

Low Power Verification Methodology Using UPF

Freddy Bembaron
Texas Instruments Israel
freddyb@TI.com

Rudra Mukherjee
Mentor Graphics
rudra_mukherjee@mentor.com

Sachin Kakkar
Mentor Graphics
sachin_kakkar@mentor.com

Amit Srivastava
Mentor Graphics
amit_srivastava@mentor.com

Abstract—Power aware verification has become an increasingly critical issue for the semiconductor industry. Shrinking process geometries have led designers to use various approaches to minimize static and dynamic power and has put immense burdens on verification teams to ensure power aware verification is complete.

This paper provides a comprehensive, holistic approach to power aware verification, where design and verification operate from a common, consistent basis for defining power intent using the latest IEEE P1801 Unified Power Format (UPF) standard.

While it is possible to use ad hoc methods to verify these design elements, the methods are often far from ideal and can be slow, cumbersome, and incomplete. This paper emphasizes how UPF enabled verification tools provide a superior solution and can add tremendous value to low power verification.

This paper will cover in detail various power management design techniques and the specification of power state tables, switching networks, signal isolation, state retention, and restoration. We will also share our experiences on some of key challenges in power management validation, utilizing examples that incorporate state-of-the-art power management techniques.

This paper also illustrates an all-inclusive checklist and ways for verification engineers to use built-in (provided by EDA tools) as well as user-specified static and dynamic rule checks to validate power intent and the specification of low power designs.

I. INTRODUCTION

In real world mobile applications, it is essential to save power in parts of the chip that are not in use. Nowadays, chips integrate several systems on a single chip (SoC); each of these SoCs is also called intellectual property (IP). In order to save current consumption, each IP can move between power modes (power-off/power-on/sleep). Each IP is divided into power-domains, and those power domains can be turned on and off as well, according to the power-mode. Memory and register values can be restored after being shut-down (also called

retention). There are isolation cells which keep the turned off IP outputs in a previously defined value, and this is how the shut-down IP does not corrupt other active IP functionality. Our power-management verification methodology verifies the functionality of all the power-management elements described above.

II. POWER MANAGEMENT

The power management scheme is built by the power architect of the SoC in order to reduce the current consumption of the chip. There are several common building blocks which are used by the power management architect, as will be described in the next sections.

A. Power Domains and Switching

In many low power designs, parts of the design have a switchable supply. Those parts are called switchable domains. In our example designs, most of the memories and all of the IP have a switchable supply. The power-management module is the one that controls the power-up and power-down sequences. It toggles the control signals to each of the analog switches according to the power sequence and allows current to relevant power-domains by closing analog switches. The main idea of using power-switches is to turn off massive unused parts of the design and, as a result, gain low current consumption.

B. Retention

Memories can be switched on and off in order to reduce their current consumption. Memories are shut-down when their IP or power-domain is in shut-down mode or when they are not in use. Some memories need to retain their values for fast wake-up. For these memories, only the memory array stays powered on using a low power mode, and the peripheral interfaces are powered off. The retention mode consumes a little more current than power-off mode but allows fast recovery of the memory content after waking from sleep mode.

A register’s power is turned off when its power-domain or IP is turned off. Some registers also need to retain their values after being turned off. These are called retention registers, which restore their previous active value after being shut-down. These registers are important for fast wake-up.

C. Isolation

Powered-off domains and IP do not drive their outputs anymore, and their outputs become floating nodes. This might be a problem when some other active module gets those floating nodes as an input. This unexpected signal value might cause high current consumption because of an inappropriate logic level, or it might cause improper logic behavior of the active module because of the “gibberish” inputs coming from the turned off domain. In order to solve this problem isolation cells are placed between power domains. The isolation cells are powered by a constant supply and drive 0, 1, or latch the old value of the turned off domain. The isolation cells ensure that when a domain is turned off, its output will have some predefined or latched value, and this is how other active domains are not affected by turning domains off.

III. UNIFIED POWER FORMAT

The Unified Power Format (UPF) [1] enables designers to easily specify the power intent early in the design process — at the register transfer level (RTL) or earlier. Since traditional hardware description languages (HDL) are not adequate to specify the power design information, UPF provides a consistent format without impacting the existing HDL. UPF is designed to be used throughout the flow, so what is implemented is the same as what has been verified.

