February 25-28, 2013
DoubleTree, San Jose

accellera SYSTEMS INITIATIVE

Power Aware Verification Strategy for SoCs

by

Boobalan Anantharaman
Design Engineer Staff

Arunkumar Narayanamurthy
Design Engineer Staff

Cypress Semiconductor technology India Pvt Ltd

AGENDA

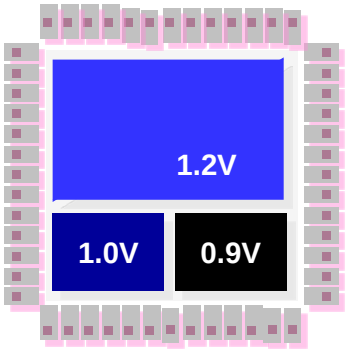
- Introduction
- SoC Power Aware Verification
- Power Aware Verification Flow
- Power Aware Verification Planning
- Tips for Catching Bugs Faster
- Debugging Tips
- Re-Usability Tips
- Conclusion

Introduction

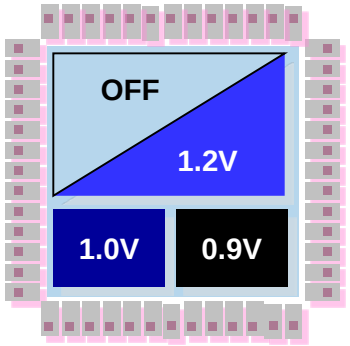
- Power management is essential for
  - Building Portable, lighter products
  - Longer Battery life
  - More features in the consumer applications
- Dynamic power
  - Signal switching consumes power
  - Major contributor to power consumption
- Static power
  - Static leakage can consume significant power
  - Major concern for power optimization

Power Management Techniques

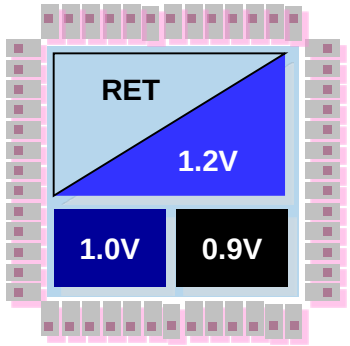
- The mandate to reduce system power consumption led to the increasing use of low-power IC design techniques
- IC designers - use advanced power management techniques to minimize static and dynamic power in the SoCs



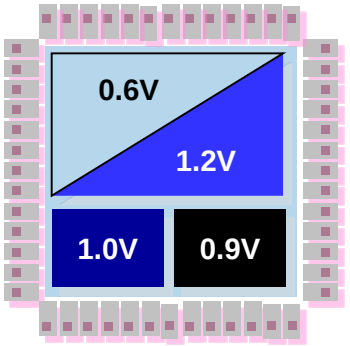
Multi-Vdd (MV)



Power gating (shut down)

