What’s New in Common Power Format

Version 2.0
October 2019

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Contents

1

What’s Changed from CPF 1.1 to CPF 2.0

New Features

New Concepts

Introduced Concept of Power Source Domain

Introduced the concept of a power source domain, that is a power domain that contains the output supply of a voltage regulator.

Introduced the Concept of Global Cell

The global cell replaces the always-on cell. Like the always-on cell, the global cell is a special cell that has secondary power or ground pins in addition to the primary power and ground pins (followpins). In some cell designs, when the primary power or ground are switched off, the cell function can be different from the normal function when the primary power and ground are on. Also, in some cases, the cell can also have isolation logic built in at the cell input pins.

To support this new cell, the `define_global_cell` command was added.

Introduced the Concept of Generic Mode

Introduced the concept of a generic mode (also referred to as mode), which is a static state of a design that performs one or more intended design functions. Typically, it is determined by the states of memory elements, states of power domains, and signal values.

To create a generic mode that applies to a subset of domains, the `create_mode` command was added.

Introduced Concept of Power Design

Introduced the concept of a power design, that is a unique power structure that can be associated with either a top design, or with one or more logic modules.

Introduced Concept of Input and output Domains for Level Shifter Cells

Inputs and output domains are the domains whose primary power and ground nets are connected to the input and output supply of the level shifter cells.

Introduced Concept of Domain Crossing

A domain crossing is a connection between one driver and one load in two different power domains. This concept replaces the net segment terminology which was not as precise and implied that multiple isolation cells could be inserted on the same crossing. Using the concept of domain crossing, clearly indicates that insertion of isolation cells and level shifters occurs on a full crossing basis.

Power Domain Changes

- Added Support for Assigning Leaf-level Library Cell instances to Power Domains on page 8
- Made Instance and Boundary Port Specification in Domain Specification More Flexible on page 8
- Added Flexibility to Update the Members of a Power Domain on page 8

Added Support for Assigning Leaf-level Library Cell instances to Power Domains

In CPF 1.1, you could only specify instances of hierarchical modules and macro cells for the `-instances` option of the `create_power_domain` command. In CPF 2.0, the semantics have been expanded to allow instances of any library cells.

Made Instance and Boundary Port Specification in Domain Specification More Flexible

To support this feature, the `-exclude_instances` and `-exclude_ports` options were added to the `create_power_domain` command.

Added Flexibility to Update the Members of a Power Domain

The `-instances` and `-boundary_ports` options were added to the `update_power_domain` to allow appending newly-created instances and ports to the member list of a previously created power domain.

Hiearchical Flow Improvements

- Added Support for Analog Ports in Macro Models on page 9
- Added Support for Specification of Macro Cell Ports Connected to Bonding Ports on page 9
- Added Support for Specification of Macro Cell Ports Connected to Diode Cells on page 9
- Added Capability to Create a Single Macro Model that can Apply to Multiple Cells on page 9
- Changed Semantics for set_design on page 10
- Added Support to Link Multiple HDL Modules to a Power Design on page 10
- Appending Power Intent to an Existing Design on page 10
- Expanded Semantics for Virtual Ports on page 10
- Clarified Semantics for Instance Port Mapping on page 10

Added Support for Analog Ports in Macro Models

The `set_analog_ports` command allows you to specify a list of analog signal pins that must also be connected to pins declared as analog in other macro models or in pad cells.

Added Support for Specification of Macro Cell Ports Connected to Bonding Ports

To specify a set of macro cell ports that can directly connect to a bonding port at the chip level, the `set_pad_ports` command was added. No low power logic (such as level shifters and isolation cells) can be inserted between the top-level bonding ports and the pad ports of a macro.

Added Support for Specification of Macro Cell Ports Connected to Diode Cells

To specify a list of pins of a macro cell that connect to the positive and negative pins of a diode, the `set_diode_ports` command was added.

Added Capability to Create a Single Macro Model that can Apply to Multiple Cells

To formally allow a single macro model to apply to more than one cell, the `-cells` option was added to the `set_macro_model` command. For example, memory cells with the same architecture but different bits can use the same macro cell definition.

This option also facilitates better error checking in cases where the wrong macro model was applied to an instance of a cell.

Changed Semantics for set_design

The semantics of the `set_design` command have been changed:

- CPF1.1 specified the power intent for one module.
- CPF2.0 specifies an entity, referred to as a power design, that defines power intent. A power design may be applied to any of the following:
  - ❑ a top design
  - ❑ instantiations of any module
  - ❑ instantiations of one or more explicitly specified modules

Added Support to Link Multiple HDL Modules to a Power Design

The `-modules` option was added to the `set_design` command to associate multiple HDL modules to a power design.

Appending Power Intent to an Existing Design

- CPF1.1 appends power intent to an existing design using additional `set_design end_design` pairs
- CPF2.0 appends power intent to an existing power design using the new `update_design` command paired with `end_design`

Expanded Semantics for Virtual Ports

In CPF 1.1, the `-ports` option of the `set_design` command specified input ports only.

In CPF 2.0, new options explicitly specify `-input_ports`, `-output_ports`, and `-inout_ports`.

Note: The existing `-ports` option still exists for backward compatibility.

Clarified Semantics for Instance Port Mapping

The semantics of the `-port_mapping` option of the `set_instance` command have been clarified to include the specification of real ports (such as macro model pins) in addition to virtual ports.

