Case Study: Low-Power Verification Success Depends on Positive Pessimism

John Decker
Cadence Design Systems, Inc.
2655 Seely Ave.
San Jose, CA 95134
1-908-735-5358
jdecker@cadence.com

ABSTRACT

Low power has quickly become a primary requirement for a large percentage of designs. As companies rush forward to incorporate the latest low power features, they are faced with the growing challenge of how to verify these complex structures and ensure successful silicon. As with any large change in methodology, one can see the industry converging on a set of known best practices; a set of design and verification techniques that become second nature because they avoid issues, improve turn-around-time, and provide a more predictable path to the coveted first pass silicon.

This case study consolidates the experience of several customers as they evolved their verification methodology to face the unique challenges of low power design. It describes the changes that were instituted to instill the time honored verification tradition of "positive pessimism" into the flow. This principal dictates that the flow takes a conservative approach whenever possible, to ensure that the design works in all conditions and is tolerant to changes in the design environment.

The study describes a basic power shut-off (PSO) flow including an overview of power modes, state-retention and isolation. The methodology and automation changes will also be detailed including both the items that worked well and those that did not. This real-world review of the methodology transformation will enable audience members to plan their own low-power verification improvements restoring the positive pessimism that makes us so successful.

1. INTRODUCTION

Low power introduces a host of challenges to the verification methodology. It starts with the added complexity of additional low power modes of operation and the complex control and interactions between power regions. Even more fundamentally it adds power as a new aspect to the design that needs to be correctly modeled throughout the entire verification flow. This flow includes RTL and gate simulation, design and equivalency checking, verification planning and emulation. This modeling not only has to account for the synthesis of low power constructs but also the impact of physical design on the power network.

An example of this complexity is the modeling of even a simple power shutoff design. The verification process has to model the fact that this logic can be powered off by corrupting the logic at the correct time. The system has new modes of operation that need to be verified and ensure the transitions between these modes work correctly. The RTL verification environment has to model the implementation of isolation on the boundaries of domains consistent with what will be done during the implementation flow. Finally, it has to model the physical implementations of the power switch network and the delays associated with that network.

This paper will first introduce the key components of a low power verification flow. An understanding of the basic requirements and the concept of a closed loop flow is critical to successful low power verification. The remainder of the document will provide specific recommendations based on real world usage of the low power flow. These recommendations are designed to provide more detailed view of what is required for a pragmatic verification methodology that uses the concept of "positive pessimism" to ensure the best possible silicon results.

2. INTRODUCTION TO LOW POWER VERIFICATION METHODOLOGY

The low power verification is simply an extension of a typical verification flow. All of the advanced verification techniques and methodologies in a traditional verification flow should be leveraged. This includes using methodologies like UVM, verification planning and metric driven verification. The flow in figure 1 is a high level description of a typical flow from an RTL design perspective. The key components of the flow and how low power influences are described to set a context for the recommendations that are the core topic of the paper.

The flow depicted below is a simplified flow; a typical low power flow would also include simulation at higher levels of abstraction primarily to enable more system level performance analysis and application level scenarios. It would also have more detailed physical implementation and verification steps. While these are very important to low power, for the purpose of this document, we've limited the flow to just the RTL and gate level portions.

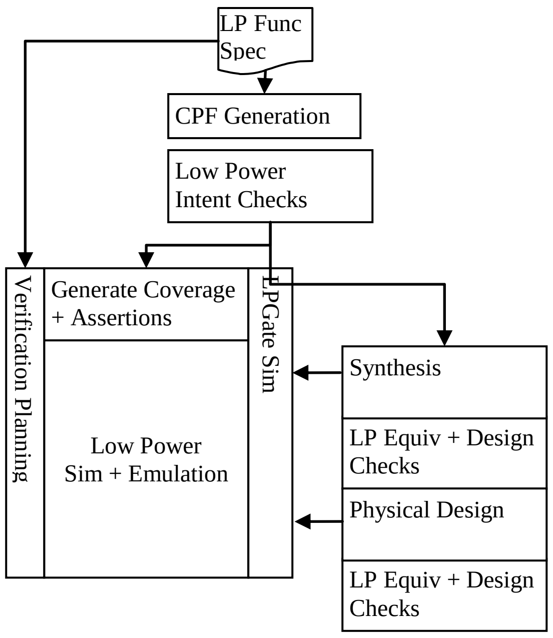

Figure 1- RTL Level Low Power Verification Flow

2.1 Initial Power Intent Creation + Checking

The flow starts with a functional specification of the low power intent. This specification is developed through a coordinated effort with system architects, implementation architects, logic designers and even marketing.

