shut-off mode is applied.

`-library_set` *`library_set`*

References the library set to be used to search for the specified cells. Specify the library set name. All matching cells will be used.

If you omit this option, all library sets are searched and all matching cells will be used.

`-non_dedicated` Allows to use specified cells as normal function cells.

`-power` *`LEF_power_pin`*

If this option is specified with the `-ground_switchable` option, it indicates the `POWER` pin of the specified cell.

If this option is specified with the `-power_switchable` option, it indicates the `POWER` pin in the corresponding LEF cell to which the power that is on during power shut-off mode is applied.

`-power_switchable` *`LEF_power_pin`*

Identifies the `POWER` pin in the corresponding LEF cell to which the power that is turned off during power shut-off mode is applied.

`-valid_location {from | to}`

Specifies the location of the isolation cell. Possible values are

- `from`—indicating that the cell must be stored with the source power domain
- `to`—indicating that the cell must be stored with the destination power domain

Default: `to`

define_level_shifter_cell

```
define_level_shifter_cell
     -cells cell_list [-library_set library_set]
     [-always_on_pin pin_list]
     -input_voltage_range {voltage | voltage_range}
     -output_voltage_range {voltage | voltage_range}
     [-direction {up|down|bidir}]
     [-output_voltage_input_pin pin]
     { -input_power_pin LEF_power_pin [-output_power_pin LEF_power_pin]
     | [-input_power_pin LEF_power_pin] -output_power_pin LEF_power_pin }
     -ground LEF_ground_pin
     [-valid_location { from | to}
```

Identifies the library cells in the .lib files that can be used as level shifter cells.

Options and Arguments

`-always_on_pin` *`pin_list`*

Specifies a list of cell pins which must always be driven.

Note: A pin specified with this option, can be specified with other options as well.

`-cells` *`cell_list`* Identifies the specified cell as a level shifter.

Note:

The libraries loaded will be searched and all cells found will be used.

`-direction {up | down | bidir}`

Specifies whether the level shifter can be used between a lower and higher voltage, or vice versa.

Default: `up`

`-ground` *`LEF_ground_pin`*

Identifies the name of the `GROUND` pin in the corresponding LEF cell.

`-input_power_pin` *`LEF_power_pin`*

Identifies the name of the `POWER` pin in the corresponding LEF cell that must be connected to the power net to which the voltage of the source power domain is applied.

`-input_voltage_range {`*`voltage | voltage_range`*`}`

Identifies either a single input voltage or a range for the input (source) voltage that can be handled by this level shifter.

The voltage range must be specified as follows:

`lower_bound:upper_bound:step`

Specify the lower bound, upper bound and voltage increment step, respectively.

`-library_set` *`library_set`*

References the library set to be used to search for the specified cells. Specify the library set name. All matching cells will be used.

If you omit this option, all library sets are searched and all matching cells will be used.

`-output_power_pin` *`LEF_power_pin`*

Identifies the name of the `POWER` pin in the corresponding LEF cell that must be connected to the power net to which the voltage of the destination power domain is applied.

`-output_voltage_input_pin` *`pin`*

Identifies the input pin that drives a gate inside the level shifter cell that is powered by the power supply connected to the pin identified by the `-output_power_pin` option.

By default, the gates (inside the level shifter cell) driven by the input pins, are assumed to be powered by the power supply connected to the pin identified by the `-input_power_pin` option.

Note: If the cell is also listed in the `define_isolation_cell` command, this pin is the enable pin of the isolation cell.

`-output_voltage_range {`*`voltage | voltage_range`*`}`

Identifies either a single output voltage or a range for the output (source) voltage that can be handled by this level shifter.

The voltage range must be specified as follows:

`lower_bound:upper_bound:step`

`-valid_location {from | to}`

Specifies the location of the level shifter cell. Possible values are

- `from`—indicating that the cell must be stored with the source power domain
- `to`—indicating that the cell must be stored with the destination power domain

Default: `to`

define_open_source_input_pin

```
define_open_source_input_pin
     -cells cell_list -pin pin_name
     [-library_set library_set]
```

Specifies a list of cells that contain open source input pins.

input pins that must be isolated when the power supply of the driver is on, but the power supply of the cells to which the input pin belongs is shut off.

Options and Arguments

`-cells` *`cell_list`*

Specifies the cells to which the open source input pins belong.

`-library_set` *`library_set`*

References the library set to be used to search for the specified cells. Specify the library set name. All matching cells will be used.

If you omit this option, all library sets are searched and all matching cells will be used.

