Power Aware Verification Strategy for SoCs

Boobalan Anantharaman
Elect Design Engineer Staff
Cypress Semiconductor Technology India Pvt. Ltd.
65/2 Bagmane Tech Park,
C.V. Raman Nagar, Bangalore, INDIA.
(91)-80-6707-3102
boh@cypress.com

Arunkumar Narayanamurthy
Elect Design Engineer Staff
Cypress Semiconductor Technology India Pvt. Ltd.
65/2 Bagmane Tech Park,
C.V. Raman Nagar, Bangalore, INDIA.
(91)-80-6707-3636
akny@cypress.com

ABSTRACT

The mandate to reduce system power consumption and the energy-efficient Integrated Circuit designs led to the increasing use of low-power IC design techniques. In addition to well-established techniques like clock gating, IC designers today are using advanced techniques to minimize static and dynamic power in their designs. Some System on Chips (SoCs) today has different power domains and hundreds of power modes. To verify these kinds of SoCs, Power Aware (PA) verification is an essential one. Verification engineers must ensure that the chip functions correctly in each power domain and in different power modes. Because of this complexity, verification planning is essential for low-power designs. The Power aware verification effort should start with a measurable, executable plan that sets forth goals and priorities. This plan should guide verification efforts all the way to verification closure, which occurs when goals are met.

The traditional approach to verify the power management logic in a SoC is running the simulations on Power-Ground (PG) connected Gate Level Netlist. Since PG connected Gate Level Netlist is needed for verification, the power management verification will be done very late in the verification flow. To do the power management verification at an earlier stage (e.g. RTL level), a power intent format based approach is required. Languages such as the Unified Power Format (UPF) or the Common Power Format (CPF) were designed to meet this need. This paper will provide a verification strategy for SoCs using UPF.

The Unified Power Format (UPF) standard provides an easy and effective way of describing power intent of the design and hence introduces power reduction techniques in the Register Transfer Level (RTL) stage itself. The same UPF used in the RTL level will be used for Power Aware verification flow too.

This paper presents the PA verification strategy for SoCs, where design and verification operate from a common UPF standard. This paper focuses only on digital power aware verification. This paper talks little bit about the UPF. This paper mainly explains about Power aware verification flow, Power aware verification planning, the faster ways to find bugs, the effective ways to debug power aware simulation failures, Re-usability of RTL test cases in PA simulation, and Re-usability of PA TB suite across different Low power SoCs.

Keywords:

SoC, RTL, TB, UPF, AHB, PA verification, GL Netlist.

1. INTRODUCTION

Power aware verification plays an important role in verifying Low power SoC designs. Low power SoC designs have different power domains and power modes. Figure 1 shows a sample SoC with different power domains. Power domain is a collection of IP instances that typically share a common supply voltage. For example, the IPs work under supply voltage (1.2V) will be grouped into one power domain. Power modes reflect power domains being in a fixed combination of predetermined states.

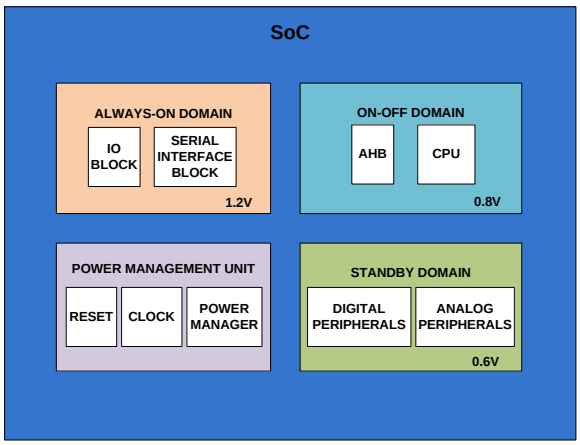

Figure 1 SoC with different power domains

The SoC shown in the Figure 1 has three power domains named ALWAYS-ON domain, ON-OFF domain, and STANDBY domain. This SoC has three power modes named active mode, low-power mode and standby mode. The IPs like IO block and Serial Interface block belong to the ALWAYS-ON domain. The CPU and the Advanced High-performance Bus (AHB) protocol interface belong to the ON-OFF

domain. The other digital/analog peripherals are grouped under STANDBY domain. The power management unit provides reset and clock in active mode of this SoC and it provides power control signals in active and low-power modes of this SoC.