Isolation Enhancements

- Added Support for Isolation Cell That Can be Placed in Power and Ground Shutoff Domain on page 11
- Added Support for Isolation Cell That Can Be Placed in Any Domain on page 11
- Added Support for Clamp-Type Isolation Cell on page 12
- Added Support for Isolation Cell with Synchronous Enable Control on page 12
- Added Support to Enable Multi-bit Isolation Cells on page 12
- Changed Semantics of Location in Rules on page 12
- Added Capability to Specify A Model for Special Isolation Logic on page 12
- Changed Semantics of -within_hierarchy in Rules on page 13
- Added Support for Isolation Logic That Can Be Placed in Parent Hierarchy on page 13
- Added Capability to Force the Insertion of Isolation Logic on page 13
- Added Capability to Specify a Name Suffix on page 13

Added Support for Isolation Cell That Can be Placed in Power and Ground Shutoff Domain

To support these type of isolation cells, the `define_isolation_cell` command can be specified with both the `-power_switchable` and `-ground_switchable` options.

Added Support for Isolation Cell That Can Be Placed in Any Domain

Isolation cells without power and ground pins that connect through abutment of the cells (followpins), that is, with only a secondary power and ground pin can be placed in any domain.

To support this type of isolation cell, the following changes were made to the `define_isolation_cell` command:

- Added the `any` value for the `-valid_location` option
- The command can be specified without the `-power_switchable` and `-ground_switchable` options

Implementation tools must connect the secondary power and ground pin of this isolation cell to the supplies of the secondary domain of the rule. For more information, see Secondary Power Domain of Isolation Instances. Implementation tools must also ensure that placement of these cells does not break the follow pin connection of the other cells that are placed in the same standard cell row.

Added Support for Clamp-Type Isolation Cell

To support clamp-type isolation cells, the following changes were made:

- Added the `-clamp` option to the `define_isolation_cell` command.
- Added the `clamp_high` and `clamp_low` values for the `-isolation_output` option of the `create_isolation_rule` command.

Added Support for Isolation Cell with Synchronous Enable Control

To support isolation cells with synchronous enable control, the following changes were made:

- Added the `-aux_enables` option to the `define_isolation_cell` command.
- Added the `-isolation_control` option to the `create_isolation_rule` command.

Added Support to Enable Multi-bit Isolation Cells

Added the `-pin_groups` option to the `define_isolation_cell` command.

Changed Semantics of Location in Rules

Previously, when you specified the `-location` and `-cells` options together in the `update_isolation_rules` command, the `-cells` option took precedence—the location of the cells was determined by the `-valid_location` specification in the `define_isolation_cell` command.

In CPF 2.0, if you specify the `-location` and `-cells` options together in the `update_isolation_rules` command, the value of the `-location` option must match the value specified for the `-valid_location` in the `define_isolation_cell` command. Otherwise an error will be given.

Added Capability to Specify A Model for Special Isolation Logic

Added the `-use_model` to the `update_isolation_rules` command to indicate that a simulation tool must use the functional model of the first cell specified in the `-cells` option. The `-pin_mapping` option was added to specify the connection between the pins of the functional model and the pins or ports in the design. If a macro model exists for the functional model, the `-domain_mapping` option can be used to specify the mapping between the domains in the macro model and the domains in the design.

Changed Semantics of -within_hierarchy in Rules

Previously, when you specified the -within_hierarchy option in the update_isolation_rules command, the power domain of the specified instance had to match the power domain of the default location or the location specified with the -location option.

In CPF 2.0, the power domain of the specified instance no longer needs to match the power domain specified by the -location option. The power domain of the instance specified with the -within_hierarchy option takes precedence.

This change allows you to insert the isolation logic in a power domain that is neither the from nor the to power domain. This may be needed when the physical constraints require the placement of the isolation in a separate domain.

Also, if you specify the -within_hierarchy and -cells options together, the -valid_location of the cells specified in the define_isolation_cell command must be compatible with the power domain of the hierarchical instance specified with the -within_hierarchy option in the update_isolation_rules. Otherwise an error will be given.

Added Support for Isolation Logic That Can Be Placed in Parent Hierarchy

To support the placement of isolation logic in the logic hierarchy of the parent instance of the nets selected using the -from, -to, and -pins options in the create_isolation_rule command, the parent value can now be specified for the -location option of the update_isolation_rules command.

Added Capability to Force the Insertion of Isolation Logic

To support forced insertion, the -force option was added to the create_isolation_rule command. You must specify the -force option with a -pins option. If you specify a -to or -from option with the -force option, they will be ignored.

If you use the -force option to create the isolation rule, the isolation logic can only be inserted in the default location or in the parent hierarchy (see the -location option of the update_isolation_rules command).

Added Capability to Specify a Name Suffix

To specify the name suffix to be used when creating the isolation logic, the -suffix option was added to the update_isolation_rules command.

Level Shifter Enhancements

Added Support for Level Shifter Cell That Can Be Placed in Any Domain

Level shifter cells whose input and output pins are specified as non-followpins can be placed in any domain.

To support this type of level shifter cell, the `any` value was added for the `-valid_location` option of the `define_level_shifter_cell` command.

If the valid location of a level shifter cell is `any`, it can be used in a rule using `-within_hierarchy` with any hierarchy location.

Added Support for Bypass Level Shifter

To support bypass level shifters, the following changes were made:

- Added the `-bypass_enable` option to the `define_level_shifter_cell` command
- Added the `-bypass_condition` option to the `create_level_shifter_rule` command.

Added Support to Specify Lists for Input and Output Voltages in Level Shifter Definition

In previous releases, you needed to specify either a single voltage or a voltage range for the input (source) and output (destination) supply voltages. However, when specifying ranges, some unintended input and output combination could be implied. Consider the following option combination in the `define_level_shifter_cell` command:

```
-input_voltage_range 0.8:1.0:0.2 -output_voltage_range 1.0:1.2:0.2
```

This could imply that the level shifter can shift from 0.8 to 1.2, while the level shifter can only shift from either 0.8 to 1.0 or from 1.0 to 1.2.

