Lets disCOVER Power States

Pankaj Kumar Dwivedi, Amit Srivastava, Veeresh Vikram Singh
Mentor Graphics Corp. 8005 SW Boeckman Rd. Wilsonville, OR 97070

ABSTRACT

Power states are a key aspect of today's low power designs. They capture the intent about the operating modes of the low power design and hence have a huge impact on the functionality. The creation of power states starts as early as the system design phase and persists through all of the implementation phases. So it becomes essential to verify the occurrence of various power states and their transitions to ensure proper operation of low power designs. A typical low power IP operates in several modes and when an SoC comprising of several of such IPs is verified it becomes critical to ensure proper coverage of those states. Because the power states are captured in UPF in an abstract manner, it becomes a challenge to capture the coverage metric of those states and their transitions. In this paper, we demonstrate a methodology and flow to enable the coverage collection of power states that is generic and customizable. The paper will also highlight the importance of various metrics associated with power state coverage and the challenges faced in modelling them.

POWER STATES

- Capture the operating modes of a low power design.
- Abstract representation of
  - Voltage and current characteristics of the supplies
  - Operating modes of the elements
- IEEE Std 1801™-2013 Unified Power Format (UPF) provides add_power_state command
  - Allows attributing power states on various UPF objects, supply sets, power domains and system/subsystem

WHY COVERAGE OF POWER STATES?

For comprehensive verification of power states, a verification engineer is interested in a number of questions, such as:

- All the desired power states reached or not?
- All desired power state transitions reached or not?
- Any illegal power state reached?
- Any illegal power state transition occurred?

These kinds of questions are easily addressed by coverage-driven verification. But, it becomes a challenge to capture the coverage information of power states due to following two reasons:

- Power states are written in an abstract manner in UPF,
- There is no pre-defined coverage metric to capture power states and their transitions.

COVERAGE METRICS FOR POWER STATES

1. STATE COVERAGE

- Ensures that all the power states are exercised during verification
- Analogous to traditional FSM coverage but power states are asynchronous

2. TRANSITIONS AND DESCRIBE STATE TRANSITIONS COVERAGE

- Ensures that
  - All valid transitions are covered
  - Invalid ones (described via the UPF command describe_state_transition) are highlighted

3. CROSS STATE COVERAGE

- Inter-connected power domains lead to various combinations of power states
- These interconnections affect the placement of power management cells
- Cross states coverage verifies these combinations

CHALLENGES IN MODELING COVERAGE METRICS

- Unified Coverage Interoperability Standard (UCIS) does not provide any metric to capture power intent (power states, etc.).
- Coverage metrics require asynchronous sampling
- More than one power state can be true at a time
- States can be defined as illegal
- The power states of a UPF object can refer to the power states of other UPF objects

COVERGROUPS TO THE RESCUE

- SystemVerilog construct that samples signal/property activities at desired sampling points through coverpoints and bins.
- Can be effectively used to collect coverage numbers of power states and their transitions.
- Generous syntax to handle a wide range of complex sampling scenarios
  - Wildcard feature for handling state transitions when multiple states are true at the same time.

FURTHER CHALLENGES

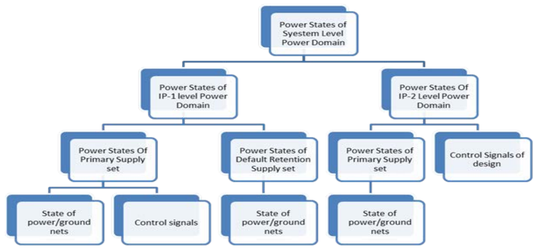
- How to access handles of power states or the objects where these states have been added?
  - Design controls, Supply Ports, Supply Nets, Power Domains and Supply Sets
- How to describe coverpoints and bins using these handles
- How to incorporate these power state metrics in the user's Design/Test?

COVERAGE METHODOLOGY

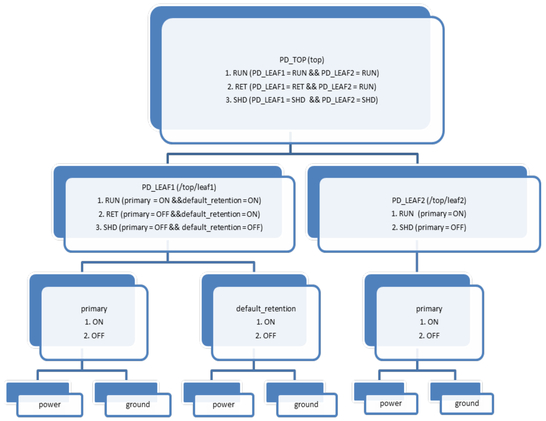
- All power state objects have corresponding checker modules containing coverage logic
- Checker modules have covergroups and the rest of the logic required to model various power state coverage metrics
- These checker modules would be inserted into the design using bind checkers.

