cādence®

Power-Aware Verification Spans IC Design Cycle
A Plan-To-Closure Approach Helps Ensure Silicon Success

By John Decker, Neyaz Khan, and Richard Goering, Cadence Design Systems

The central problem with low-power verification is the complexity caused by using today's advanced low-power design techniques. A "plan-to-closure" approach can ease power-aware IC verification tasks. This approach starts with a verification plan that is tracked through coverage metrics. It uses the Common Power Format (CPF) to capture and maintain power intent throughout the verification flow. Cadence solutions address these and more tough simulation challenges.

Contents

Introduction

The mandate to reduce system power consumption and design energy-efficient ICs has 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 such as power shutoff, back body biasing, and dynamic voltage and frequency scaling (DVFS). More and more chips have multiple operating modes as well as multiple power domains with different, and perhaps dynamically variable, voltage levels.

The central problem with low-power verification is the explosion in scope and complexity caused by low-power design techniques. Some chips today have 20 to 50 power domains and hundreds of power modes. As a result, chips may have thousands or tens of thousands of possible power states. Verification engineers must ensure that the chip functions correctly in each state that could plausibly occur, and that all transitions between states are properly handled.

Because of this complexity, verification planning is essential for low-power designs. An ad-hoc approach is not likely to succeed. The 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. Along the way, low-power design effects must be continuously analyzed and verified from the systems level to GDSII.

With low-power design, verification engineers face new challenges such as the following:

- Ensuring that the system-level architecture is using the power modes efficiently and correctly.
- Verifying the interactions between different power domains.
- Verifying the transitions between different power modes and states.
- Modeling low-power structures that do not exist in RTL.

- Ensuring that isolation, state retention, and power shutoff are handled properly, and that the device can power on correctly within a given time period.

Different power optimization techniques have differing impacts on verification. For example, clock gating has a minor impact, multiple power domains and multi-Vt libraries have a moderate impact, and power shutoff, DVFS, and multiple operating modes have a major impact (see Figure 1).

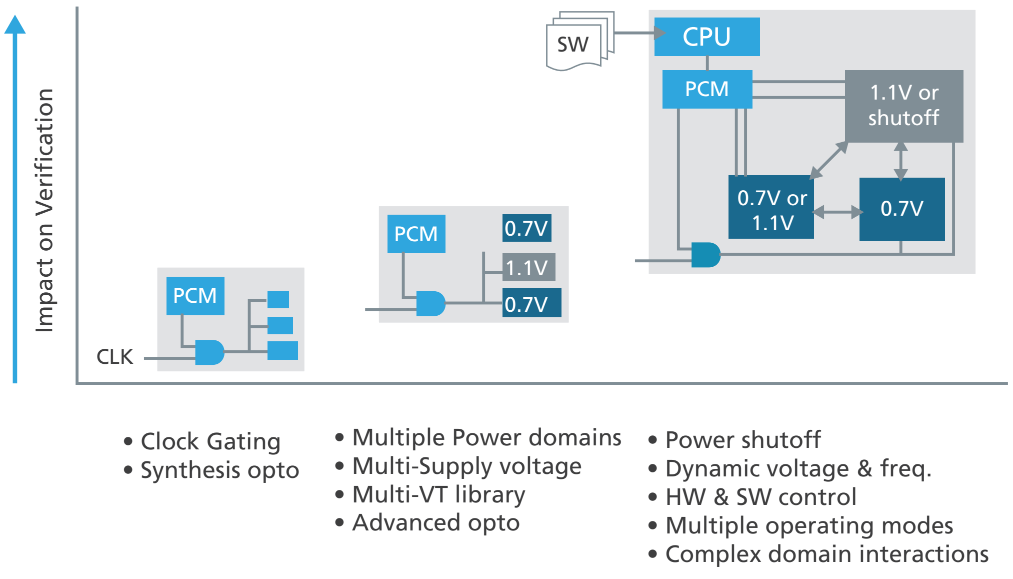


Figure 1: Some advanced low-power design techniques have a profound impact on verification.