`-pin` *`pin_name`* Specifies the name of the open source input pin.

define_power_clamp_cell

```
define_power_clamp_cell
     -cells cell_list
     -data pin_name
     -power pin_name [-ground pin_name]
     [-library_set library_set]
```

Specifies a list of diode cells used for power clamp control.

Options and Arguments

`-cells` *`cell_list`*

Identifies the specified cells as power clamp diode cells.

`-data` *`pin_name`* Specifies the cell pin that connects to the data signal.

`-ground` *`pin_name`* Specifies the cell pin that connects to the ground net.

`-library_set` *`library_set`*

References the library set to be used to search for the specified cells. Specify the library set name. All matching cells will be used.

If you omit this option, all library sets are searched and all matching cells will be used.

`-power` *`pin_name`* Specifies the cell pin that connects to the power net.

define_power_switch_cell

```
define_power_switch_cell
     -cells cell_list [-library_set library_set]
     -stage_1_enable expression [-stage_1_output expression]
     [-stage_2_enable expression [-stage_2_output expression]]
     -type {footer|header}
     [ -power_switchable LEF_power_pin  -power LEF_power_pin
     | -ground_switchable LEF_ground_pin -ground LEF_ground_pin ]
     [ -on_resistance float]
     [ -stage_1_saturation_current float] [ -stage_2_saturation_current float]
     [ -leakage_current float ]
```

Identifies the library cells in the .lib files that can be used as power switch cells.

Options and Arguments

`-cells` *`cell_list`*

Identifies the specified cells as power switch cells.

`-ground` *`LEF_ground_pin`*

Identifies the input ground pin of the corresponding LEF cell.

`-ground_switchable` *`LEF_ground_pin`*

Identifies the output ground pin in the corresponding LEF cell that must be connected to a switchable ground net.

`-leakage_current` *`float`*

Specifies the leakage current when the power switch is turned off. Specify the current in ampere (A).

`-library_set` *`library_set`*

References the library set to be used to search for the specified cells. Specify the library set name. All matching cells will be used.

If you omit this option, all library sets are searched and all matching cells will be used.

`-on_resistance` *`float`*

Specifies the resistance of the power switch when the power switch is turned on. Specify the resistance in ohm.

`-power` *`LEF_power_pin`*

Identifies the input `POWER` pin of the corresponding LEF cell.

`-power_switchable` *`LEF_power_pin`*

Identifies the output power pin in the corresponding LEF cell that must be connected to a switchable power net.

`-stage_1_saturation_current` (`-stage_2_saturation_current`) *`float`*

Specifies the Id saturation current of the MOS transistor in the specified stage. Specify the current in ampere (A).

The saturation current—which can be found in the SPICE model—limits the maximum current that a power switch can support.

`-stage_1_enable (-stage_2_enable)` *`expression`*

Specifies when the transistor driven by this input pin is turned on (enabled) or off.

If only stage 1 is specified, the switch is turned on when the expression for the `-stage_1_enable` option evaluates to `true`.

If both stages are specified, the switch is turned on when the expression for both enable options evaluates to true.

The expression is a function of the input pin. This pin must be an always-on pin.

`-stage_1_output (-stage_2_output)` *`expression`*

Specifies whether the output pin specified in the expression is the buffered or inverted output of the input pin specified through the corresponding -`stage_`*`x`*`_enable` option.

The pin specified through the -`acknowledge_receiver` option of the `create_power_switch_rule` command is connected to the output pin specified through

- The `-stage_1_output` option if the `-stage_2_output` option is omitted.
- The -`stage_2_output` option if both -`stage_1_output` and `stage_2_output` options are specified.

Note: If neither option is specified, the pin specified through the `-acknowledge_receiver` is left unconnected.

`-type {header|footer}`

Specifies whether the power switch cell is a header or footer cell.

define_state_retention_cell

```
define_state_retention_cell
     -cells cell_list [-library_set library_set]
     [-always_on_pin pin_list]
     [-clock_pin pin]
     -restore_function expression [-restore_check expression]
     [-save_function expression] [-save_check expression]
     [ {-power_switchable LEF_power_pin | -ground_switchable LEF_ground_pin}
       -power LEF_power_pin  -ground LEF_ground_pin ]
```

Identifies the library cells in the .lib files that can be used as state retention cells.

Options and Arguments

`-always_on_pin` *`pin_list`*

Specifies a list of cell pins which must always be driven.

Note: A pin specified with this option, can be specified with other options as well.

`-cells` *`cell_list`*

Identifies the specified cells as state retention cells.

The libraries loaded will be searched and all cells found will be used.

`-clock_pin` *`pin`* Specifies the clock pin.

`-ground` *`LEF_ground_pin`*

If this option is specified with the `-power_switchable` option, it specifies the `GROUND` pin of the corresponding LEF cell.