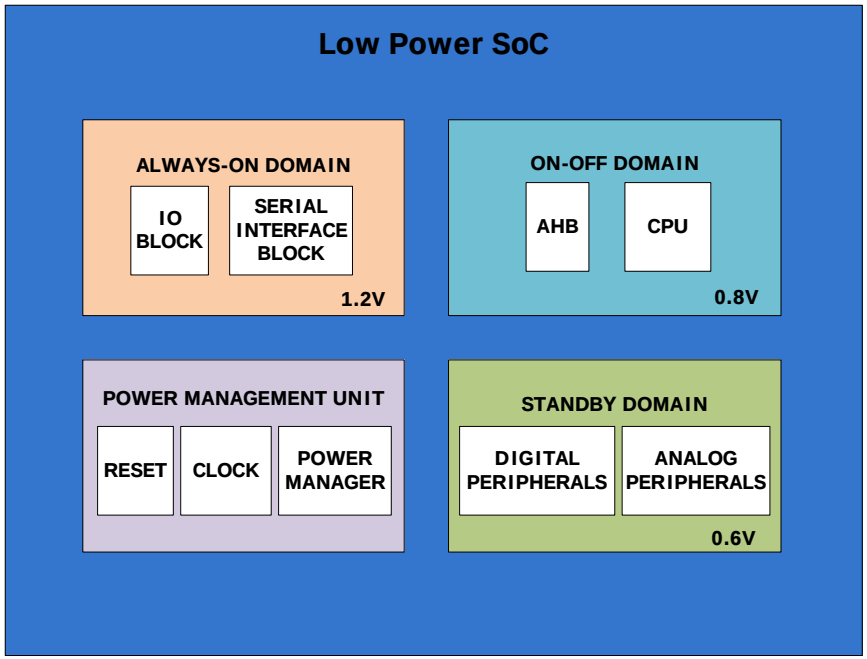


Power gating with State Retention



Low-VDD Standby

SoC Power Aware Verification



- For Low Power SoC designs with different power domains and power modes, PA Verification is a essential one for verifying
  - Power on Reset sequence
  - Power down/Power up control sequence
  - Isolation/Retention logic
  - Level shifting logic
  - Power Mode transitions
  - Power switching logic

Traditional Approach

- SoC Power Management logic is verified on PG connected netlist

Drawbacks

- Too late in the verification flow
- Defect rectification consumes more time
- Netlist simulations are slow compared to RTL
- Hard to debug the issues at netlist level

Power Intent based Approach

- SoC Power Management logic is verified using power intent specification (e.g. UPF) on RTL

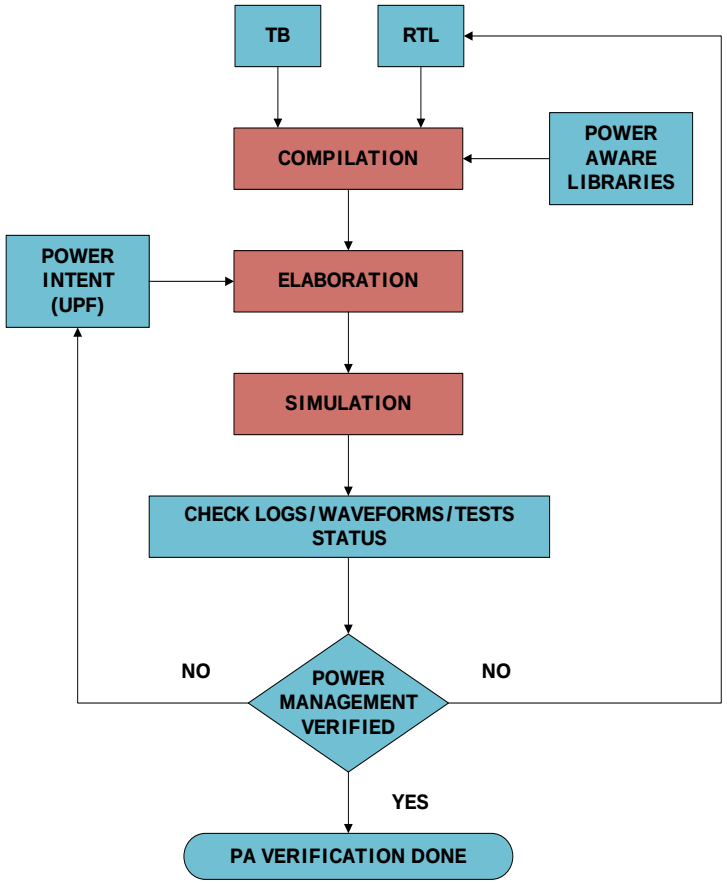
Pros

- Too early in the verification flow
- Power architecture related bugs can be caught earlier
- Easier to debug the issues at RTL level

Cons

- Creation of Power Intent specification file
- Tool dependency on Power Intent specification interpretation

Power Aware Verification flow



PA Verification flow uses

- RTL or Netlist
- Test bench
- Power Intent specification
- Power aware libraries

What is UPF?