The power intent specification consists of power domains, supply networks, power aware extensions to HDL, and power states.

A. Power Domains

The basic unit of a power management strategy is the power domain. It is a collection of instances of the user design that share the same primary supplies, like power and ground. UPF provides a command `create_power_domain` for creating a power domain and associating the design instances with the extent of that power domain.

B. Supply Networks

In order to specify the low power design constraints, it is required to specify a power supply network that can control the distribution of that supply to minimize the energy consumption. Using UPF, one can easily specify the network at an abstract level. This network consists of supply ports, supply nets, and power switches, and is a high-level abstraction of the electrical network of the chip. Supply ports provide the supply interface to the power domain and switches, whereas supply nets connect the supply ports. Since the supply network is specified apart from logic design, the logic design specification remains independent of a specific power supply network specification.

C. Power Aware Extensions to RTL

UPF also defines extensions of the logic design with power-specific capabilities without modifying the original logic specification. It enables designers to impart retention capabilities to the existing registers in the design and specify the control signals governing the retention behavior. UPF also has commands that impart isolation and level shifting capabilities in the logic design along with the control signals governing the isolation behavior.

D. Power States

Any power management strategy starts by defining the list of power states. These power states determine the mode of operation of the domains and their corresponding simulation semantics. UPF allows users to define power states and associate it with power domains, supply nets, supply ports, and supply sets.

IV. POWER MANAGEMENT AT THE CHIP LEVEL (CASE STUDY)

Figure 1 demonstrates how the power management elements should be connected at the chip level.

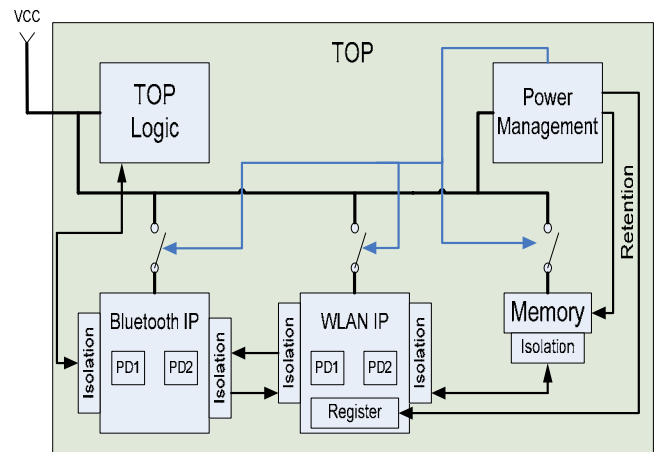


Figure 1. Placing power-management elements at the chip level.

In this example, there are two IPs: WLAN and Bluetooth. Each one of them can be turned on and off separately. In order to avoid corrupted signals coming from a turned-off driver, both WLAN and Bluetooth outputs are shielded with isolation cells. Each IP has a few power-domains which can be activated by the IP itself upon demand. The top power-management module drives the controls in order to close the power domain switches. It also drives the retention signals to retain the memory and register content when needed. There is some always-on logic which gets a constant supply. In this example, the always-on-domains are the top logic, the power-management module, and the isolation cells.

```

set_scope top_aod
#####
#PD_BT Power Domain
#####
create_power_domain PD_BT -elements { btip_i }
create_supply_port VCC -domain PD_BT
create_supply_port GND -domain PD_BT
create_supply_net VCC -domain PD_BT
create_supply_net GND -domain PD_BT
create_supply_net PD_BT_primary_power -domain PD_BT
#####
## connect supply ports to supply nets
#####
connect_supply_net VCC -ports { VCC }
connect_supply_net GND -ports { GND }
set_domain_supply_net PD_BT -primary_power_net
PD_BT_primary_power -primary_ground_net GND
#####
# Header switch for PD_BT
#####
set_power_switch pmcore/pwrswitch \
-domain PD_BT \
-output_supply_port { out_sw_PD_BT PD_BT_primary_power
} \
-input_supply_port { in_sw_PD_BT VCC } \
-control_port { ctrl_sw_PD_BT1 } \
-control_port { ctrl_sw_PD_BT2 } \
-on_state { normal_working in_sw_PD_BT {
((ctrl_sw_PD_BT1 == 1'b1) && (ctrl_sw_PD_BT2 == 1'b1)) } } \
-off_state { off_state {!(ctrl_sw_PD_BT1 == 1'b1) ||
!(ctrl_sw_PD_BT2 == 1'b1)}}
#####
# isolation Strategy for PD_BT
#####
set_isolation PD_BT_isolation -domain PD_BT -
isolation_power_net VCC -clamp_value 0 -applies_to both
set_isolation_control PD_BT_isolation -domain PD_BT -
isolation_signal pm_bt_iso -isolation_sense high -location
parent
#####
# Retention Strategy for PD_BT
#####
set_retention PD_BT_retention -domain PD_BT \
-retention_power_net VCC

set_retention_control PD_BT_retention -domain PD_BT \
-save_signal { btip_i/soc/ret_i posedge } \
-restore_signal { btip_i/soc/ret_i negedge }

map_retention_cell PD_BT_retention -domain PD_BT \
-lib_model_name CLRFF -lib_cell_type FF_CKHI

#####
# Always On (AOD01) Power Domain
#####
create_power_domain AOD01 -elements { btip_i/soc/fdo_o }
create_supply_net VCC -domain AOD01 -reuse
create_supply_net GND -domain AOD01 -reuse
set_domain_supply_net AOD01 -primary_power_net VCC -
primary_ground_net GND