1. POWER STATE MODELING

- Modeled typically in a hierarchical manner involving various supply sets and power domains.
- Defined via logic and supply expressions of the add_power_state command.



2. UPF HIERARCHY



3. CHECKER MODULE INTERFACE

- Checker modules require handles of various RTL and UPF objects referenced in the power state.These handles are declared as ports of checker modules.

4. COVERAGE LOGIC

- SV Boolean expressions representing various power states are assigned into variables called state variables.

COVERAGE METHODOLOGY(CONTD.)

- For transition coverage, an array with state variables as elements is defined (Transition variable).
- Any change in any of the state variables will trigger a clock which will act as the sampling event for state and transition coverage (Sampling Clock).
- Covergroups are defined for state and transition variables with the clock as sampling event

```
module cov_PD_LEAF1_primary_PS (input suply_net_type power, ground);
    wire state_OFF = ((power.state == UPF::OFF) || (ground.state == UPF::OFF));
    wire state_ON = (((power.state == UPF::FULL_ON) && (power.voltage == 1210000))…
    wire [3:0] curr_state = {state_OFF, state_ON ,…};
    always @(state_OFF, state_ON,…)
      cov_clk = 1'b1;

    covergroup primary_STATE_COVERAGE  @(posedge cov_clk);
    OFF: coverpoint state_OFF { bins ACTIVE  = (0=>1); }
    ON: coverpoint state_ON { bins ACTIVE  = (0=>1); }
    …
    endgroup
    covergroup primary_TRANSITION_COVERAGE  @(posedge cov_clk);
    primary_TRANSITION_COVERAGE:coverpoint curr_state {
    wildcard bins OFF_to_ON  = (4'b???1=> 4'b???1?);
    wildcard bins ON_to_OFF  = (4'b??1?=> 4'b???1);
     … }
    endgroup
  endmodule

    module cov_PD_LEAF1_PS ();
      wire state_SHD = top.leaf1.PD_LEAF1_primary_PS_coverage.state_OFF &&
top.leaf1.PD_LEAF1_default_retention_PS_coverage.state_OFF;
      assign curr_state = {state_SHD, state_RET, ……};
      covergroup PD_LEAF1_STATE_COVERAGE  @(posedge cov_clk);
      SHD: coverpoint state_SHD{bins ACTIVE  = (0=>1); }
      …
    endgroup
  endmodule
```

5. BINDING COVERAGE MODELS

- The place of insertion will be the scope where a power state object has been defined in UPF.
- UPF bind_checker command for accessing various objects and inserting the checker module at the appropriate places.

```
bind_checker PD_LEAF1_primary_PS_coverage
-module cov_PD_LEAF1_primary_PS
-elements {/top/leaf1}
-ports {{power PD_LEAF1.primary.power} {ground
        PD_LEAF1.primary.ground}}
```

5. SIMULATION RESULTS

```
# POWER STATE COVERAGE:
# ------------------------------------------------------------------------
# UPF OBJECT                                    Metric      Goal/ Status
# ------------------------------------------------------------------------
#  TYPE : SUPPLY SET /top/leaf1/PD_LEAF1.primary   100.0%    100 Covered
#    Power State OFF                               100.0%    100 Covered
#      bin ACTIVE                                      2      1 Covered
#    Power State DEFAULT_CORRUPT                   100.0%    100 Covered
#      bin ACTIVE                                      1      1 Covered
# …
#  TYPE : SUPPLY SET /top/leaf1/PD_LEAF1.primary    33.3%    100 Uncovered
#    Power State Transitions                        33.3%    100 Uncovered
#      bin OFF -> ON                                   1      1 Covered
#      bin DEFAULT_CORRUPT -> OFF                      0      1 ZERO
# …
#  TYPE : POWER DOMAIN /top/PD_TOP                  16.6%    100 Uncovered
#    Power State Transitions                        16.6%    100 Uncovered
#      bin SHD -> RET                                  0      1 ZERO
#      bin SHD -> RUN                                  1      1 Covered
# …
# TOTAL POWER STATE COVERAGE: 62.5% POWER STATE COVERAGE TYPES: 12
```