In CPF 2.0, you can specify lists of voltages and voltage ranges. If you specify a list of voltages or ranges for the input supply voltage, you must also specify a list of voltages or voltage ranges for the output supply voltage. Both lists must be ordered and have the same number of elements. That is, each member in the list of input voltages (or ranges) has a corresponding member in the list of output voltages (or ranges).

Added Support for Multi-Stage Level Shifters

To support multi-stage level shifters, the following changes were made:

- Added `-multi_stage` option to the `define_level_shifter_cell` command.
- Added the `-through` option to the `update_level_shifter_rules` command which specifies the subsequent domains for the multi-stage level shifting between the first domain (specified with `-from`) and the last domain (specified with `-to`).

  In addition, the `-cells` option of the `update_level_shifter_rules` command can accept a list of lists for multi-stage level shifters.

Added Support to Enable Multi-bit Cells

Added the `-pin_groups` option to the `define_level_shifter_cell` command.

Changed Semantics of Location in Rules

Previously, when you specified the `-location` and `-cells` options together in the `update_level_shifter_rules` command, the `-cells` option took precedence—the location of the cells was determined by the `-valid_location` specification in the `define_level_shifter_cell` command.

In CPF 2.0, if you specify the `-location` and `-cells` options together in the `update_level_shifter_rules` command, the value of the `-location` option must match the value specified for the `-valid_location` in the `define_level_shifter_cell` command. Otherwise an error will be given.

Expanded Support for Input Voltage Tolerance

In previous versions of CPF, the input voltage tolerance could only be specified for input pins of a macro model.

In CPF 2.0, the concept has been generalized for all input pins. You can also specify a different tolerance for power and ground voltages. You can specify the pins or the domain to which the constraints apply. If the input voltage tolerance is violated, a power or ground level shifter is required for the specified pins.To support these new features the syntax of the `set_input_voltage_tolerance` command was revised.

Changed Semantics of -within_hierarchy in Rules

Previously, when you specified the `-within_hierarchy` option in the `update_level_shifter_rules` command, the power domain of the specified instance had to match the power domain of the default location or the location specified with the `-location` option.

In CPF 2.0, the power domain of the specified instance no longer needs to match the power domain specified by the `-location` option. The power domain of the instance specified with the `-within_hierarchy` option takes precedence.

This change allows you to insert level shifters in a power domain that is neither the from nor the to power domain. This may be needed when the physical constraints require the placement of the level shifter in a separate domain.

Also, if you specify the `-within_hierarchy` and `-cells` options together, the `-valid_location` of the cells specified in the `define_level_shifter_cell` command must be compatible with the power domain of the hierarchical instance specified with the `-within_hierarchy` option of the `update_level_shifter_rules` command. Otherwise an error will be given.

Added Support for Level Shifters That Can Be Placed in Parent Hierarchy

To support the placement of level shifters in the logic hierarchy of the parent instance of the nets selected with the `-from`, `-to`, and `-pins` options in the `create_level_shifter_rule` command, the `parent` value can now be specified for the `-location` option of the `update_level_shifter_rules` command.

Added Capability to Force the Insertion of Level Shifter Logic

To support forced insertion, the `-force` option was added to the `create_level_shifter_rule` command. You must specify the `-force` option with a `-pins` option. If you specify a `-to` or `-from` option with the `-force` option, they will be ignored.

If you use the `-forc`e option to create the level shifter rule, the level shifters can only be inserted in the default location or in the parent hierarchy (see the `-location` option of the `update_level_shifter_rules` command).

Added Capability to Specify Input and Output Domains for the Level Shifter Cells

To specify the input and output domains for a level shifter the `-input_domain` and `-output_domain` options were added to the `create_level_shifter_rule` command.

Added Capability to Specify a Name Suffix

To specify the name suffix to be used when creating the level shifters, the `-suffix` option was added to the `update_level_shifter_rules` command.

State Retention Enhancements

- Added Capability to Specify A Model for Special State Retention Logic on page 18
- Added Support for Retention Check and Precondition on page 18
- Added Capability to Preserve Output While Power is Off on page 18
- Added Capability to Require the Implementation of Retention Logic on page 18

Added Capability to Specify A Model for Special State Retention Logic

Added the `-use_model` to the `update_state_retention_rule`s command to indicate that a simulation tool must use the functional model of the first cell specified in the `-cells` option. The `-pin_mapping` option was added to specify the connection between the pins of the functional model and the pins or ports in the design. If a macro model exists for the functional model, the `-domain_mapping` option can be used to specify the mapping between the domains in the macro model and the domains in the design.

Added Support for Retention Check and Precondition

In CPF 2.0, a `-retention_precondition` option was added to the `create_state_retention_rule` command and a `-retention_check` option to the `define_state_retention_cell` command.

Added Capability to Preserve Output While Power is Off

To prevent the corruption of the output of the state retention logic when the primary power is off, the `-use_secondary_for_output` option was added to the `create_state_retention_rule` command

Added Capability to Require the Implementation of Retention Logic

To require the implementation of retention logic, the `-required` option was added to the `create_state_retention_rule` command.

Power Mode and Power Mode Control Group Changes

Power Mode Control Group Semantic Change

When compound power modes are specified, a design can be in more than one mode at a time. This can cause problems during verification when deterministic domain states are expected.

By allowing a power domain to be specified in multiple power mode control groups, this problem can be solved.

Note: In CPF 1.1, a power domain could only belong to one power mode control group.

Added Capability to Specify Illegal Power Mode Transitions

To support this feature, the `-illegal` option was added to the `create_mode_transition` command.

Added Capability to Associate Assertions with Power Mode Transitions

To support this feature, the `-assertions` option was added to the `create_mode_transition` command.

Added Capability to Specify Condition for Power Mode

In CPF 2.0, the `-condition` option was added to the `create_power_mode` command to indicate when the design is in the specified mode.