This paper shows how a "plan to closure" approach can ease power-aware IC verification tasks. Such an approach starts with a verification plan that is tracked through coverage metrics. It makes use of the Common Power Format (CPF) to capture power intent throughout the verification flow. The paper discusses the use of behavioral simulation, RTL simulation, assertions, emulation, and formal verification for low-power verification. It looks at tough simulation challenges posed by techniques such as power shutoff and DVFS, and proposes some solutions.

Requirements for Low-Power Verification

A successful low-power verification methodology requires these prerequisites:

- **An executable, metric-driven verification plan.** Verification planning brings all stakeholders together to capture the verification intent, and to identify criteria for verification closure. Because low-power design introduces a great deal of complexity into the verification environment, a measurable low-power verification strategy must be developed up front.

- **A common view of the power architecture.** There are a large number of tools in the low-power verification flow, and all must share a coherent view of the power architecture. In the Cadence Design Systems flow, the Common Power Format (CPF) provides that view.

- **Tools with a native understanding of power intent.** Simulators and other tools in the low-power verification flow need to understand low-power states and requirements. If a test unexpectedly forces a power event, for instance, the tool must be able to recognize that. This is only practical if a tool provides high performance for low-power verification.

- **Reuse of existing IP and methodologies.** A low-power verification approach must support a high degree of reuse, even if existing design and verification IP was not developed specifically for low power. The "side file" approach used by CPF, which does not require modifications to RTL code, is one step in this direction.

- **Automated, early formal checking.** Static checking can expose problems and verify changes in the power architecture with minimal impact to the rest of the verification environment.

Power-Aware Verification Planning

Defining an architecture

The low-power verification flow begins when a power architecture is defined for the design. This architecture partitions the design into major blocks and determines which power-saving techniques can be used by each block. As decisions are made about the power architecture, they need to be captured in a format that can be used by all tools. The Cadence flow uses CPF for this purpose.

An architect will create a system-level CPF file to define the top-level power domains and system-level power modes in the design. This file will typically include information that identifies the following:

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

The following code example shows how CPF describes power domains:

```
# Define the top domain
set_design TOP

# Define the default always on domain
create_power_domain -name pdTop -default#

Define Power Shutoff domains
create_power_domain -name pdA
   -instances {uA uC}
   -shutoff_condition {!uPCM/pso[0]}

create_power_domain -name pdB
   -instances {uA uC}
   -shutoff_condition {!uPCM/pso[1]}

#Define Power Modes
Create_power_mode -name M_ON
   -nominal_conditions {pdA@v1.0  pdB@v1.0}

Create_power_mode -name M_OFF
   -nominal_conditions {pdA@off    pdB@off}

Create_power_mode -name M_MID
   -nominal_conditions {pdA@v0.8 pdB@off
```

Figure 2 – CPF code used to define a power domain

Typically, the system-level CPF file will reference several lower-level CPF files. These may include a technology file that defines the libraries and technology mapping rules, and a domain-level file that defines the basic control signal inputs and interface requirements for that domain. Since CPF is contained in a side file, no RTL changes are required to capture power intent. While the power intent is specified in the CPF file, the RTL or software running on the device under verification still needs to be updated to include the logic that controls the power management of the system.