If this option is specified with the `-ground_switchable` option, it indicates the `GROUND` pin in the corresponding LEF cell to which the ground net that is on during power shut-off mode is connected.

`-ground_switchable` *`LEF_power_pin`*

Identifies the `GROUND` pin in the corresponding LEF cell to which the ground that is turned off during power shut-off mode is applied.

`-library_set` *`library_set`*

References the library set to be used to search for the specified cells. Specify the library set name. All matching cells will be used.

If you omit this option, all library sets are searched and all matching cells will be used.

`-power` *`LEF_power_pin`*

If this option is specified with the `-ground_switchable` option, it indicates the `POWER` pin of the specified cell.

If this option is specified with the `-power_switchable` option, it indicates the `POWER` pin to which the power that is always on during shut-off mode is applied.

`-power_switchable` *`LEF_power_pin`*

Identifies the `POWER` pin in the corresponding LEF cell to which the power that is turned off during power shut-off mode is applied.

`-restore_check` *`expression`*

Specifies the additional condition when the states of the sequential elements can be restored. The expression can be a function of the clock pin and the restore pin. The expression must be true when the restore event occurs.

Note: If you want to use the clock pin in the expression, you must have identified the clock pin with the `-clock_pin` option.

`-restore_function` *`expression`*

Specifies the polarity of the restore pin that enables the retention cell to restore the saved value after exiting power shut-off mode.The restore pin must be an always-on pin.

Note: Expression is limited to the pin name and the inversion of the pin name. An expression containing only the pin name indicates an active high polarity. An expression containing the inversion of the pin name indicates an active low polarity.

`-save_check` *`expression`*

Specifies the additional condition when the states of the sequential elements can be saved. The expression can be a function of the clock pin and the save pin The expression must be `true` when the save event occurs.

Note: If you want to use the clock pin in the expression, you must have identified the clock pin with the `-clock_pin` option.

`-save_function` *`expression`*

Specifies the polarity of the save pin that enables the retention cell to save the current value before entering power shut-off mode.The save pin must be an always-on pin.

If not specified, the save event is triggered by the opposite of the expression specified for the restore event.

Note: Expression is limited to the pin name and the inversion of the pin name. An expression containing only the pin name indicates an active high polarity. An expression containing the inversion of the pin name indicates an active low polarity.