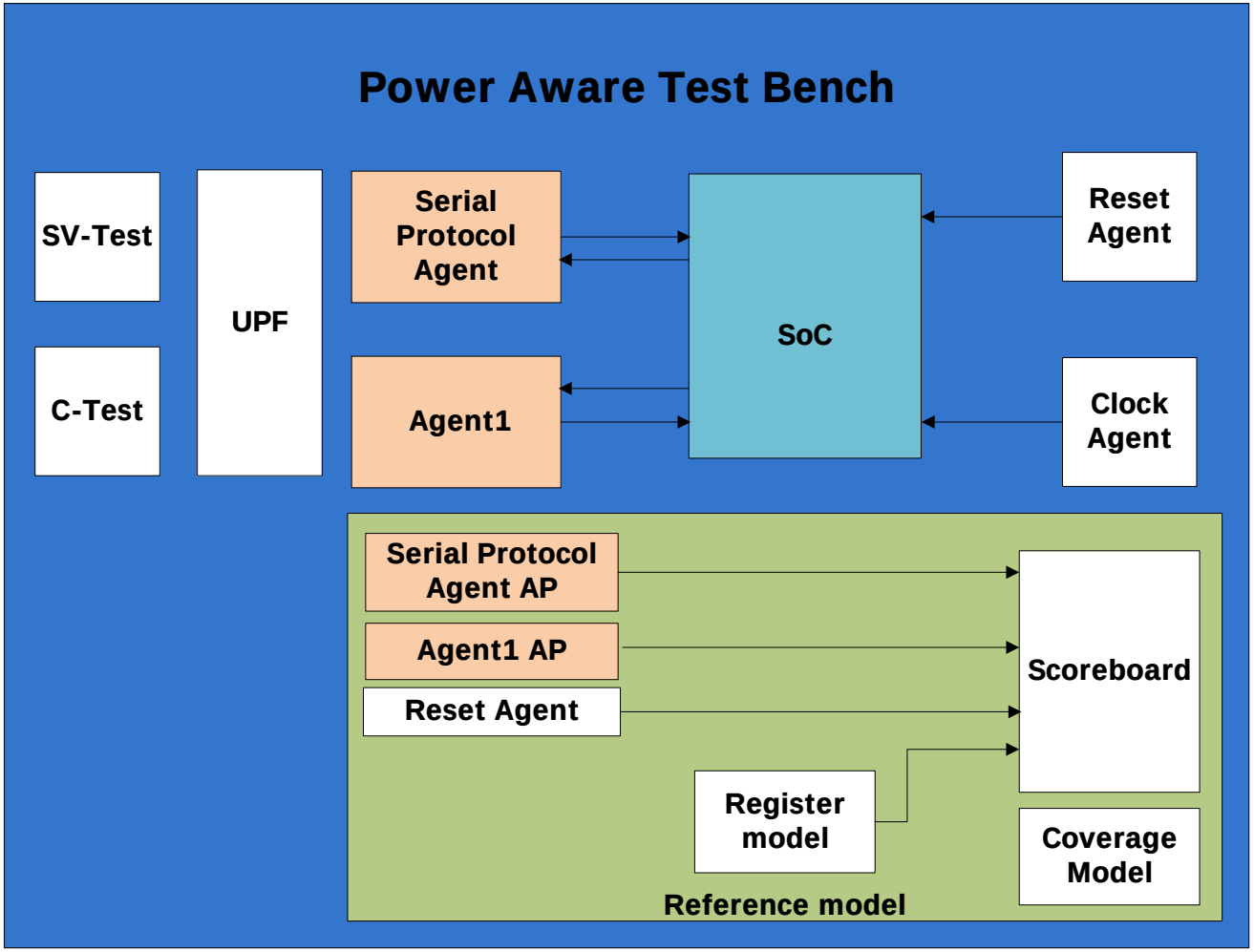
- The Unified Power Format (UPF) is a IEEE 1801-2009 standard provides the concepts and notation required to define the power management architecture for a design
- UPF specification can be used for power management verification, during RTL/GLS simulation
- UPF file typically includes information about
  - Power Domains
  - Power state
  - Isolation and Retention
  - Level Shifting
- Most leading vendor tools support UPF and it is HDL independent

UPF Example

```
/*****Create Power domain*****/
create_power_domain SLP  -elements {u_slp}
create_power_domain STNDBY  -elements {u_stndby}

/**********Isolation**************/
set_isolation VCCD_sram_iso  -domain VCCD_sw
-isolation_power_net vccd_sw  -isolation_ground_net vssd
-elements {u_cpu_mem_top/sram_rdata[4]
           u_cpu_mem_top/sram_rdata[3]
           u_cpu_mem_top/sram_rdata[2]
           u_cpu_mem_top/sram_rdata[1]
           u_cpu_mem_top/sram_rdata[0]}
-clamp_value 0
set_isolation_control  VCCD_sram_iso  -domain VCCD_sw
-isolation_signal u_cpu_mem_top/sram_isolate
-isolation_sense high  -location self
map_isolation_cell VCCD_sram_iso -domain VCCD_sw
-lib_cells {scls_lp_inputiso0n_lp scls_lp_inputiso0p_lp2 }

/*************Level shifter*********/
set_level_shifter HV2LV_LS_RULE -domain VDDD_PD \
-applies_to outputs \
-location self \
-rule high_to_low \
-threshold 0
map_level_shifter_cell HV2LV_LS_RULE
-domain VDDD_PD -lib_cells { scs8hvl_lsbufhv2lv_1 }
```