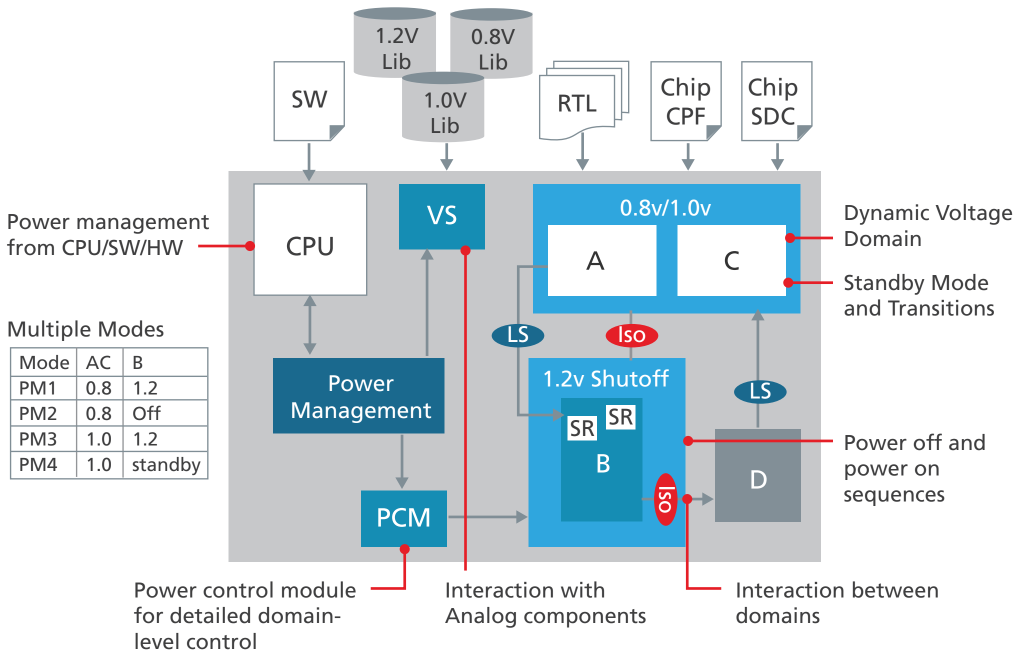
An executable plan

As the power architecture is defined, a corresponding verification plan must be developed. This plan allows all stakeholders to agree on what needs to be verified and how it will be verified. The plan defines verification closure and describes how to measure it. With the Cadence® Incisive® Enterprise Manager and the Cadence® Incisive® Plan-to-Closure Methodology (IPCM), the planning process results in a machine-executable verification plan that can track the progress of the verification effort using metrics such as functional coverage.

Verification planning is always important, but it is especially important for low-power design because the power architecture adds so much complexity to the verification. The verification space increases with each power mode, and the device must be validated in all modes.

Ideally each verification mode is fully validated, but often the verification space is too large for exhaustive verification. The verification planning process provides a documented analysis of the important modes and required verification. Verification engineers must understand which features are valid in each mode. Power modes must be entered and exited without errors, and transitions between power domains must be handled properly.

To fulfill these requirements, the verification plan should include a section on the verification of power intent. This section should describe the power modes and the requirements for exercising them. It should define the set of features that need to be verified in each of the design's power modes. It should also define the features that are not available in given power modes. Figure 3 shows some of the power-related criteria that should be covered in a verification plan.


1.2V Lib
0.8V Lib
1.0V Lib
SW
RTL
Chip CPF
Chip SDC
CPU
VS
0.8v/1.0v
A
C
LS
Iso
1.2v Shutoff
SR
SR
B
Iso
LS
D
Power Management
PCM
Power management from CPU/SW/HW
Multiple Modes
Dynamic Voltage Domain
Standby Mode and Transitions
Power off and power on sequences
Power control module for detailed domain-level control
Interaction with Analog components
Interaction between domains

Mode	AC	B
PM1	0.8	1.2
PM2	0.8	Off
PM3	1.0	1.2
PM4	1.0	standby

Figure 3 – A verification plan needs to include information about a variety of low-power design features.

In summary, the key items defined in the verification plan are:

- Description of power behavior
- Coverage items to ensure power behavior was tested
- Assertions and/or test scenarios to ensure the design operates properly

Coverage metrics

Functional coverage can be used to gauge and measure the quality and completeness of power simulations. This is done by creating a coverage model based on the power control elements of the design, and then managing the verification effort to optimize the collection of coverage data.

Power closure occurs when pre-defined verification goals are met, using specified metrics such as functional coverage. Two steps are involved in achieving power intent closure:

- Designing a coverage model that quantifies the functionality that needs to be tested. Incisive Enterprise Manager provides automatic coverage generation that ensures that all defined power modes are exercised, and all defined power control signals are toggled. It creates the cover groups needed to collect and capture metrics for low-power simulations are automatically created. To accomplish this, the CPF file is parsed for intended power intent and the corresponding *e* or SystemVerilog code is generated automatically. The automated coverage is one piece of the overall low power coverage model. The planning process will also define additional coverage metrics to fully cover the low power architecture and verification plan.
- Specifying coverage goals in the executable verification plan, and capturing results during simulation. This makes it possible to use coverage metrics from targeted cover groups to measure power coverage and assertions.