6

Quick Reference

```
create_analysis_view
     -name string
     -mode mode
     -domain_corners domain_corner_list

create_bias_net
     -net net
     [-driver pin]
     [-user_attributes string_list]
     [-peak_ir_drop_limit float]
     [-average_ir_drop_limit float]

create_global_connection
     -net net
     -pins pin_list
     [-domain domain | -instances instance_list]

create_ground_nets
     -nets net_list
     [-voltage string]
     [-internal]
     [-user_attributes string_list]
     [-peak_ir_drop_limit float]
     [-average_ir_drop_limit float]

create_isolation_rule
     -name string
     -isolation_condition expression
     {-pins pin_list | -from power_domain_list | -to power_domain_list}...
     [-isolation_target {from|to}] [-isolation_output {high|low|hold}]
     [-exclude pin_list]

create_level_shifter_rule
     -name string
     {-pins pin_list | -from power_domain_list | -to power_domain_list}...
     [-exclude pin_list]
```

```
create_mode_transition
     -name string
     -from_mode power_mode -to_mode power_mode
     -start_condition expression [-end_condition expression]
     [-clock_pin clock_pin [-cycles number | -latency float]]

create_nominal_condition
     -name string
     -voltage float
     [-pmos_bias_voltage float] [-nmos_bias_voltage float]

create_operating_corner
     -name string
     -voltage float
     [-process float]
     [-temperature float]
     -library_set library_set

create_power_domain
     -name power_domain
     { -default [-instances instance_list] [-boundary_ports pin_list]
     | -instances instance_list [-boundary_ports pin_list]
     | -boundary_ports pin_list }
     [ -shutoff_condition expression ]
     [ -default_restore_edge expression ]
     [ -default_save_edge expression ]
     [ -power_up_states {high|low|random} ]

create_power_mode
     -name string
     -domain_conditions domain_condition_list
     [-default]

create_power_nets
     -nets net_list
     [-voltage string]
     [-external_shutoff_condition expression | -internal]
     [-user_attributes string_list]
     [-peak_ir_drop_limit float]
     [-average_ir_drop_limit float]

create_power_switch_rule
     -name string
     -domain power_domain
     {-external_power_net net | -external_ground_net net}

create_state_retention_rule
     -name string
     { -domain power_domain | -instances instance_list }
     [-restore_edge expression [ -save_edge expression ]]
```

```
define_always_on_cell
     -cells cell_list [-library_set library_set]
     [ [-power_switchable LEF_power_pin | -ground_switchable LEF_ground_pin]
       -power LEF_power_pin  -ground LEF_ground_pin ]

define_isolation_cell
     -cells cell_list [-library_set library_set]
     [-always_on_pin pin_list]
     [ {-power_switchable LEF_power_pin | -ground_switchable LEF_ground_pin}
       -power LEF_power_pin  -ground LEF_ground_pin ]
     [-valid_location { from | to}]
     [-non_dedicated]
     -enable pin

define_level_shifter_cell
     -cells cell_list [-library_set library_set]
     [-always_on_pin pin_list]
     -input_voltage_range {voltage | voltage_range}
     -output_voltage_range {voltage | voltage_range}
     [-direction {up|down|bidir}]
     [-output_voltage_input_pin pin]
     { -input_power_pin LEF_power_pin [-output_power_pin LEF_power_pin]
     | [-input_power_pin LEF_power_pin] -output_power_pin LEF_power_pin }
     -ground LEF_ground_pin
     [-valid_location { from | to}

define_library_set
     -name library_set
     -libraries library_list

define_open_source_input_pin
     -cells cell_list -pin pin_name
     [-library_set library_set]

define_power_clamp_cell
     -cells cell_list
     -data pin_name
     -power pin_name [-ground pin_name]
     [-library_set library_set]

define_power_switch_cell
     -cells cell_list [-library_set library_set]
     -stage_1_enable expression [-stage_1_output expression]
     [-stage_2_enable expression [-stage_2_output expression]]
     -type {footer|header}
     [ -power_switchable LEF_power_pin  -power LEF_power_pin
     | -ground_switchable LEF_ground_pin -ground LEF_ground_pin ]
     [ -on_resistance float]
     [ -stage_1_saturation_current float] [ -stage_2_saturation_current float]
     [ -leakage_current float ]
```

```
define_state_retention_cell
     -cells cell_list [-library_set library_set]
     [-always_on_pin pin_list]
     [-clock_pin pin]
     -restore_function expression [-restore_check expression]
     [-save_function expression] [-save_check expression]
     [ {-power_switchable LEF_power_pin | -ground_switchable LEF_ground_pin}
       -power LEF_power_pin  -ground LEF_ground_pin ]

end_design

identify_always_on_driver
     -pins pin_list [-no_propagation]

identify_power_logic
     -type isolation
     -instances instance_list

set_array_naming_style
     [string]

set_cpf_version
     [value]

set_design
     module [-ports port_list]

set_hierarchy_separator
     [character]

set_instance
     [hier_instance [-merge_default_domains]
     [-port_mapping port_mapping_list]]

set_power_target
     { -leakage float | -dynamic float
     | -leakage float -dynamic float}

set_power_unit
     [pW|nW|uW|mW|W]

set_register_naming_style
     [string%s]

set_switching_activity
     { {-all | -pins pin_list | -instances instance_list [-hierarchical]}
        -probability float -toggle_rate float }
     | [-clock_pins pin_list] -toggle_percentage float }
     [-mode mode]

set_time_unit
     [ns|us|ms]
```

```
update_isolation_rules -names rule_list
     { -location {from | to}
     | -cells cell_list -library_set library_set
     | -prefix string
     | -combine_level_shifting
     | -open_source_pins_only}...

update_level_shifter_rules
     -names rule_list
     { -location {from | to}
     | -cells cell_list -library_set library_set
     | -prefix string }...

update_nominal_condition
     -name condition
     -library_set library_set

update_power_domain
     -name domain
     { -internal_power_net net | -internal_ground_net net
     | -min_power_up_time float | -max_power_up_time float
     | -pmos_bias_net net | -nmos_bias_net net | -user_attributes string_list
     | -rail_mapping rail_mapping_list -library_set library_set} ...

update_power_mode
     -name mode
     { -activity_file file -activity_file_weight weight
     | -sdc_files sdc_file_list
     | -peak_ir_drop_limit domain_voltage_list
     | -average_ir_drop_limit domain_voltage_list
     | -leakage_power_limit float
     | -dynamic_power_limit float}...

update_power_switch_rule
     -name string
     { -enable_condition_1 expression [-enable_condition_2 expression]
     | -acknowledge_receiver pin
     | -cells cell_list -library_set library_set
     | -prefix string
     | -peak_ir_drop_limit float
     | -average_ir_drop_limit float }...

update_state_retention_rules
     -names rule_list
     {-cell_type string | -cell libcell}
     -library_set library_set
```