The specification is then used to generate the initial power intent in CPF or UPF for use in the rest of the design flow. The first real tool flow step is to do "Low Power Intent Checks". These checks verify the completeness and correctness of the low power intent. The checking available for the low power includes structural checking, and power intent completeness checks.

An example check is to verify proper isolation rules between power domains. A missing isolation cell is quickly identified in formal checking, but to do the equivalent in simulation would require finding the signals that went x during power shutoff and analyzing the logic cone.

Unlike simple linters, the low power intent checks are much more than a recommendation; they should be treated as a requirement of the flow. A large percentage of low power design issues are detected in a series of formal and structural checks. These checks can find issues early and before costly synthesis and simulation runs, and can greatly improve productivity and turnaround time.

2.2 Verification Planning and Metric Driven Verification

Verification planning is a critical step in any low power design. Low power architectures such as power shutoff or dynamic voltage and frequency scaling introduce new modes of operation for the design. A verification plan helps define the required scenarios to ensure these features a fully validated. This often involves planning to ensure each mode is entered and exited, but also to define what features are valid for each mode.

Metric driven verification (MDV) defines a verification plan and the corresponding set of coverage and checking metrics to validate the design. The process builds on the traditional verification flow to integrate data across the complete regression suite. One difference is that the low power intent can be used to automatically generate a coverage model and a set of assertions on power control. As the simulation runs are executed, the coverage and assertions are tracked and reflected back into the verification plan.

2.3 Power Aware RTL verification

The functional verification of the design is the next step in the flow. In a low power flow, the functionality of the design is specified by a combination of the low power intent and the RTL. The simulation and emulation engines need to model the low power intent as accurately as possible.

Dynamic simulations of the low power intent ensure the functionality of the design in the presence of the low power features. On critical area is to verify the power management and control. Often the control is part of the firmware or system software, so it is impossible to verify this logic using static checks.

The following diagram shows an example of a simple power shutoff design and illustrates the type of modeling that is required. In the past, modeling this low power intent was mostly done on through a complex set of ad-hoc scripts, PLI's,or manual coding. Today, the modeling is automated based on the low power intent files (CPF or UPF). The basic requirements are described below, but additional requirements are given in the recommendation section.

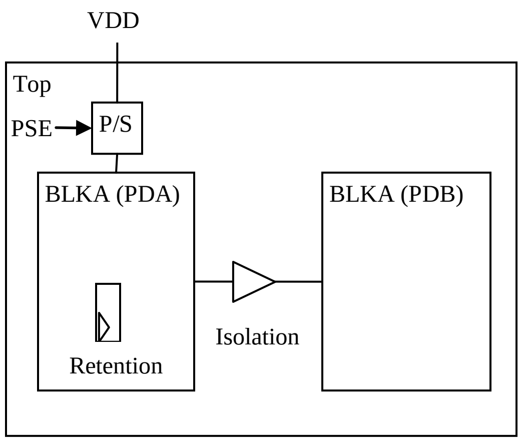

Figure 2 - Sample Power Shutoff Design

A power shutoff design requires the modeling of:

- **Power domain state and voltage**
  - State can be On, Off, standby
- **System level power mode**
  - The system mode is determined by the state of all the individual power domains
- **Power shutoff** – the P/S block is the power switch for the design, based on input PSE the power domain PDA will power off. During power off the logic inside the block needs to be corrupted to X to reflect its unpowered state
- **Isolation**: To protect the inputs of BLKB from the effects of PDA's power shutoff, special cells are inserted to clamp the

values to a valid value. These do not exist in the RTL but are inferred from the power intent

- **State Retention** – Some designs include state retention. State retention is the process of saving the state of sequential elements before power down and restoring it after power up. This is typically done by special cells that include a small save latch. Again, this is specified in the RTL, it is in the power intent.

The simulation engine models the low power intent to reflect the actual implementation. Keeping the simulation as accurate as possible is required to ensure that a successful simulation results will translate into successful silicon.