This option is only interpreted by the simulation and verification tools.

Pad and Regulator Modeling Changes

- Simplified Pad Modeling on page 20
- Simplified Pad Instantiation on page 20
- Improved Capability to Model an On-Chip Voltage Regulator on page 21

Simplified Pad Modeling

Until now, you could only use a CPF macro model to model the IO pads. This method can be tedious and error-prone.

The `define_pad_cell` command allows you to create a model for simple pad cells. This command allows you to identify

- Pins that will be directly or indirectly connected to the board
- Pins which connect to the core and have internal isolation logic and their corresponding isolation control signal
- Groups of pins that belong to the same power domain
- Analog signal pins that connect to other analog pins

This model is not sufficient when your pad cell contains any of the following:

- Internal power switch and/ or state retention
- Complex internal isolation control
- Internal feed- through nets for data signals

In this case, you need to use a CPF macro model for the pad cell.

Simplified Pad Instantiation

The `create_pad_rule` command allows you to specify the mapping to a top-level domain of either

- A pin group in the pad cell definition
- A power domain in the macro model definition

Improved Capability to Model an On-Chip Voltage Regulator

To model a voltage regulator the following features were added:

- Introduced the concept of a *power source domain*, that is a power domain that contains the output supply of the voltage regulator.
- Added the `-power_source` option to the `create_power_domain` to identify a power source domain.
- Added the `set_power_source_reference_pin` command to identify an input pin of a macro model as the voltage reference of the voltage regulator.
- Allowed specification of voltage ranges in the `create_nominal_condition` command.

Simulator-Related Changes

- Added Capability to Specify Simulator Action when Power Domain is Switched off or Restored on page 22
- Added Capability to Specify a CPF Model for a Testbench on page 22
- Added Control over Signal Values in a Power Off Domain on page 22

Added Capability to Specify Simulator Action when Power Domain is Switched off or Restored

To allow users to specify the immediate action to be taken when the power is switched off in power domains or restored to power domains, the `set_sim_control` command was added.

For example, initial statements are non-synthesizable code used in simulation to create proper startup conditions at time zero of the simulation. Previous CPF versions had no provision to specify the action to be taken when the power is restored to a switchable power domain. This time is very similar to initialization of the simulation at time zero.

Added Capability to Specify a CPF Model for a Testbench

To support this feature, the `-testbench` option was added to the `set_design` command.

Added Control over Signal Values in a Power Off Domain

To control the signal values of elements in a power domain when this domain is being switched off, the `-power_down_states` option was added to the `create_power_domain` command.

Other Improvements

- Added Capability to Specify Inverted Polarity for Equivalent Control Pins on page 23
- Added Capability to Specify Libraries Specific to Power Analysis on page 23
- Improved Support for Clamp Diodes on page 23
- Improved Support for Open Source Pin on page 24
- Added Support for Deep Nwell and Deep Pwell on page 24
- Added Additional Operators for Boolean Expressions on page 24
- Added Capability to Specify a Global Connection using Ports or Liberty PG Type on page 24
- Expanded Definition of Library Set for Operating Corners on page 24
- Added Capability to Specify a Default Analysis View on page 25
- Added Capability to Find Design Objects within a Specified Scope on page 25

Added Capability to Specify Inverted Polarity for Equivalent Control Pins

By allowing specification of equivalent pins with inverted polarity in the `set_equivalent_control_pins` command, insertion of special low power logic during synthesis can sometimes be simplified by using a control signal with the opposite polarity.

For example, with previous versions of CPF, when a limited number of isolation cells is available in the library, a combination of some conditions in the isolation rules might require the addition of an inverter to the control enable line. By using a control signal with the opposite polarity, the inverter is no longer needed.

Added Capability to Specify Libraries Specific to Power Analysis

To support this feature, the `-power_library_set` option was added to the `create_operating_corner` and `update_nominal_condition` commands.

Improved Support for Clamp Diodes

Previous versions of CPF only supported power clamp cells with one data pin.

To improve the support of clamp diodes, the `define_power_clamp_cell` was superseded by the new `define_power_clamp_pins` command.

This new command supports

- Power, ground, and power and ground clamp cells with one or more data pins
- Complex cells that can have input pins with built-in clamp diodes

Improved Support for Open Source Pin

Added the `-type` option to the `define_open_source_input_pin` command to specify the type of the transistor connected to the pin within the cell.

Added Support for Deep Nwell and Deep Pwell

Added the `-deep_nwell` and `-deep_pwell` options to specify bias nets in the `update_power_domain` command.

Added the `-deep_nwell_voltage` and `-deep_pwell_voltage` options to the `create_nominal_condition` command.

Added Additional Operators for Boolean Expressions

The following bit-wise and logical operators are now allowed in CPF Boolean expressions: `~ ^ && ||`

Added Capability to Specify a Global Connection using Ports or Liberty PG Type

The `create_global_connection` command allows specification of a global connection between

- the top-level module ports
- an instance’s cell pins by matching the `pg_type` in the corresponding Liberty definitions of the cell pins

Expanded Definition of Library Set for Operating Corners

The `create_operating_corner` command now facilitates MMMC analysis and optimization by allowing the specification of more than one library set in the `-library_set` option.

Added Capability to Specify a Default Analysis View

The -default option was added to the `create_analysis_view` command to specify the default binding of libraries for different corners to the corresponding power domains.

Added Capability to Find Design Objects within a Specified Scope

To locate and return design objects which can then be used as arguments to other CPF commands, the `find_design_objects` command was added.

