The UPF 2.1 library commands: Truly unifying the power specification formats

Amit Srivastava, Mentor Graphics (amit_srivastava@mentor.com)
Awashesh Kumar, Mentor Graphics (awashesh_kumar@mentor.com)
Vinay Singh, Mentor Graphics (vinay_singh@mentor.com)

***Abstract*- Today's low power SoCs involve a variety of IPs including hard/soft macros and complex power management architecture described by IEEE 1801 UPF. This architecture is implemented by special power management cells like isolation/level shifter/retention, etc. The earlier versions of UPF described commands for capturing the power intent but not the library cells or hard macros. The designers have to rely on other standards like Liberty formats to completely capture the details of such cells. The UPF 2.1 (latest version) has introduced new commands to capture the intent of special power management cells and the hard macros. In this paper, we take a deeper look at library commands in UPF 2.1 and analyze their merits over Liberty counterparts. We will demonstrate, using relevant examples, how the user can achieve a truly UPF based flow for capturing the power intent.**

I. INTRODUCTION

A low power SoC design may have a variety of IPs including hard/soft macros with sophisticated power management architecture. The power management architecture is typically described in a side file written in IEEE 1801 UPF. It gets implemented in the design via special power management cells, like isolation, level shifter, retention, power switches, etc. Also, some of the IPs have power management architecture which is pre-implemented. For proper integration of those IPs, it is important that the SoC power intent understands the details of power management at the interface of the IP. The earlier versions of UPF described commands for capturing the power intent for the design. However, they relied on external formats (Liberty) to describe the power management related information for the pre-implemented IPs and special power management cells. Sometimes, there was no information available for the hard IPs causing loss of information leading to functional bugs related to integration of the IPs. This dependence on third party standards to capture power aware information results in incomplete information being captured or the information is incompatible with power intent captured in UPF. UPF 2.1 has introduced new commands to capture power intent of hard macros and power management cells. It has unified the design intent commands with library commands to provide a single standard format for capturing complete power aware information

In a UPF based environment, the cells are typically treated as a blackbox where default UPF based semantics don't apply. Hence, the surrounding environment is ignorant of the power management capability of those cells. This causes problems during integration and hence affects both verification and implementation flow. A typical set of information comprises of the details about pg_pins(supplies), related_supplies of the non-pgpins, operating voltages of the supplies, etc. To fulfill the requirement, such information is captured in Liberty files. This creates a dependency on Liberty processing in a UPF based flow causing a burden on both users and tools. The lack of well-defined semantics of low power attributes in Liberty complicates the situation further, often causing ambiguity in tool behavior. Moreover, Liberty lacks the support for more advanced power intent capability like power states that form the basis of power management.

Liberty format, though widely used is a separate standard which evolves at a different pace than UPF standard. Sometimes the Liberty information might not be available in early stages of development of the hard IP. Liberty files may not contain the correct power aware information. Hence, the verification engineer may need to fix it in the library. The verification engineer may not be able to change the Liberty file to add/remove attributes needed for more complex power aware verification.

UPF 2.1 has taken leaps in providing new commands to empower users to capture the details about library cells in UPF itself. Being part of the UPF standard, they have a benefit of much tighter integration with the original UPF

commands and also have well defined semantics that will result in consistent behavior across tools. They also simplify the migration from a CPF based environment to UPF based environment.

In this paper, we will describe how to use these commands effectively and provide guidelines to capture power intent for hard IPs. We will also provide their mapping with Liberty commands for easy migration from a Liberty based flow to the truly unified UPF based flow.

II. Need For Library Commands

A typical SoC design is comprised of instantiations of various IPs which are hardened with power aware information. Along with hard IPs, there may be various power management cells present or needed to be inserted in the design to implement the power management architecture. These cells perform specific power management functionality like isolation, level shifting, retention, etc. Tools need the power aware information of the hard IPs and power management cells for implementation and verification. The SoC that integrates the hard IP requires information at the interface of the IP e.g.

- The top level supplies which need to be connected to the IP.
- The top level power domain which contains the definition of power states for the IP.
- The different power states of the IP, including legal/illegal states.
- The driving and receiving supplies of the data pins that are at the interface of the IP.
- Any power management cells that are inserted at the boundary of the IP and hence should not be re-inserted.