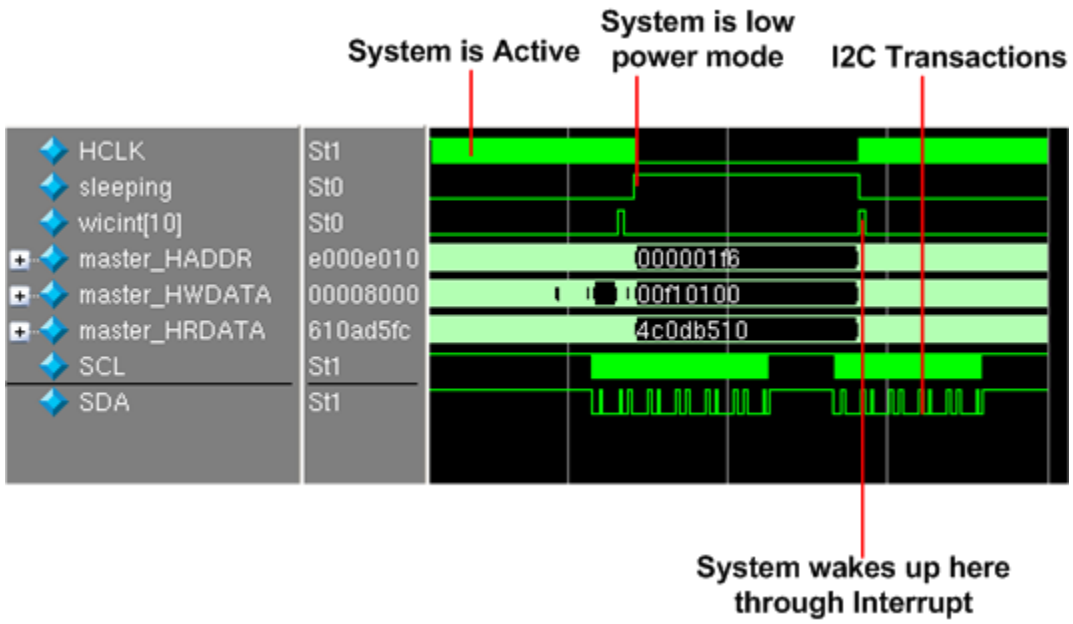
```
/******* Retention Strategy *******/
set_retention ret_stndby -domain STNDBY
 -retention_power_net VDD_rail
 -retention_ground_net VSS_rail
set_retention_control ret_stndby -domain STNDBY
 -save_signal {save_cnt high}
 -restore_signal {restore_cnt high}
map_retention_cell ret_stndby -domain STNDBY
-lib_cell_type LIB_CELL_NAME

/***** Create Power state table****/
create_pst PST -supplies [list vddd vccd vccd_sw
vccdpslp vccstndby
u_cpu_mem_top/u_fm_32K/vpwri_fm
u_cpu_mem_top/u_fm_32K/vpwri_iref
u_cpu_mem_top/u_fm_32K/vpwri_vneg
u_cpu_mem_top/u_srom_0/vpwrv
u_cpu_mem_top/u_srom_1/vpwrv ]
add_pst_state active  -pst PST  -state {vddd_state
vccd_active vccd_sw_active vccslp_active vccstndby_active}
```

PA Test Bench Environment



Power Aware simulation result

- Waveform: Shows the active mode to low power mode switching and low power mode to active mode switching



Power Aware Verification planning

Power Management Bugs

- Control Bugs
  - Power down/power up control sequence bugs
  - Power state transition and sequencing bugs
  - Failure to reset after power down/power up
- Partitioning Bugs
  - Bugs due to incorrect implementation of system power modes
  - Bugs due to Cyclic domain state interdependencies
- Power Bugs
  - Bugs due to incorrectly structured power switching network
  - Bugs due to incorrect powering of logic elements
- Structural Bugs
  - Bugs due to missing Isolation cells/retention registers
  - Bugs due to missing, incorrect or redundant level shifters

Power Aware Verification planning Contd..

