The UPF 2.1 Library Commands: Truly Unifying the Power Specification Formats

Amit Srivastava, Awashesh Kumar, Vinay Singh

Mentor Graphics Corp. 8005 SW Boeckman Rd. Wilsonville, OR 97070

Abstract

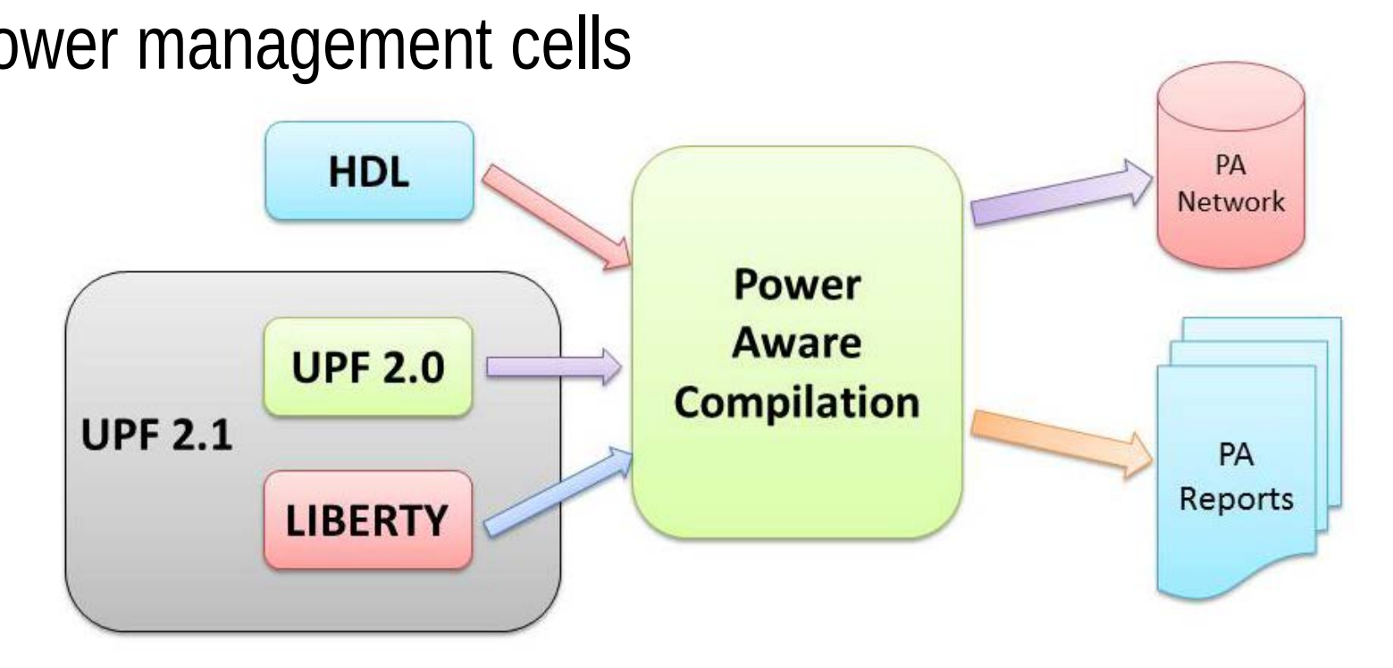
Today's low power SoCs involve a variety of IPs including hard/soft macros and complex power management architecture described by IEEE 1801 UPF. This architecture is implemented by special power management cells like isolation/level shifter/retention, etc. The earlier versions of UPF described commands 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.

Why Library Commands ?

- IPs have power management architecture which is pre-implemented.
- It is important to know power management details at the interface of IP for proper integration in SoC environment
- A typical set of information comprises of the details about pg_pins(supplies), related_supplies of the non-pgpins, operating voltages of the supplies, information about power management cells etc.
- UPF 2.0 relied on external formats(Liberty) to describe power management information of pre-implemented IPs
- The lack of well-defined semantics of low power attributes in Liberty complicates the situation, often causing ambiguity in tool behavior.

UPF 2.1 Library 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



Power Model Commands

`begin_power_model, end_power_model:`

These commands define a power model using other UPF commands. UPF commands can be used to capture a number of liberty attributes-

- power/ground pg_pins of IP
- related_power_port/related_ground_port of the IP pins
- power_down_function of IP pins

Example Syntax:

```
  begin_power_model upf_model
    ... UPF Commands ...
  end_power_model
```

`apply_power_model:`

Instantiates the power model and allows connection of supplies via supply set handles.

Example Syntax:

```
  apply_power_model upf_model -elements I1\
    -supply_map {{PD.ssh1 ss1} {PD.ssh2 ss2}}
```

Power Management Cell Commands

Identifies the power management cells present in the library and define the characteristics of those cells. Captures following key details about the power management cell.

