Free Yourself from the Tyranny of Power State Tables with Incrementally Refinable UPF

Progyna Khondkar, Ping Yeung, Gabriel Chidolue, Joe Hupcey, Rick Koster and Madhur Bhargava
Mentor Graphics Corporation, Fremont, CA.

Introduction

The recent edition of IEEE 1801 specifies

- ❑ The power state table (PST) construct should be phased out as legacy, and
- ❑ Replaced by the new semantics of the '**add_power_state**' UPF command.

This new construct allows

- ❑ Incremental refinement of Power states for
  - Power domains and
  - Its associated supply sets.

- ❑ This feature provides a way to model any possible combination of power states for a power domain and its available supply sets,
- ❑ As well allowing modification of these defined power states to accommodate the same UPF construct for different design abstraction levels.
- ❑ Consequently such features serve the purpose of the legacy PST, but in more complete and controllable ways.

The New Power State Constructs

- ❑ Power State of Supply Set and Power Domain

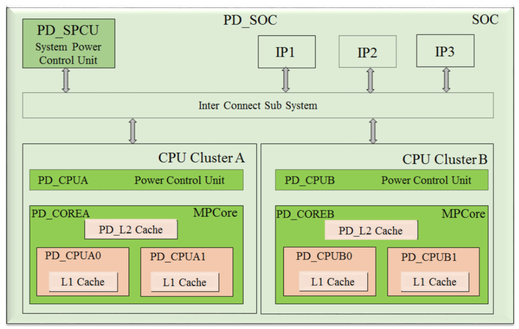

Figure 1. Example of a Complex SoC (Courtesy: ARM SOC)

- ❑ Power States of a Supply Set

```
add_power_state PD_CPUA0.primary
-state {ON -simstate NORMAL -logic_expr {pwr_ctrl==1}
 -supply_expr {power=={FULL_ON,1.0} &&
ground=={FULL_ON,0}} }
```

- ❑ Power States of a Power Domain

```
add_power_state PD_CPUA0
-state {ON -logic_expr{PD_CPUA0.retention ==ON &&
nRETCPU0==1 && PD_CPUA0.primary==ON}}
```

- ❑ Example of the roots of fundamental power states of an object (i.e. power domain PD_CPUA0)

```
add_power_state PD_CPUA0 -domain \
-state {UNDEFINED -logic_expr {PD_CPUA0 != RUN  &&
PD_CPUA0 != SHD} } \
-state {RUN –logic_expr {primary == ON} } \
-state {SHD –logic_expr {primary == OFF} } \
-state {ERROR -logic_expr {PD_CPUA0 == RUN && PD_CPUA0
== SHD} }
```

Limitations of Legacy PST

- ❑ Power states may also represented by the states of a power state tables (PST),

- add_port_state, add_pst_state, and create_pst.

These commands are legacy in UPF 2.1 LRM.

- ❑ Because of their limitations to coordinate with supply sets; specifically PST states are defined based on the supply net only, and supply nets are usually available after the synthesis and post-layout levels of design abstraction.
- ❑ There are no UPF methodologies for PST states to refine states.
- ❑ Dependency on upfront information of supply ports and supply nets severely delays the point at which power aware verification can start.

- ❑ Other known limitations
  - PST states - defined on supply net state only
  - Logic net states cannot be used
  - No logical view of the system states
  - Power domain states cannot be specified
  - Complex PST composition is not possible
  - No support for hierarchical reference to state
  - Reduction of states is not possible

The New Concept of Fundamental Power States

- ❑ The fundamental power state refinement concept extends the UPF specification and associated PA verification boundary to early stages of RTL design and allows verification virtually at any level of design abstraction.

- ❑ Definite Power State
  - Defining **–logic_expr c**ontains only operators == and &&
  - Refers only to other Definite states (of same or dependent objects)
  - Refined by derivation or branching

- ❑ Deferred Power State
  - A Definite State that is not yet fully defined
  - No defining expression (i.e. no ***–logic_expr***{})
  - Refined in place (actually through ***–update***)

- ❑ Indefinite Power State
  - Defining ***–logic_expr*** contains !, !=, or ||
  - Refers to an Indefinite state of a dependent object
  - Not refinable (UNDEFINED state)
  - But not a misnomer; Usable for don't care states!