```

Figure 2. UPF Example

V. VERIFICATION GOALS

The power-management goals are to verify the functionality of all the power-management elements and see how they all work at the system level. We would like to specifically focus on the following scenarios:

- Shut-down and turn on the power of each IP.
- 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 IP is in shut-down.
- Check that the active logic is protected from the turned-off IP by the isolation cells.

VI. VERIFICATION CHECKLIST

A. Proper Power Control Network

In any power management strategy it is important that the power intent is properly captured into a UPF file and communicated to the verification tools. The designer can then statically determine the correctness of the power network by the following:

1) Static Reports

The tool generates static reports which list all the relevant information, such as:

- Power domains and what portion of the RTL design hierarchy is affected by it.
- Isolation cells and the signals that are isolated by them.
- Retention cells and the models that they are mapped with.

2) Static Checks

The tool can also perform static checking [2] of the placement of isolation cells and level shifters and can notify the user with the following information:

- Incorrect type of level shifters present at the power domain boundary. e.g. The UPF says that high_to_low kind of level-shifters will be placed at the boundary, but after static analysis of the power intent, it is found that the voltage swing from moving from one power domain to the other is in the reverse direction.
- Missing isolation and level shifter cells.

B. Power Control Sequence Protocol Violation

This is the most important check that the designers need to ensure the correctness of the power controller [2]. The important things that they need to check are:

1) Dynamic Checks

- Isolation-enable is triggered before the power to the domain goes down. It remains active throughout the power down period and until sometime after the power goes up.

- If the power domain exhibits retention capability, then its controls are triggered at the right time. For example, retention SAVE is triggered before power down, and RESTORE is triggered after power up.
- 2) *Dynamic Reports*
- Power domain transitioning at the proper simulation time.
 - Activity on the switch controls and the state of the switch.
 - Activity on the controls of the isolation cell and the retention cells.

C. Incorrect Retention Behavior

Today's designs exhibit complex retention capabilities consisting of complicated sequencing of various control signals. A typical retention strategy involves two basic operations – SAVE and RESTORE. A SAVE operation is the one which causes the register to save its content prior to power shut-down. A RESTORE operation is the one that causes the restore of the saved value after the power is switch on. These operations could be dependent on complex sequences of various control signals of the register.

Any incorrect state of the register controls like clock, set, reset or violation of power control sequence (like the control signals causing SAVE operation or RESTORE operation) could cause incorrect retention values on the retention register or memory. Thus it is necessary to verify the proper retention behavior through simulation to ensure proper behavior after power-up. One of the items to look for is:

- Retention absent in retention register

This could be caused by either a failed SAVE operation or a failed RESTORE operation. A failed SAVE operation results when the current value of the register could not be saved because of some protocol violation. Similarly, a failed RESTORE operation could be because of an earlier failed SAVE operation or due to another protocol violation during the RESTORE operation.

D. Non-essential RTL Activity

There are some scenarios in power management where it is desired that there should not be any activity in the domain which is switched off or under some sort of body biasing. It is required to detect any activity on the inputs and notify the user of such activity through assertion failures.

