Power State to PST Conversion: Simplifying static analysis and debugging of power aware designs

Madhur Bhargava, Pankaj Gairola
Mentor Graphics Corp. 8005 SW Boeckman Rd. Wilsonville, OR 97070

Introduction

- **Electronic devices have become complex and energy aware**
  - Chips have multiple power domains each having multiple operating power modes and dynamically changing voltage levels.
  - Require sophisticated power management architectures
  - Complex protocols and power modes need to be verified
- **PST based verification**
  - Accellera UPF provided commands to define the possible values of supply ports ("port states") used to deliver power to a system, together with "power state tables" (or PSTs) that defined legal combinations of port states
  - Verification tools and engineers relied on PST-based analysis of the possible combinations of power supply values to determine where isolation and level shifting would be required in a given design
  - Requires implementation of power management in detail before verification could start, tending to delay power aware verification
- **Use of Power States & Supply Sets**
  - To address above issues, IEEE Std 1801 UPF added a number of new concepts such **as supply-sets and power-states** that enables users to construct complex power state definitions sufficient to model various possible situations. Define power states on a more abstract level of supply sets/power domains with the help of add_power_state command
  - Extremely flexible and powerful command that enables the user to construct very complex power state definitions sufficient to model any possible situation. This flexibility and power causes complexities in understanding the supply dependencies between two power domains being evaluated for static analysis.
  - As power states are composed of not only supply information but also the logic information, it becomes more difficult to get the overall supply dependencies between two power domains for static analysis.

✓ Highlight the power aware verification challenges involved for a design having power states defined using add_power_state command.

✓ Demonstrate an approach to simplify the process of static analysis and debugging for such designs.

✓ Guidelines to define power state definitions adhering to which can help ease the verification process.

Power Aware – Basic Concepts

Power State Table

- Defines the legal combinations of port states, i.e., **those combinations of port states that can exist at the same time during operation of the design.** PST based analysis of the possible combinations of power supply values enables UPF-supporting tools to **perform static analysis so as to determine the isolation and level shifting requirements of a given design**.

Supply Sets

- Represents the power provided to a power domain for a particular use, such as the primary supply of the domain, an isolation supply, or a retention supply. Supply sets are **an abstraction of connections to a power distribution network**, and they can be used to model incoming power to the domain before the supply distribution network has been defined.

Power States

- Represents a subset of this set of **all possible functional states of an object**, or equivalently, a region within the functional state space of the object. The defining expression of a power state evaluates to True for every value combination in this subset and evaluates to False for every value combination outside this subset. Power states are defined with the add_power_state command.

✓ **add_power_state** command is an extremely flexible and powerful command that enables the user to construct very complex power state definitions sufficient to model any possible situation.

✓ However, that flexibility and power, if used indiscriminately, can result in power state definitions that are difficult to understand, difficult to debug, and even difficult for tools to analyze

Static Analysis – Power Aware Verification

Catch all structural errors – correct placement and connections of PA Cells

Static analysis of the design for Level-shifter requirement:

- Level shifters are normally required for the signals crossing the power domain boundaries when source and sink supplies operate at different voltage levels when they are on

Identify the signals that are crossing the voltage island boundary

↓

Detect the operating voltages of the two islands. Using the information from the power state table, the tool can easily figure out the operating voltage of a particular domain. Depending on whether there is a difference in the operating voltage, then the particular signal is a potential candidate for a level shifter.

↓

Depending on whether 'a level shifter is required or not' and 'a corresponding level shifting strategy is specified or not' we can detect 'a valid', 'a missing' or a 'redundant' level-shifter. Further by comparing properties of specified and required level-shifter we can deduce 'incorrect' level-shifter

Static analysis of the design for Isolation requirement:

- Isolation cells are clamps that drive a particular value on the signal when the isolation-enable control is triggered. They are required to mask the floating values on the inputs that are driven by signals from the domain that is switched off.

Identify the signals that are crossing the voltage island boundary

↓

Determine whether the primary supply could be switched or not. This is done by tracing back the primary power and ground pins until they terminate at the port of a switch or a port of the domain boundary. Supplies that are switched can be matched with entries in a power state table to figure out whether they are switched at the same time or not.

↓

Depending on whether 'an isolation is required or not' and 'a corresponding isolation strategy is specified or not' we can detect 'a valid', 'a missing' or a 'redundant' isolation.

