Addressing the Complex Challenges in Low-Power Design and Verification

Madhur Bhargava (Mentor Graphics)
Durgesh Prasad (Mentor Graphics)
Jitesh Bansal (Mentor Graphics)
Gabriel Chidolue (Mentor Graphics)
Agenda

• Introduction
• Low-Power Verification using UPF
• Debugging challenges and their solution
  – UPF Specification complexities
  – Illegal power state transitioning
  – Unwanted X debugging
  – Expecting X during power down
  – Supply network debugging
  – Design Power-Up failure
• Conclusion
Introduction

Today's SoCs
• Are incredibly Complex
• Have sophisticated power management strategies for highly power efficient design
• Integrate variety of implementation cells
  – Isolation, retention, buffers etc.

They Must
• Verify the power management
  – Debug the complex issues
Unified Power Format (UPF)

- RTL is augmented with a UPF specification
  - To define the power architecture for a given implementation

- RTL + UPF drives implementation tools
  - Synthesis, place & route, etc.

- RTL + UPF also drives power-aware verification
  - Ensures that verification matches implementation
Low Power Verification Using UPF

• Different Systems have different power management
• Power Gating
  • Isolation
  • Retention
• Multi-Voltage
  • Level Shifting
• Body Bias and Power Gating
• Power States
• UPF provides commands to
  • express the power management strategies e.g set_isolation
  • verify the power architecture e.g bind_checker
UPF Migration issue

• **Problem:** UPF supplies default to OFF state with UPF 2.0 however they were default on in UPF 1.0

• **Solution:** Utilize UPF package functions to explicitly turn on all necessary supplies

module tb;
import UPF::*;
...
initial begin
    supply_on ("tb/dut_inst/VDD", 1.1);
    supply_on ("tb/dut_inst/GND", 0.0);
end
...
dut dut_inst (...);
...
endmodule
UPF wildcard expansion issue

• **Problem**: HDL/IP block placed in incorrect power domain

```verilog
module dut;
...
Ip_module my_ip1();
Ip_module my_ip2();
// my_ip3 is powered separately from my_ip1/my_ip2
Ip_module my_ip3();
endmodule

UPF: create_power_domain pd_dut -elements {my_ip*}
```

• **Solution**: Use `save_upf` command to create interpreted UPF code or use `find_objects` command to print out expanded TCL variable

```tcl
set ip_list [find_objects -scope dut -pattern {my_ip*} \
-object_type instance]
puts $ip_list
```
Incorrect/Missing VCT Specification

- **Problem:** HDL GND pin driven to a logic “1” when connected to UPF GND supply

- **Solution:**
  - Use proper VCT option `connect_supply_net` command OR
  - Use `set_port_attribute` command to designate HDL GND pin as `primary_ground` OR
  - See if your EDA vendor provides some option to automatically detect GND pins

```bash
connect_supply_net upf_GND -ports {hm_inst/GND} -vct UPF_GNDZERO2SV_LOGIC
set_port_attributes -pg_type primary_ground -ports {hm_inst/GND}
```
Illegal Power State Transitioning

**Problem:** How to ensure that unwanted power state and power state transitions don’t occur

**Solution:** Leverage UPF 2.0 `add_power_state` and `describe_state_transition` commands to declare illegal power states and state transitions

```upf
add_power_state PD_ALU_SS -state ON4 { \
  -logic_expr { !pwr_alu && !pwr_ram } \
  -simstate CORRUPT \n  -illegal}
```

# ** Error: UPF_ILLEGAL_STATE_REACHED: Time: 129 ns, Supply set 'PD_ALU_SS' reached an illegal power state 'ON4'.
# File: src/parser_test22/demo.upf, Line: 73, Power state:ON4
Illegal Power State Transitioning

• **Problem:** How to ensure that desired power State and power state transitions occur during simulation

• **Solution:** Leverage vendors capabilities in displaying and reporting power state and power state transition coverage data
Unwanted X: Power Aware Corruption

- **Problem:** Is X value on a signal due to power domain corruption
- **Solution:** Use Waveform highlighting to distinguish X value is caused by power domain corruption
Unwanted X: Missing PA cell