- The power aware verification plan should address all the scenarios which can catch all the power management related bugs
- Sample Scenarios
  - Does the power controller sequence power down correctly?
  - Are outputs isolated before power is switched?
  - Are states retained before power is switched?
  - Are outputs isolated to the correct state?
  - Does the power controller sequence power up correctly?
  - Is power restored before state retention registers restore their states?
  - Is power restored before isolation is removed?
  - Do all outputs have known value before isolation is removed?

Power Aware Verification planning Contd..

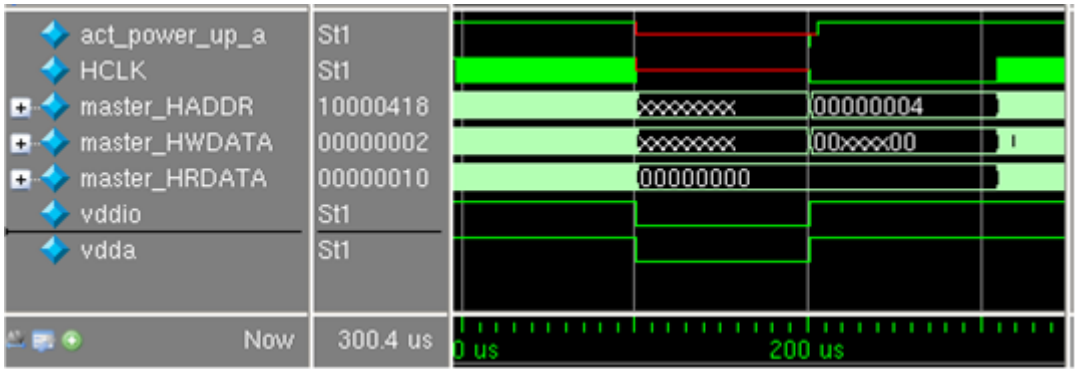
- Sample Scenarios
  - Does the system wake up correctly using different wake-up sources after being put in different low power modes?
  - Does the system function correctly when some parts are powered down?
  - Does the system change power states correctly?
  - Does the system recover back after Power On Reset is applied randomly?
  - Does the system recover back when external reset is applied after power down/up?
  - Are the system power modes implemented correctly?
  - Do the state machines in different Power domains restore to states that does not create deadlock in the design?

Tips for Catching Bugs Faster

- Cover all the power mode transitions earlier in the verification cycle
- Check all power mode failures by using assertions or by using self checking logic in the test case
- Enable multiple low power mode wake-up sources simultaneously and verify
- Route IP outputs to SoC IO pads in Low power mode if SoC design supports this feature
- Check all the wake-up sources in low power modes
- Check Retention/Non-Retention logic in all the low power modes

Tips for Catching Bugs Faster Contd..

- In any power mode, remove the main power supplies (vddd /vdda)
- Using this scenario, bugs related to power sequencing issues can be caught earlier



- Add assertions using power ports to check the port values in different power modes
- Add assertions for power mode transitions to find the mode transition related bugs faster

Tips for Catching Bugs Faster Contd..

- While serial/Parallel interface transactions are happening in active mode, trigger the Low power mode entry/exit and check that the active transactions are not aborted
- Check the low power mode entry and exit when SoC is operating at Max, Min, and Mid range SYSCLK frequencies
- Add power mode transition timing related assertions

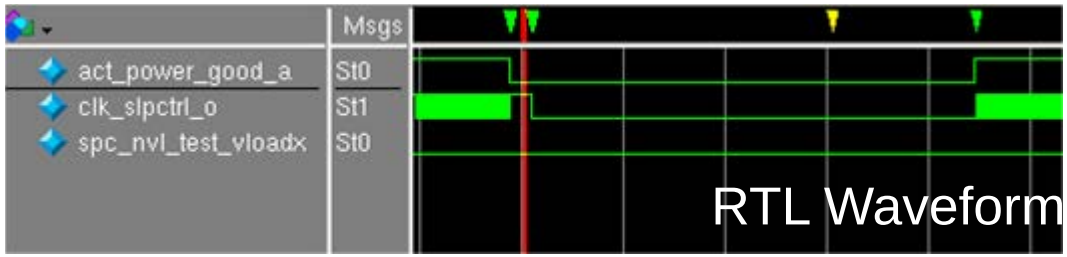
  e.g. Active->Sleep (Min time: 1us, Max time: 3us)
- Add display statements about mode transitions and about handshaking between the C-Test and SV-Test