Challenges in static analysis & debugging of design with Power States

- With the usage of PSTs in UPF 1.0; **both the debugging and static analysis of design were relatively simpler tasks** as the power state definitions were defined on supply nets/ports, and also these definitions were represented in **tabular manner** which was easy to interpret.
- Defining power states on a more abstract level of supply sets/power domains with the help of **add_power_state** command
  - Extremely flexible and powerful - allow use of boolean operators in supply and logic expression
  - However, that flexibility often result in power state definitions that are difficult to understand, difficult to debug, and even difficult for tools to analyze

Static Analysis of the design with power states

- **Incremental refinement of power states**: One of the methodologies followed in low-power verification is to simulate and verify the design at various states of UPF implementation. Low-power design can be statically verified for isolation requirements even if the implementation UPF containing the definitions for supply nets/ports is not present. This verification was not possible with the help of PSTs as it required the actual supply nets/ports. However the analysis for isolation requirements requires analysis of simstates of driver/receiver logic.
- **Getting the inter-dependencies of supply states** : The power state of a system depends on the power states of its constituents IPs which are then sub-dependent on states of their supply sets. The power states of these supply sets are further dependent on state and voltage values of supply nets and ports. In PSTs based analysis, getting these dependencies was fairly straightforward as everything was available in tabular format. However with add_power_state command which allows usage of various Boolean operators, **getting the dependencies of supplies of two power domains is a tedious process**. This problem gets even more complex, when the power state definitions for driver/receiver logic do not have direct/indirect reference of each other, however may have a common signal in power states definitions.

Power State to PST Conversion

- Power state definitions using add_power_state commands are hierarchical in nature. Power state of a system > depends on power state of its constituent IPs > depends on power state of its supply sets > further dependent on state of supply functions. As a result, it can be seen that **every power state can be represented as combination of values of supply ports & control signals**.
  - ➢ The power objects (power domain, supply set, supply nets, control signals) are represented as column objects in the converted power state table.
  - ➢ The power states defined for this power object are represented as rows objects of converted power state table.
  - ➢ The power/supply sets state of the power objects are the entry values in the power state table.

Basic properties of this approach of power state to PST conversion:

- One power state table is constructed for one power object (supply set/power domain)
- At any point of time, the system will be in a defined state. That is at any point of time, **one of the rows of this power state table will evaluate to be true**.
- All the other power objects (Power Domain, Supply Sets, Control Ports/Nets, Supply Ports/Nets) referred in the power state definitions are represented as column objects in the converted power state table.
- One additional column of simState is added to the table, which represent the simstate of supply sets when the power object for which PST is created is a supply set. In case of power domain, this simState represent the simstate of primary supply set of power domain for which PST is created. **This simState helps in determining whether isolation is required during the static analysis of the low-power design**.
- A power state of a power object is represented as one or more rows in the converted power state table.
- The use of Boolean operator's is limited to "&&" and "||". The "&&" operator effectively means that either the values of rows will get refined or new column's will be added to the table. The "||" operator effectively means that splitting of rows will take place, and new rows will be added in the table.
- Don't cares may get added in the cell values. This represent that the power object can be in a defined particular state even if one of the dependent power object is in unknown (but defined) state.
- This approach is in accordance with successive refinement methodology where the power state table will also get refined based on the incremental updates in power state definitions.
- Consistency checks can be done at the time of conversion to catch issues in the power state definitions.
- The converted power state table can be analyzed at any point of time by the verification tools to do the static analysis for isolation/level shifting requirements. For isolation requirements the converted PSTs can be analyzed even before implemented UPF is available. This facilitates early verification of design before the supply network gets implemented.

Case Study 1

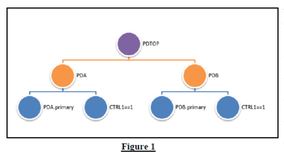

Figure 1

UPF Code

```
add_power_state PD_TOP -state TOP1 \
    { -logic_expr { PDA == A1 && PDB == B1 &&
PD_TOP.primary == TOPSS1} }
add_power_state PD_TOP.primary -state TOPSS1 \
    { -logic_expr { TOPPWR } -simstate NORMAL }
add_power_state PDA -state A1 \
    { -logic_expr { PDA.primary == ASS1 && CTRL } }
add_power_state PDB -state B1 \
    { -logic_expr { PDB.primary == BSS1 && CTRL } }
add_power_state PDA.primary -state ASS1 \
    { -simstate CORRUPT }
add_power_state PDB.primary -state BSS1 \
    { -simstate NORMAL }
```

➢ Even at this stage when implementation UPF is not present, the power state tabular representation can still be done. Verification tool can perform static analysis to determine the isolation requirements.