Automatic coverage generation will also be provided for SystemVerilog by the Cadence® Incisive® Unified Simulator.

System-level power verification planning

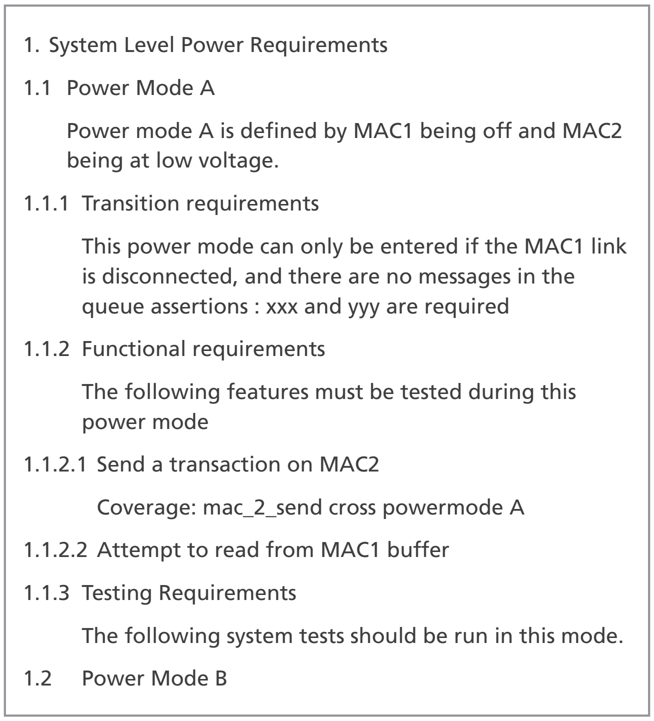
System-level verification spans multiple domains. At this level, the two main planning goals are to verify that the design works as expected in all power modes, and to provide the verification strategy for the power management logic and software. Planning should include the following:

- Defining which system tasks should occur –and should not occur -- in each power mode of the design, and developing appropriate tests
- Defining the interactions and isolation requirements between power domains
- Ensuring proper behavior during a power mode transition
- Verifying interactions between power modes and power domains
- Verifying that power mode transitions operate correctly
- Validating that the system has been fully tested in each power mode

At the system level, it is important to verify power management decisions. During the planning stage, engineers should define what modes of operation and transitions between nodes are valid, and describe the scenarios that will cause the transitions. They should verify all methods that will cause a node change – such as writing a register, responding to external stimuli, or side effects of other transitions.

Next, it is important to define coverage items to ensure that the expected scenarios appear. This typically involves a mix of power modes and other variables – for instance, "in power mode X, did I get transaction Y?" Finally, assertions can be used to verify the sequences of power management tasks, and to ensure that power events only happen at legitimate times.

To ensure that power modes are verified, the plan should define the set of scenarios, tests, transactions, and features that must be covered in each unique power mode. It should identify sequences of power mode transitions, specify the required time periods for each mode, and note whether specific transactions, such as memory writes, can cross power modes. The planning phase should also identify expected or undesired domain interactions. For example, one domain might need to transition before another domain, or system-level transactions might still need to work after a domain is turned off.



1. System Level Power Requirements

1.1 Power Mode A

Power mode A is defined by MAC1 being off and MAC2 being at low voltage.

1.1.1 Transition requirements

This power mode can only be entered if the MAC1 link is disconnected, and there are no messages in the queue assertions : xxx and yyy are required

1.1.2 Functional requirements

The following features must be tested during this power mode

1.1.2.1 Send a transaction on MAC2

Coverage: mac_2_send cross powermode A

1.1.2.2 Attempt to read from MAC1 buffer

1.1.3 Testing Requirements

The following system tests should be run in this mode.