2.4 Low Power Equivalency and Design Rule Checking

After synthesis the netlist needs to be check to validate that it is an accurate implementation of the RTL and power intent. Any design transformation step in the flow is required to be validated against that power intent. This is what is referred to as a closed loop flow: All steps and stages have are verified back to the original intent and matches what was used in functional verification. Without a closed loop flow, it's possible to implement different intent then was verified.

The Low Power equivalency checking is complex; the low power intent introduces behavior that is not described in the netlist. For instance, power shutoff affects the logical output of a cone of logic, but needs to be modeled by the equivalency checker. Special cells like isolation and level shifting can have multiple domains that affect its output value, and all of this needs to be accounted for in the tool.

Finally, the low power design checks ensure the implementation followed accepted design rules and synthesis process correctly maintained the power intent. As the design moves through physical implementation additional checks are needed to verify the power and ground network and connectivity.

3. LOW POWER VERIFICATION RECOMMENDATIONS

When the results of five years of low power simulation and verification on hundreds of designs are analyzed, one can derive a set of common features that help ensure successful verification. The following section highlights these features and elaborates on modeling requirements for the power aware tools.

3.1 Early Qualification of Power Intent

As mentioned earlier, the first step of any low power flow should be to ensure that the low power intent is complete and correct. Tools like Conformal Low Power can automate this task and should be a gate keeper before proceeding to any verification or implementation tasks.

The power intent checking can statically check for a number of issues that would cause incorrect results in the simulation process.

- Missing isolation and level shifting
- Illegal mode definitions
- Missing design objects
- Incomplete power control specification
- Incomplete domain specifications
- Library consistency checks
- Power intent linting checks

The recommendation based on our customers experience is to run these low power checks early and often. Ideally, anytime the power intent or RTL change the checks should be re-run. This often viewed as overly cautious, but a quick, exhaustive check for missing isolation rules can safe hours of wasted simulation runtime. The key is that this type of static checking is exhaustive; a simulation run is only as good as its stimulus.

These checks go way beyond simple linting checks, and can detect serious structural issues. Additional details on this can be found in the case study by Luke Lang DVcon 2011 paper, "Case Study: Power-aware IP and Mixed Signal Verification".

3.2 Ensure the same Power intent is used throughout the design flow

Traditionally, the verification and implementation teams worked off of the same low power functional spec but independently modeled/implemented the low power intent. With the advent of power intent languages, such as CPF and UPF, the same intent can be used for both verification and implementation. But even today there is a temptation to have separate implementation specific and verification specific files.

Any difference in the intent files could mean that what was simulated does not match what was implemented. At one customer, the implementation team removed an isolation rule they deemed was not necessary but did not make the change to the common intent file. It turns out that the isolation was in fact needed, but this problem was not found until just before tape-out. This would have been found quickly in either the LP design checks or in RTL simulation if the power intent was properly updated.

It is very common have separate files for the general power intent and detailed implementation power intent. But the same intent should be read by all of the tools. This ensures a consistent view and has no negative impact (the front-end tools will simply ignore any information not relevant at that level of abstraction).

Whenever there is a difference in what is simulated versus what is implemented you run the risk of functional errors in silicon. These issues can be avoided with three simple steps:

- Employ revision control on the power intent files
- Ensure changes to the intent are made in the source files not in any intermediate files. Discipline is needed to avoid making changes in the output files.
- Use a tool flow that employs a closed-loop methodology that ensures the original power intent matches the final netlist.

Bottom line: Power intent is a design file like RTL, it is NOT a tool script file since it effects functionality. Its required to be consistent and complete throughout the design flow.

3.3 Always Run Power Aware Simulations

Originally, some customers defined a set of tests for low power, and only ran these specific tests with the low power modeling. In theory, this should be sufficient to test the low power functionality. But in reality it is incomplete and can cause fatal errors to be masked.

In one customer's case, the power control logic had an error in it. This caused an unexpected power shutoff to a domain. Since this was unexpected, the domain was not fully isolated, and an X was pushed out onto the system bus. This caused the system bus to lock up; in hardware, it would have required a reboot to clear the bus.

This issue **was only detected** because the user switched to running all verification scenarios with power aware simulation. If the user had not made this switch, the issue would not have been detected until after tapeout.