- **Problem**: Is X value on a signal due to missing UPF inserted logic or the power of UPF inserted logic is not turned on

- **Solution**: Trace drivers in schematic window
Unwanted X: Power-up failure

- **Problem**: Low power simulation failures occur after power up of a domain
- **Solution**: Use Waveform compare feature to easily detect simulation differences where X values remain after power up also enable low power messages and or assertion checks
Unwanted X:Control signal corruption

• **Problem**: Power control signals are unexpectedly getting corrupted when certain domains are powered down

• **Solution**: Ensure that any buffers on these “always on” nets have PA semantics disabled via
  – Manually exclude these cells using vendor specific exclusion mechanism such a exclude files or setting DON’T_TOUCH attributes on them
  – Leverage Liberty files which contain always_on attributes to auto exclude them from corruption
Expecting ‘x’ during power down

• Possible Reason: Improper Power Domain Supplies

• Solution:
  – Use the tool processed static reports to find out if power domains has been rightly created

Creation Scope: /mem_test/top

Extents:
  1. Instance: /mem_test/top
  2. Instance: /mem_test/top/mem_r, Lower Boundary

  – Use the tool dynamic messages for power domain

  e.g. ** Note: (vsim-8902) MSPA_PD_STATUS_INFO: Time: 64 ns, Power domain 'pd_top' is powered down.
Expecting ‘x’ during power down

- Possible Reason: Simulation semantic is disabled due to
  - The design element is already power aware with pg-pins connected via UPF.
  - The design element is an always-on cell
  - The design element is a power aware cell but it could not be mapped to it’s UPF strategy

- Solution: Identify the exact reason via looking at the tool processed report/messages.
  - e.g ** Note: Power Aware simulation semantics disabled for u_top_0/u_ip_1 because it is an always_on cell.
Supply Network Issues

• The complexity in UPF supply network is due to
  – connections involving many UPF objects and multiple hierarchies - e.g supply nets, ports, switches and power aware cells.
  – Multiple supply ports can drive a same supply net
    • resolution function on the supply net
      `create_supply_net N1 -resolve parallel`
      `connect_supply_net N1 -ports { bot1/P1}`
      `connect_supply_net N1 -ports { bot1/P2}`
  – Connection from UPF supply net to HDL nets
    • Implicit or explicit vct is required.
  – Driver/receiver direction not clear from command
    “`connect_supply_net N1 -ports { bot1/P1}`”
  – The ports/nets defined in liberty only
Debugging Supply Network Issues

• Static debugging
  – Check for correct VCT and resolution functions
  – Determine driver/receiver from connect command

```bash
create_supply_port P1 -direction out
create_supply_port P2 -direction in
...
connect_supply_net N1 -ports { bot1/P1}
connect_supply_net N1 -ports { mid1/P2}
```

– Lookout for tool generate reports for complete connection
Debugging Supply Network Issues

• Dynamic debugging
  – Use the tool dynamic simulation and UI capability
    • To get the connection between design elements
    • To get the value of supply nets/ports.
Design Power-up failure

• Missing/incorrect Isolation and level-shifter
  – Level-shifters required to translate logic values properly between 2 domains.
    • Missing or incorrect cell may lead to read a logic value ‘1’ as ‘0’ and vice versa
  – Isolation cells are required to isolate corrupt value from powered down region to a valid value
    • Controlled via isolation enable signal
    • Missing or incorrect cell may cause ‘x’ to propagate in a powered-up domain
Isolation/level-shifter issue debugging

• Static Verification
  – Specify expected states via add_port_state/add_pst_state (UPF 1.0 style) or add_power_state (UPF 2.x style)
  – Use the tool static checks capability to find out
    • Incorrect/missing cells, Not required cells

• Dynamic Simulation helpful if
  – The design intention specified via PST/add_power_state doesn’t match with testbench vectors
  – Use the tool dynamic checks capability to find out
    • Incorrect/missing cells, Not required cells

** Error: UPF_MISSING_LS_CHK: Time: 90 ns, Missing level shifters for domain boundary, LOW_VOLT_PD (Operating Voltage: 5.000000 V) => HIGH_VOLT_PD (Operating Voltage: 5.200000 V) for Source port: out1 -> Sink port: in1
  File: ./src/test.upf, Line: 32, Power Domain: LOW_VOLT_PD
Isolation/level-shifter issue debugging

• Use Bind checkers
  – UPF provided way to write custom assertions
  – bind_checker command can access UPF supply nets/ports as well as design signals.