1.2 Power Mode B

Figure 4: A system-level verification plan

Domain-level power verification planning

A power domain is a collection of cells that each have the same low-power behavior and control. Domain-level verification ensures that each power domain executes all its possible states and transactions, and that control logic operates correctly. Verifying at the power domain level is similar to verifying a block level design; it verifies the basic function of each domain independently of its system-level context. Power domain-level verification planning includes the following:

- Translating power control module requirements and sequence requirements into coverage and assertions. Coverage metrics include each domain state and transaction. Assertions ensure proper state transitions and check individual control signals.
- Setting internal state requirements for power transitions.
- Defining state retention/isolation requirements.
- Specifying isolation requirements.

Low-Power Verification Throughout The Flow

Following the development of a verification plan, design and verification teams turn to a variety of tools and methodologies to reach verification closure. These may include formal property checking, equivalence checking, simulation at various levels of abstraction, emulation, assertions, and coverage metrics. An integrated flow that has a common understanding of power intent is important. This section shows how such tools and techniques can be used to verify power-aware IC designs.

Static (formal) checking

CPF quality checking is typically one of the first steps that’s taken after a verification plan is developed. This is because the architectural specification itself must be analyzed and verified for functional, electrical, and structural completeness and correctness.

Cadence® Encounter® Conformal® Low Power provides automated checks that can ensure that the specified power intent is complete and correct. It also provides a “linting” capability that checks for proper syntax.

Encounter Conformal Low Power is used throughout the verification flow. Early checks might include tests for missing isolation or level shifter cells, tests for power control functionality, and checks that state retention and isolation control signals are driven correctly by domains that remain powered up. Later on, post-placement checks can ensure that gate power pins are hooked to the appropriate power rails, that always-on cells are appropriately powered, and that there are no "sneak" paths from power-down domains back to logic.

Encounter Conformal Low Power also provides logical equivalence checking (LEC), and that too can be used throughout the RTL-to-GDSII verification flow. Equivalence checking can prove that isolation and state retention cells have been inserted correctly, even though these cells are in the CPF file rather than the RTL source. Figure 5 shows how static checking and LEC are used during the low-power verification flow.

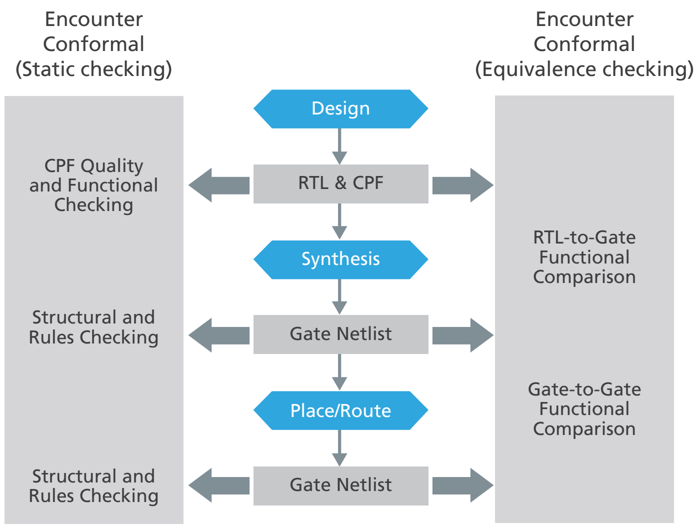

Encounter Conformal (Static checking)
Encounter Conformal (Equivalence checking)
Design
CPF Quality and Functional Checking
RTL & CPF
RTL-to-Gate Functional Comparison
Synthesis
Structural and Rules Checking
Gate Netlist
Place/Route
Gate-to-Gate Functional Comparison
Structural and Rules Checking
Gate Netlist

Figure 5 – Encounter Conformal Low Power can be used throughout the verification flow.