The traditional approach to verify the power management logic in a SoC is running the simulations on Power-Ground (PG) connected Gate Level Netlist. Since this approach is based on Gate Level Netlist, power management verification will be done very late in the verification flow and also the defect rectification takes lot of time. Gate level simulations are very slow compared to RTL and also it is hard to debug the issues with a gate level netlist.

The power intent based power aware verification is a better approach than the traditional approach to verify the power management logic in a SoC. With this approach, power management verification is done at the early stage of verification flow. Since the power intent based verification can be done at RTL stage itself, power architecture related bugs can be caught earlier. It is easier to debug the issues at RTL level. This power aware verification approach uses the Unified Power Format (UPF) standard to describe power intent of the design.

Unified Power Format is standardized as IEEE 1801-2009. Most leading vendor tools support UPF and it is HDL independent.UPF provides the concepts and notation required to define the power management architecture for a design. A UPF specification can be used to drive the implementation of power management for a given design, during synthesis or subsequent implementation steps. A UPF specification can also be used to drive verification of power management, during RTL simulation and gate-level simulation. The ability to use UPF in conjunction with RTL simulation enables early verification of the power management architecture. The use of UPF eases verification by enabling reuse of power management specifications throughout the verification flow.

Isolation cells and Retention cells play a vital role in low power designs. The Isolation cells ensure that when a domain is turned off, its output will have some pre-defined or latched value; this is how other active domains are not affected by turning domains off. The UPF defines the Isolation strategy for different power modes. Some registers need to retain their values after being turned-off. These are called retention registers, which store their previous active value after being shutdown and re-store their previous active value after wake-up. These registers are important for fast wake-up. When two blocks work under different voltage level communicate with each other, a level shifter is needed. The UPF defines this kind of level shifting strategy as well. Using the UPF, the Power aware verification should verify the following scenarios in the low power SoCs,

- Shut-down and turn on the power of each IP in the SoC.
- Shut-down and turn on the power domains of each IP according to its power modes.
- Shut-down and turn on memories, with and without value retention.
- Shut-down and turn on registers, with and without value retention.
- Check the Isolation cell outputs when the IP is in shutdown.
- Check that the active logic is protected from the turned-off IP by the Isolation cells.

2. POWER AWARE VERIFICATION FLOW

This section explains about the power aware verification flow and the Unified Power Format (UPF) which is used in the PA verification flow.

2.1 Low power SoC Verification flow

The low-power verification flow begins when power architecture is defined for a design. The PA verification flow uses UPF, Power aware libraries, and RTL or netlist for verification. Figure 2 shows the SoC Power aware verification flow diagram.

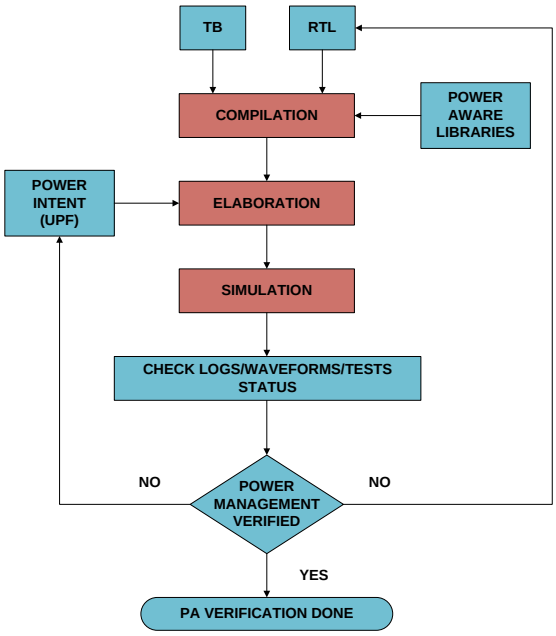

Figure 2 SoC Power Aware verification flow

The above flow diagram shows the power aware verification on RTL. In the power aware verification flow, first the RTL, TB and Power aware libraries will be compiled. To the verification team, the power intent specification file (UPF) will be delivered by the