- Identifies a power management 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.
- 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

UPF 2.1 provides following define_* commands:

define_always_on_cell: Identifies always-on cells
define_diode_clamp : Identifies the diode clamp cells
define_isolation_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

Migration from Liberty to UPF 2.1 Library Commands

The following table shows how a hard macro modeled in liberty can be expressed using UPF 2.1 commands

Power intent described in Liberty file

```
library (macro_switch) {
...
Voltage_map (PVDD, 1.0);
Voltage_map (PVVDD, 1.0);
Voltage_map (PVSS, 0.0);

cell(MACRO) {
is_macro_cell : true;
switch_cell_type :
fine_grain;
...
pg_pin(PVDD) {
voltage_name : PVDD;
pg_type : primary_power;
direction : input;
}
pg_pin(PVSS) {
voltage_name : PVSS;
pg_type : primary_ground;
direction : input;
}
pg_pin(PVVDD) {
voltage_name : PVVDD;
pg_type : internal_power;
direction : internal;
switch_function : "CTRL";
pg_function : "PVDD";
}
pin(CTRL) {
direction : input;
switch_pin : true;
related_power_pin : PVDD;
related_ground_pin : PVSS;
...
}
pin(A) {
direction : input;
is_isolated : true;
isolation_enable_condition :
"en";
related_power_pin : PVVDD;
related_ground_pin : PVSS;
...
}
pin(Z) {
direction : output;
power_down_function : "!PVDD
+ PVSS";
related_power_pin : PVVDD;
related_ground_pin : PVSS;
...
}
}
```

Equivalent UPF 2.1 commands file

```
begin_power_model MACRO

create_power_domain pd_MACRO \
-supply {SS_PVDD} –supply {SS_PVVDD}

create_supply_set pd_MACRO.SS_PVDD \
-function { power PVDD } \
-function { ground PVSS } –update

create_supply_set pd_MACRO.SS_PVVDD \
-function { power PVVDD } \
-function { ground PVSS } –update

create_power_switch internal_sw \
-output_supply_port {out PVVDD} \
-input_supply_port {in PVDD} \
-control_port {ctrl CTRL} \
-on_state { ON in !ctrl }

set_port_attributes -ports {CTRL} \
-receiver_supply PD.SS_PVDD
set_port_attributes -ports {A} \
-receiver_supply PD.SS_PVVDD
set_port_attributes -ports {Z} \
-driver_supply PD.SS_PVVDD

add_power_state PD.SS_PVDD –supply \
-state {ON -simstate NORMAL \
-supply_expr {power == {FULL_ON 1.0} &&
ground == {FULL_ON 0.0}}}\
-state {OFF -simstate CORRUPT \
-supply_expr {power == OFF || ground ==
OFF}}

add_power_state PD.SS_PVVDD –supply \
-state {ON -simstate NORMAL \
-logic_expr {!CTRL} \
-supply_expr {power == {FULL_ON 1.0} &&
ground == {FULL_ON 0.0}}} \
-state {OFF -simstate CORRUPT \
-logic_expr {CTRL} \
-supply_expr {power == OFF || ground ==
OFF}}

set_isolation internal_iso \
-domain pd_MACRO -elements {A} \
-isolation_signal {en} \
-isolation_sense {high}

set_port_attributes -ports {A Z} \
-clamp_value 1

end_power_model
```

The following table shows how a retention cell modeled in liberty can be expressed using UPF 2.1 library command

Liberty cell definition for retention cell

```
cell (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) {
  related_power_pin : VDDB;
  related_ground_pin : VSS;
  retention_pin(restore, "0");
  restore_action : "H";
  restore_condition : "!CK";
  restore_edge_type : "leading";
}
pin(SAVE) {
  related_power_pin : VDDB;
  related_ground_pin : VSS;
  retention_pin(save, "0");
  save_action : "H";
  save_condition : "!CK";
}
...
```

Modelling using Power Management cell command

```
define_retetion_cell \
    –cells ret_cell \
    -cell_type retdiff \
    -power VDDB \
    -ground VSS \
    -power_switchable VDD \
    -save_function {SAVE high} \
    -restore_function {RESTORE high} \
    -save_check !CK \
    -restore_check !CK
```

Migration from CPF to UPF 2.1 Library Commands

Migration from CPF to UPF 2.1 is simplified if one understands the direct mapping with Library commands as shown below:

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

Verification Impact of UPF 2.1 Library Commands

Power model commands:

- Check for consistency during IP integration process
  - Detect when an incorrect supply is connected to IP which operates at voltage levels beyond the normal operation of IP
  - Clamp value of isolation cell matches with constraints specified in IP, using set_port_attributes command
- May provide power aware simulation behavior for non-PA simulation models by applying simstate semantics on the pins of the IP as per power state description

Power management cell commands:

- Check to ensure the verification model matches the power management cell requirements.

Example:

```
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
```

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.

Example:

```
 define_retention_cell –cells RET_CELL \
                -clock_pin CK
                -save_check {!CK}
                -restore_check {!CK}
 set_retention RET \
                -instance RET_CELL
                -save_condition {!clk} \
                -restore_condition {!clk}
```

Tool may do some static/dynamic 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*

Pros & Cons of Using UPF 2.1 Library Commands

Pros:

- Better compatibility with other UPF commands.
- Flexibility in capturing key features, such as power states.
- Concise representation of power related information of power management cells using *define_** commands
- Lack of well-defined semantics in Liberty causes different interpretations.
- Liberty files may not be available at the early phases in the verification cycle.

Cons:

- Relatively new, not many tools support them.
- Legacy libraries present in user flows.

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 re-spins.

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.