Command Change Summary

Renamed Command Options

Command	Old Option (1.1)	New Option (2.0)
set_input_voltage_tolerance	-ports	-pins
	-bias [lower:]upper	-power lower[:upper] -ground [lower:]upper

New Commands

```
create_mode
create_pad_rule
define_global_cell
define_pad_cell
define_power_clamp_pins
find_design_objects
set_analog_ports
set_diode_ports
set_pad_ports
set_power_source_reference_pin
set_sim_control
update_design
```

New Command Options

Command	New Option/Argument
create_analysis_view	-default
create_global_connection	-pg_type
	-ports
create_isolation_rule	-force
	-isolation_control

Command	New Option/Argument
create_level_shifter_rule	-bypass_condition
	-force
	-input_domain
	-output_domain
create_mode_transition	-assertions
	-illegal
create_nominal_condition	-deep_nwell_voltage
	-deep_pwell_voltage
create_operating_corner	-power_library_set
create_power_domain	-exclude_instances
	-exclude_ports
	-power_down_states
	-power_source
create_power_mode	-condition
create_state_retention_rule	-required
	-retention_precondition
	-use_secondary_for_output
define_isolation_cell	-aux_enables
	-clamp
	-pin_groups
define_level_shifter_cell	-bypass_enable
	-multi_stage
	-pin_groups
define_open_source_input_pin	-type
define_state_retention_cell	-retention_check

Command	**New Option/Argument**
`set_design`	`-inout_ports`
	`-input_ports`
	`-modules`
	`-output_ports`
	`power_design`
	`-testbench`
`set_macro_model`	`-cells`
`update_isolation_rules`	`-domain_mapping`
	`-pin_mapping`
	`-suffix`
	`-use_model`
`update_level_shifter_rules`	`-suffix`
	`-through`
`update_nominal_condition`	`-power_library_set`
`update_power_domain`	`-boundary_ports`
	`-deep_nwell_net`
	`-deep_pwell_net`
	`-instances`
`update_state_retention_rules`	`-domain_mapping`
	`-pin_mapping`
	`-use_model`

Other Syntax Changes

Command	**New Option**
`create_isolation_rule`	`-isolation_output` option can take `clamp_high` and `clamp_low` values
`create_operating_corner`	allow specification of more than one library set in the `-library_set` option.

Command	**New Option**
`define_isolation_cell`	`-power_switchable` and `-ground_switchable` options were made optional
	allow specification of combination of `-power_switchable` and `-ground_switchable` options
	`-valid_location` can take `any` value
`define_level_shifter_cell`	`-valid_location` can take `any` value
	`-input_voltage_range` and `-output_voltage_range` allow specification of list of voltages and voltage ranges
	`-input_ground_pin`, `-input_power_pin`, `-output_ground_pin`, `-output_power_pin`, `-ground` and `-power` options were made optional
`update_isolation_rules`	`-location` option can take `parent` value
`update_level_shifter_rules`	`-location` option can take `parent` value

Semantics Changes

Command	**New Option**
`create_power_domain`	-instances allows instances of any library cells
`set_equivalent_control_pins`	allow specification of equivalent pins with inverted polarity
`set_instance`	-design now takes a power design name specified by set_design.
`set_macro_model`	macro model definitions are now scope insensitive. first definition is used.
`update_level_shifter_rules`	-cells can take a list of lists for multi-stage level shifters

2

What’s Changed from CPF 1.0 to CPF 1.1

New Features

Introduced Concept of Base and Derived Power Domains

CPF 1.1 introduces the concepts of base and derived power domains.

- A power domain whose primary power supply provides power to another power domain through some power switch network is called a ***base*** domain.
- A power domain that derives its power from another power domain through some power switch network is called the ***derived*** domain.

To support this concept the `-base_domains` option to the `create_power_domain` command. This enables specification and simulation of domain dependencies.

Introduced Concept of Secondary Power Domain

CPF 1.0 only had the concept of a power domain which was defined as collection of instances that use the same power supply during normal operation and that can be switched on or off at the same time.

However, physical cells with multiple power pins were hard to model. Therefore the concept of secondary power domain was introduced.

A power domain X is a ***secondary*** power domain of a special low power instance if the primary power and ground nets of the domain X provide the power supply to the secondary power and (or) ground pins of the instance.

The secondary domain can be used to

- Enable the simulation of always-on cell
- Enable the simulation of deep sleep mode of retention cell
- Enable the physical implementation of always-on, retention logic and power switches

To support this concept the following changes were made:

- Added `-secondary_domain` option to the `create_isolation_rule` and the `create_state_retention_rule` commands.
- Added `identify_secondary_domain` command.

Improved Hiearchical Flow

To improve the hierarchical flow, the following concepts were introduced:

- Introduced Concept of Domain Mapping
- Introduced Concept of Power Mode Control Group
- Introduced Concept of Macro Model

Other improvements include

- Improved IP Support
- Defined Precedence for Rules
- Enhanced CPF Readability

Introduced Concept of Domain Mapping

CPF1.0 did not support mapping a block-level power domain into a higher-level power domain.

For example, when creating a hierarchical design, the domains of an IP are created without knowledge of final use. By introducing the concept of domain mapping, mapped power domains are treated as a single power domain.

Domain mapping can be used to

- Merge a block-level power domain into a top-level power domain in a hierarchical flow
- Reconfigure block level power domains at the top level

Mapping power domains causes some complications such as

- How to handle rules defined for the block-level and top-level power domains.
- How to merge power modes

To support this concept the following changes were made:

- Added `-domain_mapping` option to the `set_instance` command.
- Added `-honor_boundary_port_domain` option to the `set_design` command.

For more information on domain mapping, refer to Power Domain Mapping Concepts.

Introduced Concept of Power Mode Control Group

CPF 1.0 only supported top-level power mode definitions. However, in a hierarchical flow, a block can have its own power mode definitions. To describe how the power modes at the top-level are related to the power modes of a block, the concept of power mode control group was introduced.