designer after doing Multi Voltage Rule Check (MVRC). MVRC checks whether the supplied UPF file and design file's power connections are correct. It also checks for missing Isolation/Level shifter cells and redundant Isolation/level shifter cells in the UPF. The MVRC is done by the design team. So this is not part of the PA verification flow shown in the Figure 2. The UPF will be modified to make it vendor specific format (this is specific to our PA verification flow). Then elaboration will be done with the UPF and some set of sanity PA tests will be run to ensure about the correctness of the UPF. After this, all power aware tests will be run on the RTL. Once the simulation is done, the simulation results like logs and waveforms will be checked against the power management specifications of the design. The power controller logic is checked through the scoreboard which has the prediction logic for comparison. The power mode sequences correctness is verified through assertions and scoreboard checks. Power mode transitions are verified through coverpoints in the coverage model. When a block is powered down, output values after isolation is checked through assertions. The Analog and digital Hard IPs outputs are verified through assertions. Based on the power modes, 1->0 and 0->1 transition on the voltages supplies will be checked through assertions/checker.

If all the power management functionality passes with assertion and scoreboard checks, the power aware verification will be closed. If failure occurs due to UPF issues, then UPF needs to be updated and the flow starts from elaboration again. If failure occurs due to RTL bugs, then RTL will be fixed and the flow starts from the beginning again.

This power aware verification flow is used to run many scenarios from the power aware test plan, where the IPs and power domains are shutdown and turned-on. This flow is used to verify Power-On-Reset (POR) sequence, power mode switching, Isolation, Retention and Restore logic, and so on.

Following are the benefits of power intent based verification flow,

- Power architecture related bugs can be caught early in the verification cycle.
- Debugging effort is saved when compared to Gate level debugging because PA verification is done at RTL level in this flow.
- Power strategy related thoughts start much earlier in the SoC design cycle via UPF.
- It qualifies the PG connections of the behavioral models used for PA verification.

Following are the limitations of power intent based verification flow,

- Power intent files creation.
- Tool dependency on UPF interpretation.

Figure 3 shows a sample power aware simulation result. The waveform in the Figure 3 is an example for the active mode to low power mode switching and low power mode to active mode switching. Here the SoC is in active mode first and doing Advanced High-performance Bus (AHB) protocol transactions. Then it enters into the low power mode. The "sleeping" signal is high when the SoC is in low power mode. Then the SoC wake-up happens through a serial protocol based interrupt. Once the SoC wakes up, it does serial protocol based transactions with external devices.

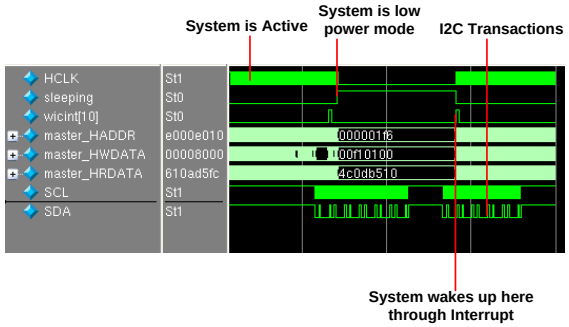

Figure 3 SoC Power aware simulation result

2.2 Unified Power Format

For Power aware verification flow, power intent (UPF) is an essential one. An architect will create a system-level power intent specification file (UPF) to define the top-level power domains and power states of a design. This UPF file will typically include information that identifies the following,

- Major power domains.
- Use of power shutoff, retention, isolation, level shifting, multiple supply voltages, and other techniques.
- Major modes of operation.
- Power-up restoration policies.
- Interface requirements between domains.