All this information becomes critical in proper integration of the IP in the parent environment and also enabling tools to ensure the constraints of the IP are validated after the integration.

Apart from the hard IPs, it is also important for tools to be aware of the power related characteristics of special cells that are inserted. These cells have some special attributes on them which are relevant for power management.

- Some of these cells may contain multiple power rails, the knowledge of which plays a role in the appropriate placement of these cells. For example, a multi-rail isolation cell has two power and ground supplies, out of which one is a switchable supply whereas the other is a non-switchable/backup supply. These kinds of isolation cells can only be placed in a region where both the supply rails are available. In contrast, there are single rail clamp cells which also perform the function of isolation. Such cells can be placed at locations where a relevant supply isn't switched off.
- Sometimes there may be different flavors of power management cells available for a specific purpose. The choice often affects the area and protocol requirements. Hence, it is important to capture such requirements so that users can express their requirements during the implementation process. For example, state retention cells can have two different structures, a slave-alive or balloon latch structure. In a slave-alive retention cell, the state of the cell is stored in the slave latch when the primary power of master-slave register goes down. In the balloon-latch structure, the state of the cell is stored in an auxiliary latch (shadow register) when primary power of master-slave register goes down.
- In some legacy designs, the power management cells are hand instantiated, in such cases, the verification and implementation tools need to identify them so that they can avoid re-inserting new cells in the design that perform the similar functionality. The power aware information captures the details about the characteristics of these cells required for automatic detection as power management cells.

III. Dependence of Earlier UPF on other Formats

One of the primary requirements for capturing power intent for library cells is the ability to capture information for a model (that describes that cell) that gets applied to all its instantiations in the given design. In UPF 2.0, the power intent commands are design or instance specific. There was no automatic way to specify a set of commands for a model which can be applied to all the instances of that model. Along with that, the UPF commands are directive in nature. This implies that the implementation/verification tools will create power management architecture resulting from the application of the UPF commands, for example, a set_isolation command in UPF will result in the placement of isolation cells at the specific instance in the design. If users are using the commands to model the power management that is already implemented, then they have to rely on ad-hoc mechanisms to disable the semantics of application of UPF commands. There is a command in UPF, set_simstate_behavior, which users can use to disable such semantics for the hard IP model.

UPF 2.0 does provide some attributes to capture the details of library cells. The *set_design_attributes* and *set_port_attribute* command have a *-model* option which allows adding attributes to all instances of the model.

However, it only defines a small set of predefined attributes which are insufficient to capture complete details of the power management cells.

As a result of these limitations users have used Liberty format, which is already a popular format for capturing library related details. Liberty has pin level attributes to specify the related_supplies and pg_type of pins. It also provides information about the pins that already have isolation placed on them. For example, the attribute "*is_isolated*" on a pin specifies that this pin already has isolation. The cells in Liberty files also contain information about the supply pins that are present at the cell interface. This information is necessary to make the proper supply connections to those cells.

IV. UPF 2.1 Power Model and Power Management Cell Commands

UPF 2.1 addressed the limitations of earlier version of UPF by introducing two sets of commands to capture power intent for hard IPs and power management cells. There is one set of commands that apply to power models. A power model contains other UPF commands and it is used to document the power intent of a hard IP. Since the hard IP has power management architecture already implemented in the IP, the power model describes that architecture at the interface of the IP using UPF commands.

There is another set of commands, referred as "define" commands, for capturing power intent for power management cells such as isolation, level-shifter and retention cells. These new commands remove the dependency on Liberty files for capturing information about power management cells.

A. Power model commands

There are 3 commands added in UPF 2.1 for power models. They are *begin_power_model*, *end_power_model* and *apply_power_model*.