A power mode control group is a set of power domains with an associated set of power modes and mode transitions that apply to this group only. A power mode group can contain other power mode groups.

To support power mode control groups, the following changes were added:

- Added the `set_power_mode_control_group` command.
- Added the `end_power_mode_control_group` command.
- Added the `-group_modes` option to the `create_power_mode` command.
- Added the `-group_views` option in the `create_analysis_view` command.

For more information, refer to Power Mode Control Groups.

Introduced Concept of Macro Model

Proper modeling of macro cells (such as a RAM) with complex power network is important because it serves as a specification for the

- Implementation tool to properly hook up the power and ground pins of the IP at the top level
- Verification tool to check for consistency between the implementation and the constraints specified in the CPF.
- Simulation tool to verify the behavior

Because the macro cell has only a behavioral model to describe its functionality, implementation and verification tools have to rely on the CPF modeling of the internal power network using the boundary ports.

To support modeling of macro cells, the concept of macro model was introduced.

To support this concept the following changes were made:

- Added `set_macro_model` command
- Added `end_macro_model` command

- Added `set_floating_ports` command
- Added `set_input_voltage_tolerance` command
- Added `set_wire_feedthrough_ports` command

For more information, refer to Modeling a Macro Cell.

Improved IP Support

To improve support of IP's, the following changes were made:

- Added support for parameterized IP by adding the `-paramater_mapping` option to the `set_design` command.
- Added support for specifying isolation constraints at the inputs of an IP by making the `-isolation_condition` option in the `create_isolation_rule` command optional.
- The `define_related_power_pins` command was added to specify the relationship between the power and data pins for IPs that have more than one set of power / ground pins.

Defined Precedence for Rules

In a hierarchical flow, block-level and top-level domains each can have rules defined with them. When you map power domains, the associated rules must be handled properly.

To handle the rule precedence, two categories of CPF rules are considered:

- **Specific** rules *explicitly* specify the targeted design objects.
- **Generic** rules do not explicitly specify the targeted design objects, but the targeted design objects can be *derived* from some option specified with the rule.

The following general rules of precedence apply:

1. Specific rules have a higher priority than generic rules.
2. Generic rules specified with both the `-from` and `-to` options have a higher priority than those generic rules specified with only `-from` or `-to` option.
3. If multiple CPF rules with the same priority are applied to the same design objects, the last rule wins.

In addition, for a hierarchical flow, the following rules of precedence apply:

The following policies should be followed by all tools when a block-level CPF is integrated into the top-level CPF using the `set_instance` command:

1. Block-level rules always overwrite the top-level rules if both rules apply to the same objects at the block-level after domain mapping.
2. Top-level rules are only applied to block-level objects if the block-level objects have no rule specified and the block is not a macro model.

   In this case, the tools should issue a warning to inform users that top-level CPF rules are applied to the block.
3. Block-level rules are never propagated to the top level.
4. For more information on the precedence of rules, refer to the Rules chapter in the *Common Power Format Language Reference*.

Enhanced CPF Readability

To improve the readability of the CPF files, the following enhancements were made:

- The `end_design` statement can now contain the name of the module used with the corresponding `set_design` command.
- The end_macro_model statement can now contain the name of the macro cell used with the corresponding `set_macro_model` command.

Isolation Enhancements

The following enhancements were made:

- Added support for isolation cell without enable pin by adding `-no_enable` option to the `define_isolation_cell` command and the `-no_condition` option to the `create_isolation_rule` command.

  IP blocks can have special requirements for input ports. For example, when the signals driving these input ports are switched off, the signals must be held at specific values. For these signals, no isolation condition can be specified because the IP developer has no knowledge of how the IP will be used.
- Added support for a default isolation rule for a power domain by adding the `-default_isolation_condition` option to the `create_power_domain` command.

- Added support for tristate isolation output by adding the `tristate` value to the `-isolation_output` option of the `create_isolation_rule` command.
- Added support for instantiating the isolation logic in any logic hierarchy within a power domain by adding the `-within_hierarchy` option to the `update_isolation_rules` command.
- Added support for isolation cells that must be put in the `on` or `off` domain by adding the `on` and `off` values for the `-valid_location` option to the `define_isolation_cell` command.

Level Shifter Enhancements

The following enhancements were made:

- Added support for instantiating the level shifters in any logic hierarchy within a power domain by adding the `-within_hierarchy` option to the `update_level_shifter_rules` command.
- Improved support for level-shifter cells with an enable pin by adding the `-enable` option to the `define_level_shifter_cell` command.
- Added support for ground level shifter by adding the `-ground_input_voltage_range`, `-ground_output_voltage_range`, `-input_ground_pin` and `-output_ground_pin` options to the `define_level_shifter_cell` command.
- Added support for level shifter cells that can be put in the driving or receiving domain by adding the `either` value for the `-valid_location` option to the `define_level_shifter_cell` command.

State Retention Enhancements

The following enhancements were made:

- Added support for latches by adding the `-target_type` to the `create_state_retention_rule` command.
- Added support for level-sensitive save and restore control by adding the `-restore_level` and `-save_level` options to the `create_state_retention_rule` command.
- Added support to more accurately match the implied RTL behavior with gate-level models by adding the `-restore_precondition`, `-save_precondition`, and `-secondary_domain` options to the `create_state_retention_rule` command.
- Added support to specify a list of instances that must be excluded from replacement with state retention elements by adding the `-exclude` option to the `create_state_retention_rule` command.
- Added support for master-slave type state retention cells which have neither a save nor restore pin. For more information, refer to Handling Master-Slave Type Retention Cells.

Power Switch Enhancements

The following enhancements were made:

- Added capability to specify the resistance of the power switch *per stage*.

  To support this feature, the `-on_resistance` option of the `define_power_switch_cell` command was replaced with the `-stage_x_on_resistance` options, where *x* corresponds to the stage.