Converted PSTs

PDA.primary			PDB.primary	
PDA.primary	PDA.primary.SimState		PDB.primary	PDB.primary.SimState
ASS1	CORRUPT		BSS1	NORMAL

PDA			
PDA	PDA.primary	CTRL	PDA.primary.SimState
A1	ASS1	1	CORRUPT

PDB			
PDB	PDB.primary	CTRL	PDB.primary.SimState
B1	BSS1	1	NORMAL

PDTOP										
PDTOP	PDTOP .primary	TOPPWR	PDTOP .primary .simstate	PDA	PDA .primary	CTRL	PDA .primary .SimState	PDB	PDB .primary	PDB .primary .simstate
TOP1	TOPSS1	1	NORMAL	A1	ASS1	1	CORRUPT	B1	BSS1	NORMAL

✓ Once we have a PST based representation, verification tool can perform consistency error check to determine if the power state definitions are correct.

✓ In this case, as the CTRL signal is common in both the objects, it is added as one column. Tool checks if the value of CTRL is same for both CTRL (PDA) and CTRL (PDB) as the these objects points to same cell for TOP1 state.

Case Study 2

Consistency / Error Checks in power state definitions

```
#-----------------------------------
# Power state "refinement in place"
#-----------------------------------
add_power_state PDA -domain \
      -state {A1 -logic_expr {C1} }
add_power_state PDTOP -domain \
-state {TOP1 -logic_expr {PDA == A1 && C2} }

# The above is equivalent to
# add_power_state PDTOP -domain \
   -state {TOP1 -logic_expr {C1 && C2} }
#-----------------------------------
```

PDTOP	PDA	C1	C2
TOP1	A1	1	1

```
#--------------------------------
add_power_state PDA -domain -update -state
{A1 -logic_expr {!C2} }
# The above is equivalent to
# add_power_state PDA -domain -update -
state {A1 -logic_expr {C1 && !C2} }
# And it causes a ripple effect on PDTOP
# add_power_state PDTOP -domain \
# -state {TOP1 -logic_expr {C1 && C2 &&
!C2} } ; # <= contradiction
#--------------------------------
```

The last add_power_state command cases the contradiction and is error case.

➢ Such erroneous scenarios can be easily identified and presented to user with the approach of converting power states to PSTs.

➢ Here the row1 needs to be updated with the new value of C2. However as the new value is not in accordance with the previous value, tool will give out error message.

PDTOP	PDA	C1	C2
TOP1	A1	1	1 (old value) 0 (new value) -- error

Guidelines for modeling power states with add_power_state command

UPF command add_power_state which is used to define power states on supply sets and power domains is an extremely flexible and powerful command. It allows usage of boolean operators in supply and logic expression to define the relationship between two objects. Using random Boolean operators like xor, xnor in power state definitions can result in power state definitions that are difficult to understand, difficult to debug, and even difficult for tools to analyze.

- ➢ -logic_expr in power state definition of power domains should refer to control signals, power states of supply sets of this power domain and the power states of its descendants power domains.
- ➢ Avoid use of -supply_expr in power state definitions of power domains.
- ➢ -logic_expr in power state definition of supply sets should refer to control signals
- ➢ -supply_expr in power state definition of supply sets should refer to its supply functions
- ➢ In supply_expr and –logic_expr, try limiting the use of Boolean operators to "&&" and ==". Avoid use of "||" operator in –logic_expr as it can introduce don't care in power state table. Also avoid use of "!=" in -supply_expr. This brings uncertainty in the voltage and sate values of supply functions.

Conclusion

To specify the power modes, power states are now defined on supply sets & power domains using UPF add_power_state command. As this UPF command provides lot of flexibility and power, it often results in potentially complex and difficult to understand power intent specifications. With PST's, the static analysis and debugging of design was a fairly straight forward process. However the static analysis & debugging is not intuitive with power state definitions involving supply and logic nets. In this paper, we first highlighted the power aware verification challenges involved for a design having power states defined using add_power_state command. Additionally, we demonstrated how the process of static analysis can be eased up if power states are converted and interpreted as PSTs. The paper also includes guidelines to define power state definitions following which can help ease the verification process.

References

[1] IEEE Std 1801™-2013 for Design and Verification of Low Power Integrated Circuits. IEEE Computer Society, 29 May 2013
[2] Static and Formal Verification of Power Aware Designs at the RTL Using UPF (Rudra Mukherjee, Amit Srivastava and Stephen Bailey), DVCon 2008
[3] Unleashing the Full Power of UPF Power States (Erich Marschner, John Biggs), DVCon 2015
[4] Debug Challenges in Low-Power Design and Verification (Durgesh Prasad, Madhur Bhargava, Jitesh Bansal, Chuck Seeley), DVCon 2015