Refinement by Derivative for Definite States

- ❑ Example of Definite Power States

```
add_power_state PD_COREA -domain \
-state {RUN –logic_expr {primary == ON}} \
-state {SHD –logic_expr {primary == OFF}}
```

- ❑ Definite Power States Refinement by Derivatives

```
add_power_state PD_COREA  -domain –update \
-state NEW_RUN {-logic_expr \
{(power == {FULL_ON, 1.1} ) && (ground == {FULL_ON, 0.0})}}
```

- ❑ A new power state, with a new ***–state*** name and updated ***-logic_expr***, based on the original power state

Refinement In Place for Deferred States

- ❑ Example of Deferred Power States

```
add_power_state PD_COREA.primary \
-state ON {-logic_expr {ln3 == 1} -simstate NORMAL} \
-state SHD {-logic_expr {ln3 == 0} -simstate CORRUPT}
```

- ❑ Deferred Power States Refinement in Place

```
add_power_state PD_COREA  -domain –update \
-add_power_state PD_COREA.primary -supply  -update\
-state {RUN –logic_expr {nPWRUP_CON==1'b0}}
{FULL_ON, 0.0})}}
```

- ❑ Refinements through ***–update*** implies that it actually modifies the original definition rather than creating a new definition or a new power state.

Advantages of Definite and Deferred States

- ❑ Allows to model UPF, i.e. power management architecture from very early stage of design.
- ❑ Allows to integrate design IP any time in the power management architecture.
- ❑ Allows to analyze and validate UPF strategy requirements.
- ❑ Allows to compute accurate state transition coverage information through interdependent states and
- ❑ Prevent intermediate state transitions of definite and deferred power states during refinement.

Case Studies

- ❑ UPF Strategies Analysis and Validation through Definite Power States

```
add_power_state PD_CPUA0.primary \
-state ON {-logic_expr {ln3 == 0} -simstate NORMAL} \
-state OFF {-logic_expr {ln3 == 1} -simstate CORRUPT}
add_power_state PD_COREA.primary \
-state ON {-logic_expr {ln3 == 1} -simstate NORMAL} \
-state OFF {-logic_expr {ln3 == 0} -simstate CORRUPT}
```

- ❑ Static Analysis Report

```
Source power domain: ~/PD_CPUA0 to Sink power domain: ~/PD_COREA.
      Total 3 Missing isolation cells [Total Crossings: 3, Shared Crossings: 0]
      1.1. Source port: ~/q_A [LowConn] to Sink port: ~/q_A [HighConn], width:1
      Total 1 Missing isolation cells [Total Crossings: 1, Shared Crossings: 0]
          1.1.1. Inferred type: ISO_MISSING, count: 1
Possible reason: 'Isolation is required from (~/PD_CPUA0)=>(~/PD_COREA)
and neither isolation strategy nor isolation cell is present in design' Analysis link:
[PD1_to_PD2].
```

- ❑ Isolation analysis and validation done without **set_isolation** strategy definition or implementation UPF at a very early stage of design abstraction.

- ❑ Coverage Computation of Power State-Transition through **describe_state_transition**

```
-------------------------------------------------------------------------------
UPF OBJECT                                Metric     Goal   Status
-------------------------------------------------------------------------------
TYPE: SUPPLY SET /cpu_tester/dut/PD_CPUA0.primary
State Coverage
   Power State PD_CPUA0_low_volt                  100.0%      100   Covered
      bin ACTIVE                          3       1   Covered
   Power State PD_CPUA0_moderate_volt             100.0%      100   Covered
      bin ACTIVE                          1       1   Covered
```

- ❑ Coverage Computation of Power State-Transition through **describe_state_cross_coverage -*domains*** {PD_COREA} -***depth*** 3 ….. etc.

```
-------------------------------------------------------------------------------
UPF OBJECT                                Metric     Goal   Status
-------------------------------------------------------------------------------
TYPE: POWER STATE CROSS
/cpu_tester/dut/PD_COREA(ID:PD1),
/cpu_tester/dut/PD_CPUA0(ID:PD2),
/cpu_tester/dut/PD_L2(ID:PD3)                  33.3%      100   Uncovered
Power State Cross Coverage                     33.3%      100   Uncovered
      bin \PD1:SLEEP-PD2:PD_CPUA0_off-PD3:PD_L2_off  0 1   ZERO
bin \PD1:RUN-PD2:PD_CPUA0_on-PD3:PD_L2_on          2   1   Covered
```

Conclusions

- ❑ PST replacement with **add_power_state** is straightforward and simple
- ❑ But the change will impact the power specification methodologies, power aware verification algorithms, tools, techniques and the entire design, verification and implementation flow.
- ❑ Because definite (indefinite as well) and deferred power states provides the intrinsic flexibility, essential controllability and the gifted features of *refinement* options.