The user's original methodology was designed when they had an in-house PLI solution for modeling the power shutoff. As with most PLI based low power solutions it had a pretty severe performance penalty and ran 3-4 times slower than running a normal simulation. When they switched native compiled low power simulator, like the Cadence Incisive Enterprise Simulator, they were able to run all simulations with power. The performance penalty became insignificant, especially when compared to the risk of missing issues.

A good analogy is directed tests vs. constrained random. The directed tests do an excellent job of verifying a specific scenario, but in the end constrained random simulations are used to provide better coverage. It becomes too difficult to design a test for all possible scenarios. In Low power it is the same, running the equivalent of directed test checks a specific scenario, but it limits the coverage and exposes the design to risk when the application does anything unexpected.

3.4 Pessimistic Corruption model

Corruption is the process of modeling the effects of low voltage or power shutoff in the simulator. This is one of the most important functions of a power aware simulator. If the corruption is not modeled correctly, the simulation will result in false positives, which in turn can lead to failures in silicon. It's critical to model this as accurately as possible, and when in doubt be conservative.

The modeling of corruption has evolved based on real world usage by customers, and the following sections describe some of the specific cases we found.

3.4.1 Voltage Based Corruption Model

The philosophy on how to handle voltage ramps for power shutoff designs varies from vendor to vendor and customer to customer. Engineers have a tendency to want to measure and show every value as accurately as possible. But in this case that modeling could be counterproductive.

First, let's define what the voltage ramp corruption .The voltage ramp models the fact that voltage changes don't happen instantaneously. The ramp defines how the voltage changes, and is used by simulation to determine when during a power cycle the data is valid. See the diagram below:

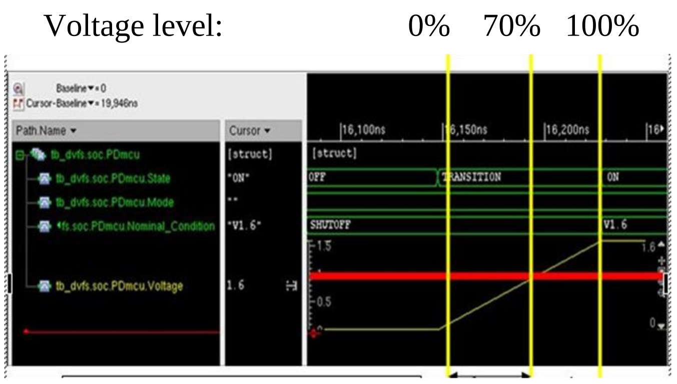

Figure 3 - Voltage based corruption

The differences in view among vendors are related to exactly when the data is valid. In the diagram above the region between 0 and 70% of voltage is universally considered corrupted, as the voltage is not high enough to support a logic functions. The region between 70% and 100% is considered a valid voltage by some tools and actions like restoring a retention cell or even clocking in data are considered valid in that region. At Cadence, we found this to be too optimistic. Instead we model the voltage as off from the instant we get the power down signal all the way until the voltage reaches 100% of its target.

The greatest risk is at the RTL level, where it simply is impossible to accurately predict the voltage ramp. Even with a non-linear model of the voltage, there is not enough information at this time to accurately model this. But it is possible to define a worst case number and use that as a constraint for the backend and for the simulations.

Why is the voltage ramp difficult to model?

- The ramp is a non-linear function with oscillations
- It is not a static function – it varies by voltage/temperature/current and cross-chip variations
- The ramp depends on the powers switch architecture: layout, number of switches, mother-daughter vs. daisy chain, etc
- It depends on number, size and layout of the cells driven by each switch

The information to model this accurate is not available at the RTL level. Trying to squeeze out a few nanoseconds of usable time from a power shutoff is counterproductive; it exposes the design to risk for something that has a small impact on overall system performance.

The recommended methodology is to specify a maximum voltage transition and use a physical verification tool to verify that the transition time is met.

3.4.2 Corruption and isolation of Constants

As customers transitions from in-house power modeling to using IES we found a number of differences in the simulation results. One major difference was that customers seldom corrupted constants. The simulation engine needs to model the hardware as accurately as possible. Constants that are not optimized away by synthesis will have a direct or indirect connection to power or ground. When the domain powers off, the value on the output of the power and ground may not be valid, so the simulation engine needs to corrupt those values. On power up, the signals need to be restored to their original values.