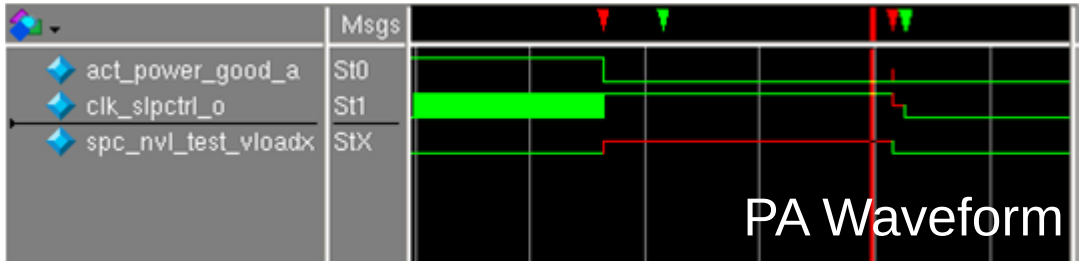
  e.g. ovm_report_info(get_type_name (),$psprintf("PASIM: DEVICE is in ACTIVE mode"), OVM_NONE);

Debugging Tips

Debug using RTL reference waveform

- 'X' propagation related issues and power-up, power-down sequence related issues can be debugged using Functional simulation waveform as reference





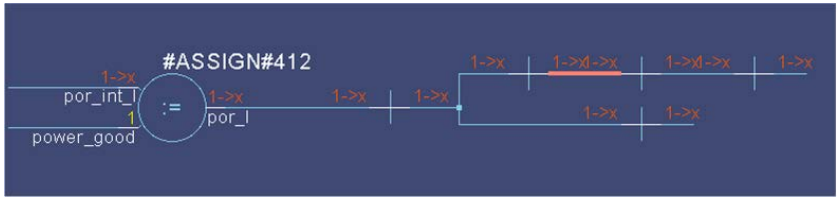
Debugging Tips Contd..

Back Tracing

- Back tracing feature can be used to debug ‘X’ propagation issues and multiple driver issues

```
VSIM(paused)> driver {sim:/p4mcu_tb/psoc4a_top_DUT/psoc4a_
top_dut/u_s8srsscore_top_wrapper/u_s8srsscore_top/u_s8srss
core_hard_top/u_s8srsscore_pwr_hard_top/ipor_reset_raw_pro
be_n {-format Default -height 18}}
# Drivers for /p4mcu_tb/psoc4a_top_DUT/psoc4a_top_dut/u_s8
srsscore_top_wrapper/u_s8srsscore_top/u_s8srsscore_hard_to
p/u_s8srsscore_pwr_hard_top/ipor_reset_raw_probe_n{-format
Default -height 18}:
#    St1   : Net /p4mcu_tb/psoc4a_top_DUT/psoc4a_top_dut/u_
s8srsscore_top_wrapper/u_s8srsscore_top/u_s8srsscore_hard_
top/u_s8srsscore_pwr_hard_top/ipor_reset_raw_probe_n
#      St1 : Driver /p4mcu_tb/psoc4a_top_DUT/psoc4a_top_du
t/u_s8srsscore_top_wrapper/u_s8srsscore_top/u_s8srsscore_h
ard_top/u_s8srsscore_pwr_hard_top/u_s8srsscore_pwr_hardip/
u_s8init_ipor/#ASSIGN#412
#
```