- Added capability to specify an acknowledge receiver condition *for each stage* of the power switch cells.

  To support this feature, the `-acknowledge_receiver` option of the `update_power_switch_rule` command was replaced with the `-acknowledge_receiver_x` options, where *x* corresponds to the stage.

Power Mode Enhancements

CPF 1.0 only covered either the on or off state of a power domain. However, there is a need to also be able to specify when a power domain is in standby mode and to support forward bias and reverse bias.

To support specification of the state of a power domain, the `-state` option with values `on`, `off`, and `standby` option was added to the `create_nominal_condition` command.

To improve the functional modeling of the biased state, the `-ground_voltage` option was added to the `create_nominal_condition` command.

To improve the implementation of designs using body bias, the `-nmos_bias_voltage` and `-pmos_bias_voltage` options were added to the `create_operating_corner` command.

To describe forward body biasing, you must specify -`pmos_bias_voltage` with a value smaller than the value of -`voltage` and -`nmos_bias_voltage` with a value greater than the value of -`ground_voltage`. Forward biasing can be used to improve performance in the `on` state.

To describe reverse body biasing, you must specify -`pmos_bias_voltage` with a value larger than the value of -`voltage` and -`nmos_bias_voltage` with a value smaller than the value of -`ground_voltage`. Reverse body biasing can be used to achieve the standby state.

To describe reverse source biasing, you must specify the voltage values for the power supply (`-voltage`) and ground supply (`-ground_voltage`) that cause the logic to be in the `on` or `standby` state.

Improved Support for DVFS Designs

- Added Capability to Specify Control Conditions for Different States of a Domain
- Added Support for Domain Specific Transition Time

Added Capability to Specify Control Conditions for Different States of a Domain

To define active state and conditions for each power domain, the `-active_state_conditions` option was added to the `create_power_domain` command.

Added Support for Domain Specific Transition Time

In CPF 1.0 mode transition times applied to all power domains involved in the same mode transition. The transition time could be specified in

- absolute time—using the `-latency` option of the `create_mode_transition` command
- clock cycles—using the using the `-cycles` option of the `create_mode_transition` command

In the CPF 1.0 extended version you can now specify a minimum and maximum number for the `-latency` and the `-cycles` option of the `create_mode_transition` command.

Starting with the CPF 1.0 extended version mode transition times can be specified per power domain. To specify the required time for a domain for a specific state transition, the following options were added to the `update_power_domain` command:

- `-transition_latency {`*`from_nom latency_list`*`}`

  Specifies the nominal condition of the starting power state, followed by a list of transition times to complete a transition to the next power state.

- `-transition_cycles {`*`from_nom cycle_list clock_pin`*`}`

  Specifies the nominal condition of the starting power state, followed by a list of transition cycles to complete the transition to the next power state, followed by the clock pin.

- `-transition_slope`

  Specifies the transition rate(s) for the supply voltage of the domain during any state transition of the domain.

The specified time can be used by simulation to simulate the domain transition time or used by physical implementation tool as constraint when designing the power distribution network

Other Improvements

- Added Capability to Identify Illegal Power Modes
- Added Capability to Associate User Attributes with An Analysis View
- Added Control over Assertion Evaluation
- Added Support for Equivalent Control Pins
- Equivalent Power and Ground Nets
- Refined Specification of SDC Files
- Change in Hierarchy Delimiter
- Change in Object Names

Added Capability to Identify Illegal Power Modes

To assert that a particular configuration of domain conditions and power mode control group conditions is illegal, the `assert_illegal_domain_configurations` command was added.

Added Capability to Associate User Attributes with An Analysis View

To support this feature, the `-user_attributes` option was added to the `create_analysis_view` command.

Added Control over Assertion Evaluation

By default, assertions remain active when the power domain is powered down. This may create a lot of false assertions.

To inhibit evaluation of any selected assertion instance when its related power domain is powered down the `create_assertion_control` command was added.

Added Support for Equivalent Control Pins

To define a list of this pins that are equivalent with a master control pin, the `set_equivalent_control_pins` command was added. The master control pin is part of the definition of a shutoff condition, isolation condition or state retention condition.

Equivalent Power and Ground Nets

If a power domain is implemented in multiple disjoint physical regions, each region has its own power and ground nets. To declare these power and ground nets equivalent to the primary power and ground nets of a power domain, the `-equivalent_power_nets` and `-equivalent_ground_nets` options were added to the `update_power_domain` command.

Refined Specification of SDC Files

Added capability to group SDC files with only hold or setup constraints by adding the `-hold_sdc_files` and `-setup_sdc_files` options to the `update_power_mode` command.

Change in Hierarchy Delimiter

Added the colon (:) as allowed hierarchy delimiter character, and removed the caret (^) as allowed hierarchy delimiter character.

Change in Object Names

Object names can no longer contain the dollar sign ($).

Command Change Summary

Obsoleted Command Options

Command	Obsolete Option(s)	Equivalent?
`set_instance`	`-merge_default_domains`	`-domain_mapping` The old option only allowed for merging of the default domains. The new option is more general.
`update_isolation_rules`	`-combine_level_shifting`	`None` By default, implementation tools should try to find a suitable cell that has both isolation and level-shifting functionality. See How to Handle Multiple Rules that Apply to a Physical Net.
	`-library_set`	`None` Implementation tools should automatically select valid cells from the library sets associated with the power domain through the `update_nominal_condition` or `create_operating_corner` commands.
`update_level_shifter_rules`	`-library_set`	`None` Implementation tools should automatically select valid cells from the library sets associated with the power domain through the `update_nominal_condition` or `create_operating_corner` commands.

Command	Obsolete Option(s)	Equivalent?