1. begin_power_model/end_power_model

These commands create a named boundary for hard IP power intent in which the power intent of a hard IP will be described using other UPF commands. The commands inside the power model have an additional semantic that they will not be re-implemented and no new logic or design objects shall be inferred within the cell instances targeted by a power model.

begin_power_model should be followed by the *power_model_name* which is the name of the power model being defined. Another option provided is *–for*, which should be followed by the names of the hard IP or macro cells to which this power model applies. It indicates that the power model represents the power intent for a family of model definitions, specifically *–for* allows a single power model to apply on more than one cell. When *–for* is not specified, the *power_model_name* shall also be a valid macro cell name. Use of this option facilitates better error checking in cases where the user mistakenly applies the wrong power model to an instance of a cell.

A power model defined with *begin_power_model* is terminated by the first subsequent occurrence of *end_power_model* in the same UPF file.

UPF code:

```
#---------------
#hard_ip.upf
#---------------
# Power Model for hardMacro for cell hard_ip
begin_power_model hardMacro –for {hard_ip}
       #Power Domains for hard_ip
       create_power_domain pd_hardIP \
       -include_scope \
       -supply { backup_ssh } \
       -supply { primary }

       #Related Supply Constraints
       set_port_attributes
       -model hard_ip \
       -applies_to outputs \
       -driver_supply pd_hardIP.primary

       set_port_attributes
       -ports portA \
       -driver_supply pd_hardIP.backup_ssh
```

```
    set_port_attributes
    - model hard_ip \
    -applies_to inputs \
    -receiver_supply pd_hardIP.primary

    #Retention Constraints
    set_retention_elements critical_regs \
    -elements { reg_a reg_b } \
    -retention_purpose required

    #Internal switchable supply
    create_power_switch

    # Isolation/level shifter and retention cells
    set_isolation ...
    set_level_shifter ...
    set_retention ...

    # System states for hard IP
    add_power_state pd_hardIP ...

    # ... Other Constraints ...
end_power_model
```

2. apply_power_model

As *begin_power_model* and *end_power_model* commands provide capability to define the power model, similarly *apply_power_model* provides capability for application of that power model in user design. It describes the connections of the interface supply set handles of a previously loaded power model with the supply sets in the scope where the corresponding macro cells are instantiated.

Use of this command facilitates better error checking in cases where arguments to the *–supply_map* options are such that the implied *associate_supply_set* commands are not legal. It also facilitates better error checking in cases when *apply_power_model* is used with *–elements* argument and underlying cell name of each instance does not match the corresponding macro cell name specified in the *-for* option of *begin_power_model* or the *power_model_name* when the *-for* option (of *begin_power_model*) is not specified.

UPF code:

```
#---------------------
#soc.upf
#---------------------
# Load the Power Models.
load_upf hard_ip.upf
# Apply the Power model for hard Macro
# and connect supplies
apply_power_model hardMacro \
-supply_map {{ pd_hardIP.primary pd_SoC.primary } \
{ pd_hardIP.backup_ssh pd_SoC.backup_ssh } \
}

# Update Supply Constraints
add_power_state pd_SoC.primary ..

# Connect Logic Controls
connect_logic_net ...
#---------------
```

3. Verification impact of power model commands

Power model commands also assist in the verification of the power management architecture.

- Verification tools can check for consistency during the integration process and ensure the IP integrator is integrating the IP with appropriate supplies that adhere to the power management implemented within the hard IP. For example,
    - An incorrect supply gets connected to the IP which operates at voltage levels that are beyond the normal operations for the IP.
    - If clamp value requirement is specified on some hard IP input port 'P' using UPF command
      `set_port_attributes -ports {P} -clamp_value 1`
      Then the verification tool may do a dynamic checking if the clamp value requirement of port P is not met when the driving power domain goes off.
- It may provide simulation behavior for non-power aware simulation models, For example, using the commands present in the power model, tools can do pin-level corruption to mimic the power down behavior.

B. Power Management Cell commands

1) The *define_** commands:

UPF 2.1 introduced a set of *define_** (starting with prefix *define_*) commands which can be used to identify the power management cells present in the library and define the characteristics of these cells. These commands allow user to capture details about the pins and power management functionality with the help of various options. These commands capture the following key details about the power management cell.