This same UPF specification will be used in the Power aware verification flow with some vendor specific modifications (specific to our PA verification flow). Most of the SoC UPF defines the following concepts and terminology for specification of power intent:

2.2.1 Power Domain

Power Domain is a collection of library cells and/or HDL module instances that typically share a common primary supply and typically are all in the same power state at a given time.

2.2.2 Power State

Power state is an abstract representation of the voltage characteristics of a power supply, and also an

abstract representation of the operating mode of the elements of a power domain.

2.2.3 Isolation

Isolation is a logic that mediates the interaction between logic elements that are in different power states. Isolation also protects elements that are powered up from the elements that are powered down.

2.2.4 Level shifting

Level shifting is a logic that mediates the interaction between logic elements that are powered up at different voltage levels. Level shifters are included to ensure that logic values generated by one element at one voltage level are correctly translated to the appropriate voltage level of the other element.

2.2.5 Retention

Retention is a logic that enables the state of selected state elements to be saved when a domain is powered down and restored when the domain is powered up again.

Figure 4 shows a SoC Power intent (UPF) example. It consists of power domains, power states, isolation, level shifting, and retention specifications.

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

/***** 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}

/**********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
```

Figure 4 SoC Power intent specification example

3. POWER AWARE TESTBENCH ENVIRONMENT

This section explains about a sample power aware test bench environment for a SoC. Figure 5 shows a sample system verilog (SV) and Open Verification Methodology (OVM) based power aware test bench environment for a SoC. This Power aware test bench environment has a serial protocol agent to initiate serial protocol based transactions, a Reset agent to generate external reset, a clock agent to generate clock, and Agent1 to drive/sample other IOs of the SoC.

The SoC processor related transactions like AHB writes/reads to registers are initiated through a C-Test and the external agent related sequences like reset sequence, clock sequence and serial protocol sequence are initiated through SV-Test. The scoreboard shown below checks the data integrity; status bits functionality and interrupt bits functionality. The coverage model collects the functional coverage of the SoC. The Power aware simulation is run using the UPF shown below on the SoC RTL. The Power aware test used in this environment constitutes two parts. One is the C-Test which configures the SoC, puts the SoC into sleep, polls/reads the interrupt bits or status bits and clears the interrupt bits or status bits. The other one is the SV-Test which triggers the

reset sequence, clock sequence and serial protocol sequences. The handshaking between the C-Test and SV-Test should be proper, so that the power down, power up sequence will function correctly.

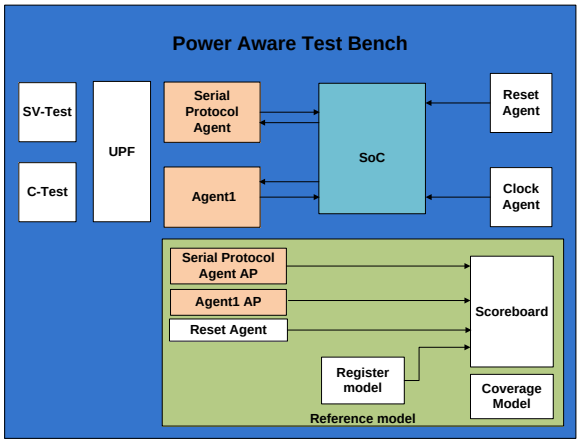

Figure 5 Sample SoC Power aware Test bench environment

4. POWER AWARE VERIFICATION PLANNING

This section explains about the power management bugs that can be caught through power intent based PA verification and power aware verification planning.

4.1 Power Management Bugs

The following types of power management related bugs can be caught by power intent based power aware verification,

4.1.1 Control Bugs

The following are the control bugs,

- Power down/power up control sequence related bugs
- Power state transition and sequencing related bugs
- Failure to reset after power down/ power up

4.1.2 Partitioning Bugs

The following are the partitioning bugs,

- Bugs due to incorrect implementation of system power modes
- Bugs due to cyclic domain state interdependencies

4.1.3 Power Bugs

The following are the power bugs,

- Bugs due to incorrectly structured power switching network
- Bugs due to incorrect powering of logic elements

4.1.4 Structural Bugs

The following are the structural bugs,

- Bugs due to missing Isolation cells
- Bugs due to missing, incorrect or redundant level shifters
- Bugs due to missing retention registers

The power aware verification plan should address all the scenarios which can catch all these power management related bugs.

4.2 Power aware verification planning for SoCs

Normally a power aware verification plan covers functional requirements, design requirements, coverage goal definition, and list of assertions to be checked. The PA verification plan should cover the functional requirements like “wake-up from a low power mode through protocol/IO interrupt and do transactions with the external serial protocol devices” (application scenarios). The PA verification plan should address the design requirements like all the power domains, power modes, Retention logic, Non-Retention logic, Isolation logic, mode transitions, power down and power up sequence logic in a SoC. The PA verification plan should address the coverpoints related to all the power mode transitions, configurations, types of resets and wake-up interrupts in a SoC to define functional coverage goal. The PA verification plan should have assertions to check the power mode sequences correctness, the output values after isolation, and the outputs of Analog and digital Hard IPs. The power aware verification plan must cover the verification scenarios which check the following:

- 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 a known value before isolation is removed?
- Can activities resume correctly after power up?
- 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 meet performance requirements while repeatedly powering up/ powering down its components?
- 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/power 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?

All the functional/design requirements specified in the PA verification plan should be thoroughly verified without any test case failures and all the PA coverpoints/assertions specified in the PA verification plan should be hit for the PA verification closure. The PA Functional coverage and PA Assertion coverage should be 100% for the PA verification closure.

5. TIPS FOR CATCHING BUGS FASTER

The following are the tips for catching bugs faster in the power aware verification cycle,

- 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 in the verification scenarios. Check whether the chip wakes up from Low power mode and enters active mode properly with wake-up sequence.
- Route IP outputs to SoC IO pads in Low power mode if SoC design supports this feature. Check whether all the signals can be routed to IO pads in the Low power mode through the assertions or through self checking logic in the test case.
- Check all the wake-up sources in low power mode. Check whether chip wakes up from Low power mode and enter active mode properly with wakeup sequence.
- Check Retention/Non-Retention logic in all the low power modes. Check whether the values written into registers and the values read from the registers are matching after exit from Low power mode by using the self checking logic in the test case.
- In all the power modes, remove the main power supplies (vddd/vdda) of the chip. This will make the chip to enter into illegal state. When the main power supplies (vddd/vdda) are inserted back, chip gets reset. Due to reset the chip should recover back to active mode. Using this scenario, bugs related to power sequencing issues can be caught earlier. Figure 6 shows the Power down and power up sequence.

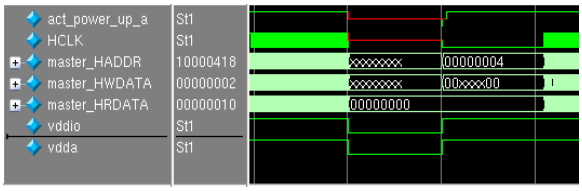

Figure 6 Power down and power up sequence

- 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. Using this scenario, low power mode entry/exit during Serial/Parallel interface protocol transaction related bugs can be caught.
  Note: This scenario is specific to the SoCs which supports Serial/Parallel interface communication during Low power mode entry and Low power mode exit.
- Check the low power mode entry and exit when SoC is operating at Maximum, Minimum, and Middle range system clock frequencies.
- Add assertions using power ports to check the port values in different power modes. This helps to find the bugs faster.
- Add assertions for power mode transitions to find the mode transition related bugs faster.
- Add power mode transition timing related assertions to find the bugs related to timings, for when the specification defines the maximum and minimum time for different power mode transitions.
  For example, ACTIVE->SLEEP transition (Min time: 1us, Max time: 3us)
- Add display statements about mode transitions and about handshaking between the C-Test and SV-Test with OVM_NONE as verbosity, so that debugging will be easier using the log file.
  For example,
  ovm_report_info(get_type_name(),$psprintf("PA_SIM: DEVICE is in ACTIVE mode"), OVM_NONE);

6. DEBUGGING TIPS

The debugging tips explained in this section are related to power aware verification issues. These may vary depending upon the EDA vendor and tool supporting capabilities.

6.1 Debug using RTL reference waveform

Since the Functional Verification test cases can be re-used for PA verification, 'X' propagation related issues and power-up, power-down sequence related issues can be debugged using Functional simulation waveforms as reference. If some test case fails in PA simulation, run the same test in RTL functional simulation and get the reference waveform of functional simulation for easier debugging. Using this method, the debugging effort will be reduced a lot. The Figure 7 shows the PA simulation result with Failures. Figure 8 shows the Reference RTL Functional simulation result.

Figure 7 PA simulation result with failures

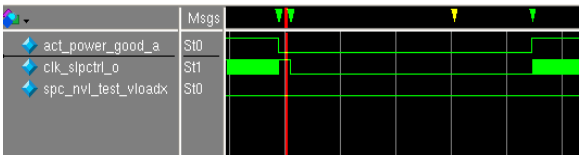

Figure 8 Reference RTL Functional simulation result

6.2 Debug 'X' Propagation using Back Tracing

If a test fails in the PA simulation due to "X" propagation, this is may be due to conditions below,

a) Isolation values are not clamped properly due to incorrect assertion or de-assertion of Isolation control signals.
b) Power supplies are not proper during power mode transitions.

The "X" propagation issues in the PA simulation can be debugged by back tracing the "X" propagating driver by using the tool supporting commands, so that the "X" driving block and signal in a SoC can be identified easily. Most of the tool vendors supports back tracing feature. The 'X' propagation during simulation due to unknown(e.g. due to multiple drivers) can be debugged through back tracing feature or by running the same simulation in the Gate Level PG connected netlist without UPF. Figure 9 shows the snapshot of the driver back tracing result.