E. Power Control Signal Corruption

It could be possible that there is some control signal passing through a domain that gets switched off. As a result, the value on such a signal will be corrupted. It becomes essential to catch such scenarios and notify the designer so that they can take appropriate action.

VII. VERIFICATION SET UP

The verification set up involves two main steps.

A. Power Aware Compilation

A designer needs to supply the following information to the tool.

1) Power Intent (UPF)

Power intent contains all the information related to the power management structure of the design.

2) HDL

RTL design files for the Design Under Test.

3) Exclude List (Behavioral Models)

There are some models of PLL, memories, and analog parts that designers prefer to exclude from being corrupted. In our simulations, the power issues are modeled for memories in their behavioral code; therefore they need to be ignored for power-aware processing. The power-aware compilation stage requires an exclude file as an input. This exclude file contains a list of those instances in the RTL that are already power-aware and have power related functionality built into the RTL.

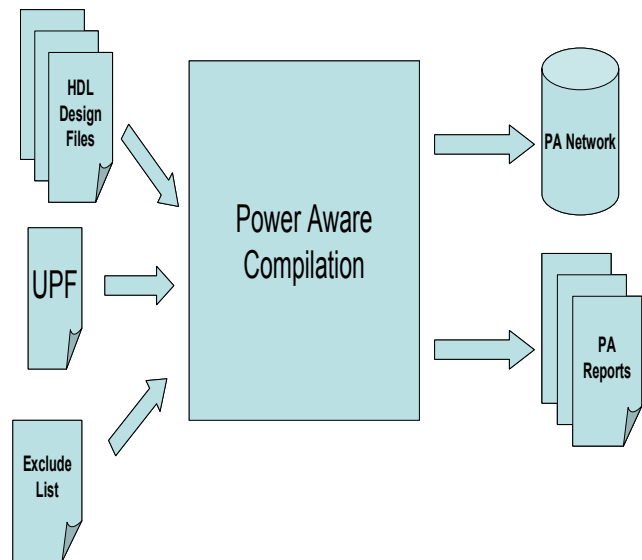


Figure 3. Power Aware Compilation Flow

Using the above supplied information, the power aware compilation creates a power control network over the design, which then is used by the simulation step to perform power aware simulation.

In addition to generating the power control network, the power aware compilation stage also generates some reports related to power intent. It also performs the static checks using the power related information from the power intent.

B. Power Aware Simulation

The second set up is used for the actual power aware simulation. We used this setup to simulate several power-aware scenarios where we switched on and off parts of the RTL according to the testcases. Whenever a power domain is shut-down, the simulator forces Xs on the shut-down domains in order to simulate their off state. It also generates some dynamic reports and performs dynamic checks.

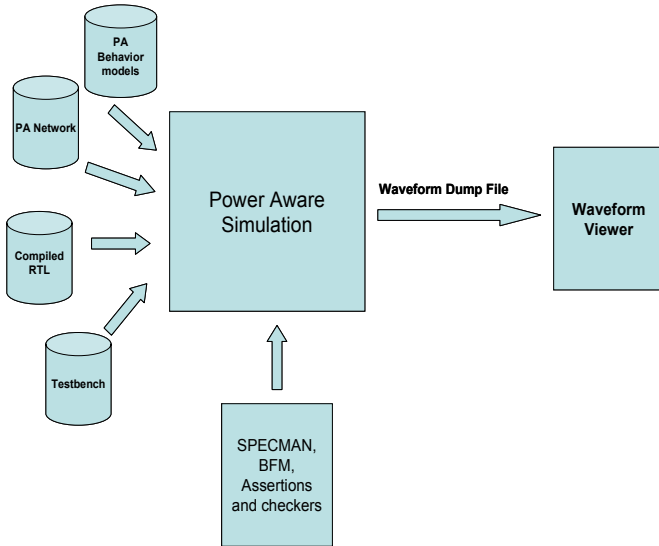


Figure 4. Power Aware Simulation

In order to run a power aware simulation, we must compile the following:

- 1) *DUT RTL*
- 2) *Testbench*
- 3) *Power aware behavioral models [3]*

These are special Verilog behavioral models that contain power aware functionality. They trigger special named events that the simulator recognizes and then performs specific actions on the target to reflect the corruption, save, and restore behavior.