- Identifies a power management cell. The name of the command specifies the type of cell being defined. For example, the *define_isolation_cell* command identifies the isolation cell.
- Identifies the supply pins of the power management cell. For example, for an always_on cell the options *–power/-ground* specify the power and ground pins of the cell respectively. Whereas the options *-power_switchable/-ground_switchable* specify the power and ground pin connected to a switchable power supply.
- Identifies the control pin of the power management cell. For example, the *define_isolation_cell –enable* option specifies the enable pin of the isolation cell.
- In addition to pin level details, the power management cell commands also specify expressions in terms of those pins which can be used to specify additional conditions. E.g. *–save_check/-restore_check* options of the *define_retention_cell* command specifies the expression defined as a function of input pins of the cell to specify the condition which must be true for state of sequential elements to be restored.

UPF 2.1 provides following *define_** commands:

- *define_always_on_cell*: Identifies always-on cells
- *define_diode_clamp* : Identifies the diode clamp cells
- *define_isoaltion_cell* : Identifies isolation cells
- *define_level_shifter_cell* : Identifies level-shifter cells
- *define_power_switch_cell* : Identifies power-switch cells
- *define_retention_cell* : Identifies state retention cells

Power management cell commands do not alter the existing library cell definitions. Their semantics play a role only when they are used with design power intent commands for a given design under consideration. These commands should be processed after the cell libraries are loaded and before any other power intent command is processed [2].

Example:

The example given below models a retention cell that saves the current value when the save control signal is activated while the power is on. It retains the saved value when the power is off, and restores the saved value when the power is turned on and the restore control is de-activated.

```
define_retetion_cell –cells my_ret_cell \
       -cell_type retcell \
       -power VDD \
       -ground VSS \
       -power_switchable VDD_SW \
       -save_action {RET posedge} \
       -restore_action {RET negedge} \
       …
```

UPF retention strategy to capture the above mentioned low power intent is:

```
set_retention RET \
       -domain PD \
       -save_signal {save posedge} \
       -restore_signal {save negedge}
       …
```

Then the retention cell "my_ret_cell" defined above can be used to implement the retention strategy RET.

Similar modelling of other types of power management cells like isolation cells, level-shifter cells, power switch cells, always_on cells and diode_clamp cells can be done using power management cell commands. This is explained with examples in Annex I of UPF 2.1 LRM [2]

2) Verification impact of power management cell commands

The *define_** commands are mainly intended for implementation tools that insert the power management architecture. However, the verification tools can still leverage the additional information specified by these commands to do some additional checking. One such check is to ensure the verification model matches with the power management cell requirements as shown by the following example.

Example 1:

```
define_level_shifter_cell –cells {LS_CELL} \
       -direction low_to_high \
       …
set_level_shifter LS1 \
       -rule high_to_low \
       …
use_interface_cell my_interface \
       -strategy LS1 \
       -lib_cells LS_CELL \
       -force_function
       …
```

In this example a level shifter cell defined with *–direction low_to_high* is used to provide functional model for level shifter strategy which has *-rule high_to_low*. So the cell LS_CELL cannot be used for functional verification of strategy LS1 and the tool may warn user about such incorrect usage.

Example 2:

Tool may check wrong cell instance specified as instantiated cell for a UPF strategy. Following command defines a cell which required net connected with clock pin CK to be low for successful save and restore.

```
define_retention_cell –cells RET_CELL \
                -clock_pin CK
                -save_check {!CK}
                -restore_check {!CK}
                …
```

The cell RET_CELL is used as instantiated cell for retention strategy RET

```
set_retention RET \
                -instance RET_CELL
                -save_condition {!clk} \
                -restore_condition {!clk}
                …
```

Tool may do a static checking to determine if the expression specified in set_retention –save_condition/-restore_condition is logically same as the expressions used with define_retention_cell –save_check/-restore_check.

V. PROS & CONS OF USING LIBRARY & POWER MODEL COMMANDS IN UPF

There are some inherent benefits of using the library and power model commands in UPF. Some of these include:

- Better compatibility with other UPF commands as the commands are part of the UPF LRM.
- More flexibility in capturing key features, such as power states, that allows a more complete description and exhaustive verification of the IP block. Although, the latest Liberty version does provide a mode_definition group to declare mode values, its usage is not clear in Liberty and it not widely used in the industry.