`update_power_domain`	`-max_power_up_time`
	`-min_power_up_time`
		`-transition_latency`
		`-transition_cycles`
		Because a power domain can be in a different on state (with different voltages) and it can also be in a standby state, a more fine grain specification of domain transition time was needed.

	`-rail_mapping` `-library_set`	`None` The options were initially added to check for power/ground connectivity for special cells with multiple power and ground pins. With the introduction of secondary power domain and domain mapping, this is no longer needed. All the power/ground net connections are described in CPF.
`update_power_switch_rule`	`-library_set`	`None` Implementation tools should automatically select valid cells from the library sets associated with the power domain through the `update_nominal_condition` or `create_operating_corner` commands.
`update_state_retention_rules`	`-library_set`	`None` Implementation tools should automatically select valid cells from the library sets associated with the power domain through the `update_nominal_condition` or `create_operating_corner` commands.

Renamed Command Options

Command	Old Option (1.0)	New Option (1.1)
`create_mode_transition`	`-from_mode`	`-from`
	`-to_mode`	`-to`
`define_isolation_cell`	`-always_on_pin`	`-always_on_pins`
`define_level_shifter_cell`	`-always_on_pin`	`-always_on_pins`
	`-output_voltage_input_pin`	`-enable`
`define_power_switch_cell`	`-on_resistance`	`-stage_`*`x`*`_on_resistance`
`define_state_retention_cell`	`-always_on_pin`	`-always_on_pins`
`update_power_domain`	`-internal_power_net`	`-primary_power_net`
	`-internal_ground_net`	`-primary_ground_net`
`update_power_switch_rule`	`-acknowledge_receiver`	`-acknowledge_receiver_`*`x`*
`update_state_retention_rules`	`-cell`	`-cells`

Note: For backward-compatibility, the old names are also supported.

Changed Command Option Value

Command	Option	Old Value	New Value
`create_ground_nets`	`-voltage`	*`string`*	`{`*`float`*`|`*`voltage_range`*`}`
`create_isolation_rule`	`-isolation_output`	`{high|low| hold}`	`{high|low|hold|tristate}`
`create_mode_transition`	`-cycles`	*`integer`*	`[`*`integer`*`:]`*`integer`*
	`-latency`	*`float`*	`[`*`float`*`:]`*`float`*
`create_power_nets`	`-voltage`	*`string`*	`{`*`float`*`|`*`voltage_range`*`}`
`define_isolation_cell`	`-valid_location`	`{from|to}`	`{from|to|on|off}`
`define_level_shifter_cell`	`-valid_location`	`{to|from}`	`{to|from|either}`
`define_library_set`	`-libraries`	`library file list`	`list of library files and library groups`
`update_power_switch_rule`	`-acknowledge_receiver_x`	*`pin`*	*`expression`*

New Commands

```
assert_illegal_domain_configurations
create_assertion_control
define_related_power_pins
end_macro_model
end_power_mode_control_group
get_parameter
identify_secondary_domain
include
set_equivalent_control_pins
set_floating_ports
set_input_voltage_tolerance
set_macro_model
set_power_mode_control_group
set_wire_feedthrough_ports
```

New Command Options

Command	**New Option/Argument**
create_analysis_view	-group_views
	-user_attributes
create_ground_nets	-external_shutoff_condition
create_isolation_rule	-no_condition
	-secondary_domain
create_nominal_condition	-ground_voltage
	-state
create_operating_corner	-ground_voltage
	-nmos_bias_voltage
	-pmos_bias_voltage

Command	New Option/Argument
create_power_domain	-active_state_conditions
	-base_domains
	-default_isolation_condition
	-default_restore_level
	-default_save_level
	-external_controlled_shutoff
create_power_mode	-group_modes
create_state_retention_rule	-exclude
	-restore_level
	-restore_precondition
	-save_level
	-save_precondition
	-secondary_domain
	-target_type
define_isolation_cell	-no_enable
define_level_shifter_cell	-enable
	-ground_input_voltage_range
	-ground_output_voltage_range
	-input_ground_pin
	-output_ground_pin
	-power
define_power_switch_cell	-enable_pin_bias
	-gate_bias_pin
	-stage_x_on_resistance
define_state_retention_cell	-always_on_components
	-cell_type
end_design	*module*
end_macro_model	*macro_cell*
identify_power_logic	-module

Command	New Option/Argument
set_design	-honor_boundary_port_domain
	-parameters
set_instance	-design
	-domain_mapping
	-model
	-parameter_mapping
update_isolation_rules	-within_hierarchy
update_level_shifter_rules	-within_hierarchy
update_power_domain	-equivalent_power_nets
	-equivalent_ground_nets
	-transition_cycles
	-transition_latency
	-transition_slope
update_power_mode	-hold_sdc_files
	-setup_sdc_files
update_power_switch_rule	-gate_bias_net
update_state_retention_rules	-set_reset_control

Syntax Changes

Command	New Option
create_isolation_rule	`-pins` option must be specified with `-from`, `-to` or both options
	`-isolation_condition` option is now optional
create_level_shifter_rule	`-pins` option must be specified with `-from`, `-to` or both options
create_power_domain	options `-instances`, `-boundary_ports` and `-default` are no longer exclusive
	modified syntax for `-default_save_xx` and `-default_restore_xx` options

Command	**New Option**
`create_state_retention_rule`	modified syntax for the `-restore_edge and -save_edge` options
`define_always_on_cell`	allow specification of combination of `-power_switchable` and `-ground_switchable` options
`define_level_shifter_cell`	made step in voltage range optional: *`lower_bound:upper_bound`*`[`*`:step`*`]`
`define_state_retention_cell`	allow specification of combination of `-power_switchable` and `-ground_switchable` options
	modified syntax for `-save_function` and `-restore_function` options