```
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
#
```

Figure 9 Driver back tracing result

'X' propagation can also be back traced using a dataflow diagram. Figure 10 shows the dataflow diagram which shows the 'X' propagation path.

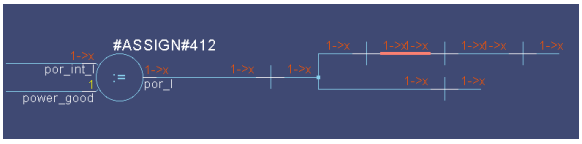

Figure 10 'X' propagation data flow diagram

6.3 Debug using waveform loadable format

Initial Smoke test bring-up of Power aware simulation is a very tough process.

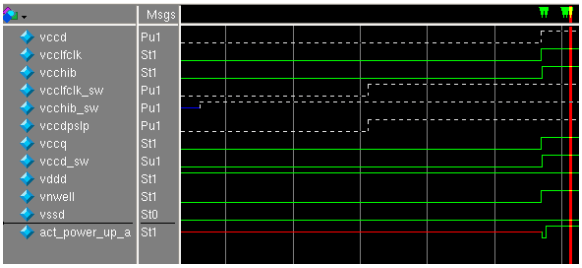

Figure 11 Waveform with all power supplies in a SoC

To make debugging easier save all the important power supplies in the SoC as waveform loadable format (e.g. .do format). Using this, for any new UPF release, sanity testing of power supplies and debugging can be done faster. Figure 11 shows a sample waveform with all the important power supplies in a SoC.

6.4 Tool issue debugging

Sometimes tools may drive the clamp values incorrectly, due to which 'X' propagation can happen. Such low power 'X' propagation can be identified by manual inspection (through waveforms) or by hacking the clamp values. When a test case fails due to low power 'X' propagation, to justify that as a tool issue, forcibly drive the clamp value to 0/1 from the test bench and run the test again to conclude the root cause of the problem faster.

Sometimes the test case may fail due to supply on-off issue. If such failure is suspected as a tool issue, run the same test case on the PG connected Gate Level Netlist. Check the test case "pass or fail" status to conclude the issue as tool issue at the earlier stage.

6.5 Add Self checking logic

Add self checking logic in each of the test cases to make sure about proper power mode transitions. This will help in faster debugging.

For example,