Driver Back tracing Result

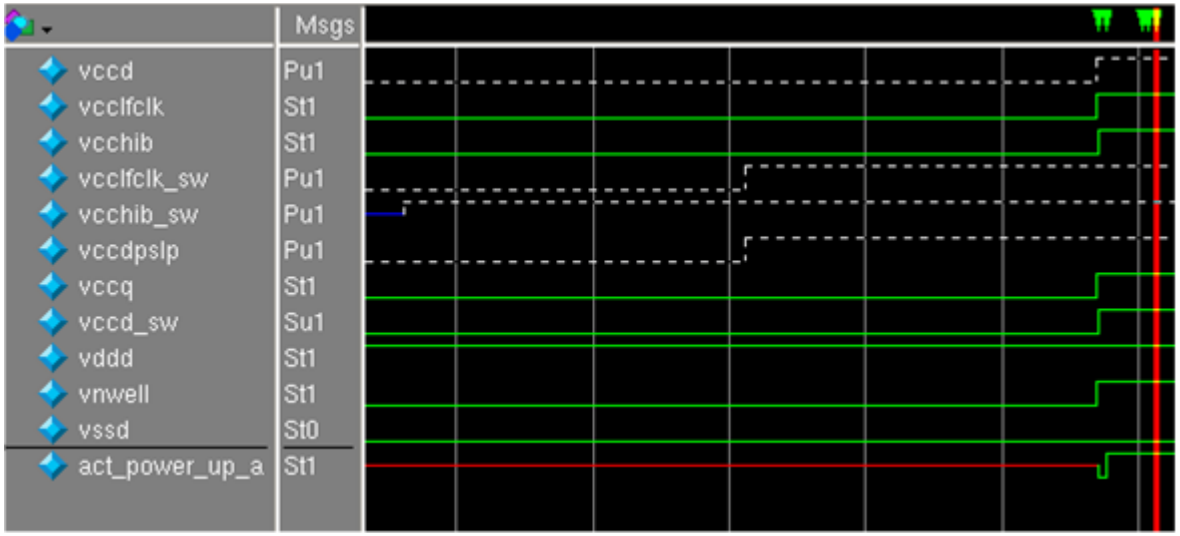


X Propagation Dataflow Diagram

Debugging Tips Contd..

Use Waveform loadable format

- To make debugging easier save all the important power supplies in the SoC as waveform loadable format



Debugging Tips Contd..

Tool Issue debugging

- Sometimes tools may drive the clamp values incorrectly, due to which 'X' propagation can happen
  - To justify whether that is a tool issue, forcibly drive the clamp value to 0/1 from the test bench
- When the supply on-off issue is suspected as tool issue, run the same test case on the PG connected GLS Netlist

Add Self checking Logic

- Add self checking logic in each of the test cases to make sure about proper power mode transitions

```
if (tst_pw_if.data[83:81] == 3'h1)
  ovm_report_info(get_type_name(),$psprintf("PA_SIM: DEVICE is in ACTIVE Mode Still"), OVM_NONE);
else
  ovm_report_error(get_type_name(),$psprintf("PA_SIM: DEVICE is NOT in ACTIVE Mode"), OVM_NONE);
```

Re-usability Tips

Re-use RTL Power related tests

- All RTL power related test cases can be re-used across PA simulation by just compiling same test bench with UPF, and PA related run-time options

Create Generic Scripts

- Create generic scripts to convert the UPF given by the design team to vendor tool specific format (if applicable)
- UPF and PA test cases can be re-used across different derivative products using scripts

Re-usability Tips Contd..

Avoid delays in Testcases

- Avoid delays while coding tests so that it can be re-used across GLS/PA/derivative projects
- Instead of delays, use events so that test case can be re-used across GLS/PA/derivative projects

```
task pa_active2dpslp_wdt_reset_ctest::run ();
  super.run();
  #4.5ms;
  // Keep the chip in Low-power mode for some time.
  ovm_report_info(get_type_name(),"The Chip entered into the Low-Power mode", OVM_LOW);
  // Stops all the process by calling this global stop request.
  global_stop_request();
endtask : run
```

```
task pa_active2dpslp_wdt_reset_ctest::run ();
  super.run();
  poll_active_mode.wait_trigger();
  ovm_report_info(get_type_name(),“Chip entered active mode", OVM_NONE);
  global_stop_request();
endtask : run
```

Re-usability Tips Contd..

UPF re-usability

- If the full chip SoC UPF contains subsystem UPF, the same subsystem UPF can be re-used for derivative projects

  e.g. load_upf /soc/power/UPF/pasim/serial_interface.upf -scope u_serial_top

Avoid C-Test for Ret/Non-Ret Verification

- Avoid using C tests for retention/non-retention register Read/Write verification to increase randomization capability and re-usability
- Instead use SV tests for retention/non-retention register Read/Write verification

Conclusion

- Power intent based PA simulation is an efficient way for verifying the power management scheme of a SoC
- It can be done at the very earlier stage of verification cycle on RTL
- It gives faster simulation results with lesser debugging effort
- RTL level power related test cases can be re-used for PA verification too
- The PA verification is used to find power sequencing/power partitioning/power structure related bugs
- This UPF based PA verification can be done at Gate level also, using a netlist with PG connections
- The PA simulation at Gate level helps to make a slightly stronger verification for the SoC power scheme

Questions?