- Concise representation of power related information. The Liberty description may contain numerous other attributes that are not relevant for power management, e.g. timing, etc. These unnecessarily complicate the processing of Liberty files to extract just the relevant power aware information. On the other hand, UPF power models and *define_** commands provide a more concise and relevant representation of power management information.
- Lack of well-defined semantics in Liberty causes different interpretations.
- Sometimes, the availability of Liberty files may not be there at the early phases in the verification cycle. This causes missing information related to power models and power management cells.
- Liberty may contain incorrect information about power management which is difficult to correct without affecting other parts of the environment. This can easily be achieved with library commands in UPF as they are applicable to the specific design in consideration.

However, there is some downside of using library commands.

- Being relatively new, there are still not many tools that support them.
- There are a lot of legacy libraries which have power management information present in them. The user may be hesitant to replicate information about those in UPF commands.

VI. Migrating To UPF 2.1 Library commands

Because there are many legacy libraries that contain the power management information in them, in order to achieve the benefits of the new library commands the user is constantly faced with a dilemma whether to migrate to new UPF commands or continue using Liberty for such information. However, if the user decides to use the new commands, it is pretty easy and straightforward to achieve that. The following section describes some of the easy ways to migrate from different formats to the new UPF 2.1 commands.

A. Migrating from UPF 2.0 to UPF *2.1*

The paper on "Hierarchical UPF Methodology", DVCon 2012,[3] demonstrates a methodology which can be used to capture the power intent for a hard-IP using UPF 2.0 commands. It describes the concept of interface UPF which is present in a separate UPF file. If users are using a similar methodology and describing the information in UPF 2.0 syntax, then it is straight-forward to migrate to UPF 2.1 power models. They just need to encapsulate the interface UPF within a *begin_power_model* and *end_power_model* wrapper.

Also, if users have used UPF to develop the IP which will be hardened and shipped, they can reuse the existing UPF commands and encapsulate it within the power model commands. However, they have to do some minor alterations to select the set of commands that are relevant for the interface of the IP and avoid referring to internal scopes which may not be present in the hardened version of the IP.

However, users need to be aware of commands/options that have been marked as deprecated in UPF 2.1. The standard already provides mapping to corresponding UPF 2.1 commands. Ref: IEEE 1801-2013 UPF – Annex-D – D.2[2].

B. Migrating from Liberty to UPF 2.1

Often the information about hard IPs and power management Cells are present in Liberty files. The UPF 2.1 LRM clearly provides the mapping of Liberty attributes and corresponding UPF 2.1 commands. Users can use that reference to convert the power management information present in Liberty to UPF 2.1.

Table 1 presents an example described in Liberty and the UPF 2.1.

TABLE 1
MAPPING OF POWER INTENT SPECIFIED IN LIBERTY FILE WITH CORRESPONDING UPF 2.1 COMMANDS

Power intent describes in Liberty file	Equivalent UPF 2.1 commands file

```
Liberty_file.lib

library(ANALOG_LIB) {
  cell (analog) {
    is_macro_cell : true;
    pg_pin(AGND) {
      pg_type : "primary_ground";
    }
    pg_pin(AVDD) {
      pg_type : "primary_power";
    }
    pin (in) {
      direction : input;
      related_power_pin  : "AVDD";
      related_ground_pin : "AGND";
    }
    pin (out) {
      direction : output;
      power_down_function : "!AVDD +
AGND";
      related_power_pin  : "AVDD";
      related_ground_pin : "AGND";
    }
  }
}
```

```
#---------------
#upf_macro.upf
#---------------
# Power Model for analog

begin_power_model upf_macro -for {analog}
#Power Domains for analog
create_power_domain pd_analog \
-elements {analog} \
-supply { SS }

create_supply_set pd_analog.SS \
-function { power AVDD } \
-function { ground AGND } -update

set_port_attributes -model analog -ports AVDD \
-pg_type primary_power

set_port_attributes -model analog -ports AGND \
-pg_type primary_ground
add_power_state pd_analog.primary -supply\
-state {ON -simstate NORMAL \
-supply_expr {power == { FULL_ON 0.8} && ground == {
FULL_ON 0}}}\
-state {OFF -simstate CORRUPT \
-supply_expr {power == OFF || ground == OFF}}

set_port_attributes -model analog -ports {in out} \
-related_power_port AVDD \
-related_ground_port AGND

end_power_model
```

Table 2 presents a mapping of Liberty cell which has been modelled by UPF 2.1 power management cell commands.