• Checker module example to find missing/incorrect Isolation
  – Source Domain: OFF, Sink Domain: ON, Signal: ‘X’

module checker_isolation(input op, src_supply, sink_supply);
  always@(src_supply)
    assert ((src_supply == ON) || (sink_supply == OFF) || (op != 'X'))
    else $error("Missing/Incorrect Isolation");
endmodule
Initial block re-evaluation

- **Problem:** For certain models, such as a ROM memory, initial block may need to be re-executed on power-up after time 0
- **Solution:** Utilize vendors individual solutions to specify which modules or blocks need to have initial blocks re-executed at power up or exclude object from PA semantics

```
module rom_mem ( input [7:0] addr, input re, clk, output reg[7:0] data_out);
  reg[7:0] mem[255:0];
  initial begin
    $readmemh("rom.mem", mem);
  end
  assign data_out = (re == 1) ? mem[addr] : 8'hzz;
endmodule
```
Retention related issues

• Retention not specified
  – ‘cnt’ needs to be retained
  – “set_retention” is required.

```verilog
always @(posedge cnt_rst, posedge clk)
begin
  if(cnt_rst)
    cnt <= 16'b0;
  else
    cnt <= cnt + 1;
end
```

• Incorrect Retention protocol
  – Typical retention protocol
    Retention save -> power down -> power up -> retention restore.
Debugging retention related issues

- **Problem**: Registers remain X after power-up. Could be missing retention in UPF or non-retention register needing reset on power-up.
- **Solution**: Utilize vendors automated retention assertion to detect incorrect retention protocol or non-retention registers that need to be reset.

Error: (vsim-8912) MSPA_NRET_ASYNCFF: Time: 12 ns, Asynchronous (set/reset) control for the following flop(s) of power domain 'PD1' is not asserted at power up:

Error: (vsim-8903) MSPA_RET_OFF_PSO: Time: 64 ns, Retention control (0) for the following retention elements in scope '/tb/top_vl' of power domain 'pd' is not asserted during power shut down:
Advance debugging of retention issues

- Using advanced bind checker

```
set_retention R1
   -domain PD
   -save_signal {SAVE posedge}
   -restore_signal {RESTORE posedge}
   -restore_condition {UPF_GENERIC_CLOCK}
   -elements {Q1 Q2}

module checker_ret #(parameter name)
   (input ret_ff, restore_sig, clock)
   always@(posedge restore_sig)
   assert (clock != 1) else $error("Incorrect restore protocol for %s", name);
endmodule

array set RET [query_retention R1 -domain PD -detailed]
set ELEMS $RET(elements)
set RSTR  $RET(restore_signal)

bind_checker ret_checker_inst
   -module ret_checker  -elements $ELEMS \n   -parameters { {elem_name UPF_GENERIC_ELEM_NAME} } \n   -ports {{ret_ff UPF_GENERIC_OUTPUT} \n               {restore_signal RSTR} \n               {clock UPF_GENERIC_CLOCK}}
```
Macro cell corruption Issues

• Corruption based on liberty
  – power_down_function and related_power
    • Uses primary supplies of macro cell
    • correct connection of these primary supplies with domain supplies is required

• Common reasons for wrong supply connections are
  – Unknown pg_type of primary supplies
  – Wrong inference of supply port direction
  – No bias supplies defined in primary supply set
  – Incorrect/No inferencing of PA cells
  – Incorrect/No backup supply connections for PA cells
Conclusion

• UPF does provide few handles for debugging
  – Save_upf
  – Tcl based usage
  – Bind checkers

• Rely on EDA tools for debugging
  – Learn the most common issues
  – Categorize the issue
  – Use the relevant feature of the tool for debugging
Questions