- 4) *Power Aware Network*

The power aware network is one of the outputs of the power aware compilation step.

We use the power aware simulation set up to run many scenarios from the power aware test plan, where we shut-down and turn-on IPs and power-domains. We switch between one power-mode to another for several IPs in parallel. The power aware simulator automatically corrupts the IP and the power-domains once the switching logic is corrupted or zeroed. It also takes care of retentions and restores the Power-On-Reset (POR) values for relevant registers.

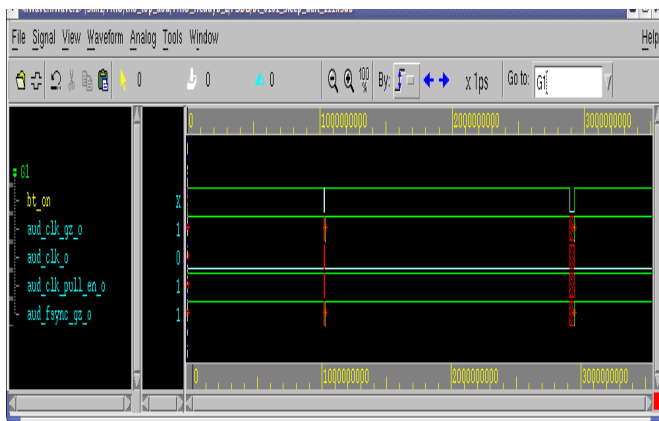


Figure 5. Power Aware Simulation Results

VIII. VERIFICATION POWER MANAGEMENT TASKS

A. IP and Power-Domain's (PD) Power Toggling

We use the power management module in order to toggle the Power-Save-Controls (PSCON). The IP and their power domains are turned off and on by the power management module and by changing the IP's power mode. The simulator inspects the PSCONs as part of the switching logic and forces Xs on the shut-down IP's outputs and sequential elements. There are functional tests for the active IP, while the other IP are in sleep and corrupted. We check that the Xs from the sleeping IP does not harm the proper behavior of the active IP.

If there is a bug in the design, the Xs coming from turned off IP will propagate to active IP and affect their proper behavior.

B. Isolation Cells (ISO)

While shutting down an IP, there should be isolation cells on its boundaries in order to reflect some legitimate value to the other domain while this same IP is in sleep. If the isolation cells are placed correctly, the Xs coming from the corrupted sleeping IP will not propagate to the other parts of the design and will be restricted to the isolation cell's boundaries. The isolation cells hold the POR value most of the time, and we check that the right value (POR) is being driven out of the isolation cell once the power domain or IP is shut-down, although the IP is driving Xs.

C. Memories (Power toggling and retention)

Memories are gathered into clusters, and each cluster can be shut-down or turned on. Some memories might have a retention option, which means that their content is preserved after there was a shut-down. The retention allows fast waking up, but consumes a little current. We would like to verify that the power and retention controls are connected properly and that the memory behaves correctly once it is turned on or off or being held in retention or not.

In our verification, we wrote values to all of the memories and then held only one cluster in retention. We shut-down all the memories and then woke them back up. Then we check that all the written values were destroyed but that of the retention memory.

The power aware feature of the simulator can be used in order to validate the functionality of a memory inside of an always on domain (AOD).

We can force Xs on the memory boundaries every time it goes into retention. If the memory content is not corrupted, then the retention feature worked for this memory.

D. Retention Of Registers

Some registers can have retention ability, which means that they preserve their value after a shut-down. The power aware simulator supports simulating this kind of behavior.

In our tests, we wrote some value to the registers, and then we shut-down their power-domain. After a wake up sequence is done, we checked that only the retention registers values are preserved. All the other registers (non-retention) should have some other value, such as a POR value.

IX. POWER AWARE ISSUES IDENTIFIED

In our last projects, we used power aware simulations massively and revealed many bugs using the power aware setup. The following paragraphs will give a short brief about the bugs we found using power aware simulations.

A. Isolation Bugs

We revealed a few missing isolation cells in the design. The lack of the isolation cells has been found when we shut-down an IP. The un-isolated driver drove an X value to the top; this X value propagated to the top logic and made it behave unexpectedly. Functional checkers in the testbench revealed the unexpected behavior in the top, which led us to a debugging session where we traced back the X's values until we found the missing isolator.