```
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);
```

7. RE-USABILITY TIPS

The following are the tips for power related tests and power aware test bench suite re-usability,

7.1 Re-use RTL Power related tests

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

7.2 Create Generic scripts

To reduce the manual effort, Power aware verification engineers can create generic scripts to convert the UPF given by the design team to vendor tool specific format.

Note: The above re-usability tip is specific to our PA verification flow. In our flow, the UPF delivered by the design team will be modified to do Ground connections and for defining the top-level module names using script.

The UPF and PA test cases can be re-used across different derivative products using scripts.

7.3 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 reused across GLS/PA/derivative projects.

The following sample code shows a test case with delay,

```
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 be calling this global stop request.
  global_stop_request();
endtask : run
```

The #4.5ms delay in the code above may not be valid in Gate level simulations due to which the test case may not work as expected in the gate level simulations. The following sample code shows the same test case coded using event for re-usability,

```
task pa_active2dpslp_wdt_reset_ctest::run ();
  super.run();
  poll_active_mode.wait_trigger();
  ovm_report_info(get_type_name(),"In Deepsleep second
watchdog reset occurred and chip entered active",
OVM_NONE);
  global_stop_request();
endtask : run
```

7.4 UPF re-usability

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

For example,
load_upf /soc/power/UPF/pasim/serial_interface.upf -scope u_serial_top

7.5 Avoid C-Test for Retention/Non-Retention verification

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

Instead use System verilog tests for retention/non-retention register Read/Write verification for better randomization and re-usability.

8. SUMMARY

Power intent based power aware simulation is an efficient way for verifying the power management scheme of a SoC, especially when the power management scheme is a very complex one. It gives faster simulation results with little effort. It can be done at the very earlier stage of verification cycle using the RTL. The debugging effort is lesser since the verification is done at RTL level. Since the UPF based simulation can be done at RTL level, the power related test cases used for RTL verification can be re-used for power aware verification too. The power aware verification is used to find power sequencing related bugs, power partitioning related bugs, power structure related bugs, and so on.

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. But power intent based Gate Level Simulation (GLS) is a very slow simulation and it can be done only at the final stages of the verification cycle because a netlist which contains all the PG connections and without timing violations is needed.

9. ACKNOWLEDGEMENT

The authors are grateful to the supportive family members and the Cypress Management.

10. REFERENCES

[1] Unified Power Format, IEEE Draft Standard for Design and Verification of Low Power Integrated Circuits, IEEE P1801/D18.
[2] Freddy Bembaron, Sachin Kakkar, Rudra Mukherjee, Amit Srivastava: “Low Power Verification Methodology Using UPF”, DVCON 2010 Paper
[3] Amit Srivastava, Rudra Mukherjee, Erich Marschner, Chuck Seeley, Sorin Dobre: “Low Power SoC Verification: IP Reuse and Hierarchical Composition using UPF”, DVCON 2012 Paper
[4] Prashant Bhargava: “CPF based Power Aware Verification”, CDNLive 2007 Presentation
[5] Barry Pangrle: “Low Power SoC Verification with a UPF-Based Flow”, EDA Tech Forum 2009 Presentation
[6] ARM Based SoC Verification from Doulos