TABLE 2
MODELLING OF LIBERTY STATE RETENTION CELL USING POWER MANAGEMENT CELL COMMAND

Liberty cell definition for retntion cell	Modelling using Power Management cell command

`cell (balloon_ret_cell) {`
`retention_cell : retdiff;`
`area : 1.0 ;`
`pg_pin(VDD) {`
`  voltage_name : VDD;`
`  pg_type : primary_power;`
`}`
`pg_pin(VSS) {`
`  voltage_name : VSS;`
`  pg_type : primary_ground;`
`}`
`pg_pin(VDDB) {`
`  voltage_name : VDDB;`
`  pg_type : backup_power;`
`}`
`pin(RESTORE) {`
`  direction : input;`
`  related_power_pin : VDDB;`
`  related_ground_pin : VSS;`
`  retention_pin(restore, "0");`
`  restore_action : "H";`
`  restore_condition : "!CK";`
`  restore_edge_type : "leading";`
`}`
`pin(SAVE) {`
	`define_retetion_cell -cells my_ret_cell \`
	`    -cell_type retcell \`
	`    -power VDD \`
	`    -ground VSS \`
	`    -power_switchable VDDB \`
	`    -ground_switchable VSSB \`
	`    -clock_pin CK \`
	`    -save_action {SAVE high} \`
	`    -restore_action {RESTORE high} \`
	`    -save_check !CK \`
	`    -restore_check !CK`

```
  direction : input;
  related_power_pin : VDDB;
  related_ground_pin : VSS;
  retention_pin(save, "0");
  save_action : "H";
  save_condition : "!CK";
}
```

C. *Migrating from CPF to UPF 2.1*

The UPF 2.1 library commands are inspired from the corresponding CPF commands. Therefore, they share a lot of similarities with respect to information being captured. Migration from CPF becomes much easier if one understands the corresponding mappings. The CPF commands *set_macro_model* and *end_macro_model* are used to pack the power intent specification of a hard macro. UPF 2.1 allows similar packing with *begin_power_model* and *end_power_model* commands. Similarly, library commands in CPF can be mapped to power management cell commands in UPF. Table 3 presents the mapping of CPF commands with corresponding UPF commands.

TABLE 3
MAPPING OF CPF COMMANDS WITH CORRESPONDING UPF COMMANDS

CPF command	Corresponding UPF command
set_macro_model	begin_power_model
end_macro_model	end_power_model
define_always_on_cell	define_always_on_cell
define_power_clamp_cell	define_diode_clamp
define_isolation_cell	define_isolation_cell
define_level_shifter_cell	define_level_shifter_cell
define_power_switch_cell	define_power_switch_cell
define_state_retention_cell	define_retention_cell

VI. CONCLUSION

The Library and power model commands in UPF have bridged the gap in UPF for capturing the power management of library cells. This truly unifies the UPF and provides a comprehensive standard for expressing the power management of low power designs. The tighter integration of library commands with power intent commands and more complete semantics ensure that users get more flexibility and reuse capability in capturing the information related to power management. This will result in much more comprehensive verification at earlier phases thereby reducing the costly respins.

REFERENCES

[1] IEEE Std 1801™-2009 for Design and Verification of Low Power Integrated Circuits. IEEE Computer Society, 27 March 2009.

[2] IEEE Std 1801™-2013 for Design and Verification of Low Power Integrated Circuits. IEEE Computer Society, 29 May 2013.

[3] Amit Srivastava, Rudra Mukherjee, Erich Marschner, Chuck Seeley and Sorin Dobre : "Low Power SoC Verification: IP Reuse and Hierarchical Composition using UPF", DVCon 2012.