The Cadence approach is to corrupt both hi and low constants. Some vendors would like only the high constants corrupted in a power switched design, and low constants corrupted in a ground switched design.

The Cadence approach is pessimistic, but it needs to be. The synthesis tools may optimize the logic and invert the constants. At the RTL level there is no way to predict, so the only safe approach is to corrupt all constants.

3.4.3 Modeling Input Pin Corruption

Another requirement in corruption modeling is the corruption of input pins. The input pin corruption ensures that combinational logic, through assigns or other logic is corrupted. Without this corruption the simulation results could be optimistic and miss real problems in the design.

While the traditional approach of corruption outputs and internal state handles most of the required power shutoff corruption, it doesn't cover everything. For instance any monitors or assertions that are checking the inputs of a design will not see the corrupted data (DRV B). In other cases, logic combinational feed through paths may not corrupt, leading to optimistic results outside of the block.(DRV A below)

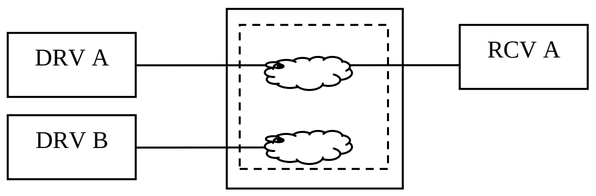

Figure 4 - Macro Input Corruption

The simulator does make a special case for feed through paths, with the premise that the implementation tools are intelligent enough not to buffer a feedthrough path inside a switchable domain. If your implementation tool doesn't have that ability, then a command line option is provided to treat feedthroughs as buffers.

3.4.3 Standby/Sleep Mode corruption.

Standby mode or sleep mode is a case of dynamic voltage scaling where the voltage for a domain is reduced to the point where it is high enough to maintain state, but too low to compute new values. If an input to the domain changes while in standby mode that input is corrupted. In reality, it is difficult to predict if logic will corrupt or not. For instance, in a power switched design, a transition to 0 is probably okay. The safe approach for a simulator is to always corrupt.

The original recommendation by Cadence was even more pessimistic. It would corrupt the entire domain if any of the inputs changed. But based on customer feedback this was relaxed to corrupt only the related inputs (and through normal event propagation their cone of logic), and issue a warning message.

The recommendation is to treat any of these warnings very seriously. It can sometimes be difficult to ensure that a single inputs corruption is detected by the simulation environment. The warning provides a flag that should be checked as part of the verification signoff process.

5. CONCLUSION

Verification of advanced low power designs needs to leverage the best practices of general verification as well as leverage the experience hard won in the field. The recommendations here represent a partial view of the results of the past 4 or 5 years deploying low power verification methodologies at numerous customers worldwide. Understanding the low power architecture and what steps are required to properly model the power intent can avoid many of the common issues in low power, and at the very least, provide for earlier detection of the issues.

Successful low power verification flows should provide methodologies to verify the power intent as early in the flow as possible. This not only means utilizing the structural checks up front, but also includes modeling the power intent such that RTL simulations provide the an accurate representation of what the final implementation will be. This ensures that the RTL verification provides the highest degree of coverage.

This process often involves making decisions on modeling or flow that are pessimistic in nature, but provide pay offs in productive and proven silicon.

This document is primarily focused on the RTL verification of low power, but this is just a single step in the full low power flow. Future papers will explore the verification as it moves up into the TLM/ESL space and also down into the gate level and more physical verification steps.

6. ACKNOWLEDGMENTS

The recommendations outlined in this document have come from gathering input from many sources. This includes customers, Cadence application engineers, product engineers, and R&D. I wanted to extend my thanks to all that have contributed, directly or indirectly to making a more robust low power verification methodology.

7. REFERENCES

[1] Common Power Format Language Reference. Version 1.1. August, 2010.
[2] Neyaz Khan and William Winkeler,. 2008. Power Assertions and Coverage for improving quality of Low Power Verification and closure of Power Intent. In DVCON (San Jose, Calif., Feb 19 - 21, 2008).
[3] Cadence CPF Methodology Guide, version 1.1, October, 2010
http://support.cadence.com/wps/mypoc/cos?uri=deeplinkmin:DocumentViewer;src=wp;q=ProductInformation/Digital_IC_Design/CPF_Methodology_v1.1.pdf

[4] A Practical Guide to Low-Power Design - http://www.powerforward.org