Most of the isolation bugs we found were related to an incorrect isolation value. When the drivers were powered down, the retention cells should drive 0, 1, or latch the old driver value. The design specification defines which value should be assigned in isolation mode to each one of the isolation cells. When we checked the isolation cells' outputs, we found many mismatches between the specification definitions and the design implementation.

B. Retention Bugs

We found a few registers which were supposed to be in retention, but actually did not have a retention option. Most of our retention registers were configuration registers, and if they lose their values, the relevant IP will behave differently than intended. These buggy registers were found by shutting down their power domain and turning it on without configuring those registers once again. Functional checkers in the testbench revealed the unexpected behavior in the register power domain, and a short debug revealed that the bugged register did not retain its value. The reason for that, of course, was that we did not use a retention register, but a regular register.

We also had a bug where a simple register had been implemented as a retention register by mistake. In this case, the register was in a synchronizer output driving a reset signal. The output reset signal got retained at value 1 by mistake, instead of toggling to 0 for a small period of time. The fact that the reset did not toggle caused many non-retention registers to be uninitialized to their POR value, and drove Xs instead. Functional checkers could find the unexpected behavior of these registers.

We also found retention signal switching between power domains. We found this issue by shutting down few power domains and restoring their power, revealing that the memories' values were not retained.

C. Power-Domain Connectivity Bugs

We had a power up loop bug, where our power module could not turn on another module. It happened because the turned off module drove Xs to the power module; those Xs propagated in the power module and corrupted the power controls to the buggy module. This is how we got the Xs loop. Since the power module drove Xs on its power controls the

buggy module stayed in a sleep mode and affected the whole power up of the IP.

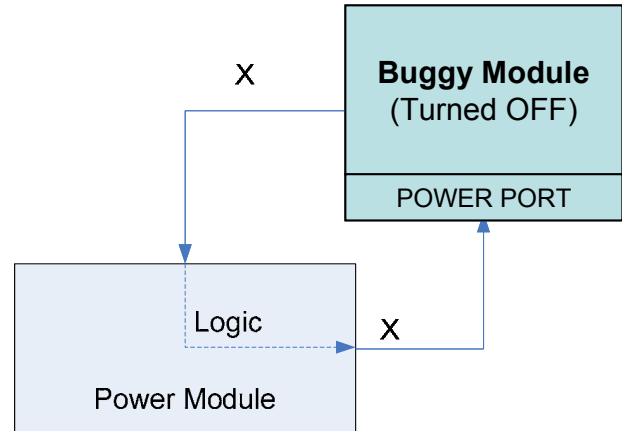


Figure 6. Power Up Bug.

In order to find this kind of bug, a designer must write his or her code in a coding style which propagates Xs, and the simulator must corrupt power domains which have Xs in their power controls, in addition to power domains which have their power controls zeroed.

X. RESULTS

The power aware simulation was a straightforward simulation. It found many bugs and gave us a good confidence in the maturity of the power management module, about the connectivity of the power controls, and about the architecture of the power scheme. We could run this simulation already in the first stages of the verification because it was based on RTL simulations for which we already had a verification environment.

The Power Aware Gate Level Simulation (PA GLS) was used later on, using a netlist with power ports. We ran PA GLS in order to make a slightly stronger verification for the power scheme. The PA GLS is a very slow simulation and it can be done only at the final stages of the project because one must have a netlist which contains all the supply connectivity and a netlist without timing violations.

XI. CONCLUSION

Power aware simulation is an essential tool for verifying the power management scheme of a design, especially when the power management scheme is a very complex one. It gives fast simulation results with little effort, and its benefits are priceless.

REFERENCES

- [1] Unified Power Format, IEEE Draft Standard for Design and Verification of Low Power Integrated Circuits, IEEE P1801/D18, 23 October, 2008.
- [2] Rudra Mukhejee, Amit Srivastava, Stephen Bailey: "Static and Formal Verification of Low Power Designs at RTL using UPF", DVCon 2008.
- [3] Stephen Bailey, Amit Srivastava, Mark Gorrie, Rudra Mukherjee: "To Retain or Not to Retain: How do I verify the states of my low power design", DVCon 2008