Low-power simulation

After CPF is used to capture power intent, design teams can simulate behavior such as power shutoff, isolation, and retention, and observe the effects. Because this information is specified in the CPF file, not the RTL, no RTL changes are required. Given the information in the CPF file and the appropriate test vectors, a CPF-aware simulator such as the Incisive Unified Simulator can exercise the power shutdown and startup sequences for the design.

CPF commands that are understood by Incisive Unified Simulator 6.2 and above include:

- Power domain definition – different power domains and operation supply voltage
- Power shutoff property specification – isolation, retention, and power-down triggers
- Level shifters between power domains that run at different voltages

Low power simulation should be enabled for all simulation runs. If power-aware simulation is run only in tests and scenarios that are designed to test power-aware features, simulations may not catch errors that unexpectedly trigger low-power features. With Incisive Unified Simulator there is virtually no run-time impact in running with low power enabled, and no reason not to do so.

Figure 6 shows the types of simulation that should be run, what device under verification (DUV) representation they use, and what else the verification environment might include, such as coverage, the Cadence® Incisive® Palladium® Accelerator/Emulator, Cadence® Incisive® Software Extensions, and Cadence® Incisive® Formal Verifier.

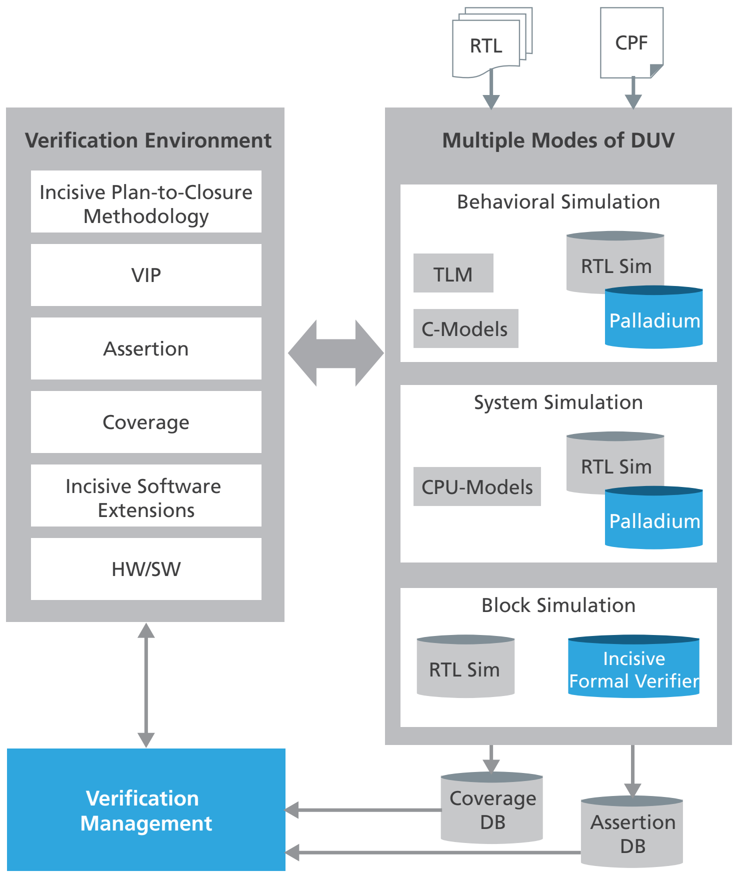


Figure 6 – Types of simulation used in low-power verification

The role of behavioral simulation is to verify the system and software architecture of the design. In the low power environment, there are two key tasks:

- Functionally verify the power architecture from a systems perspective
- Ensure that the power architecture is used efficiently by the system software and hardware

For the first task, the most common sources of functional error are the interfaces between power domains and the transitions between power modes, especially when power shutoff is used. Ensuring that all power modes and transitions have been exercised is the only way to verify that these interfaces are defined correctly. For the second task, the verification environment needs to ensure that all the available power modes are used, and that the low power modes are used as often as expected.

Verifying these issues requires extensive simulations with high-level models. This includes transac-tion level models (TLMs), C language models, and possibly emulation technology. A fairly simple model of the power architecture is probably sufficient. RTL simulation will require a much more detailed model.

System-level simulation provides a detailed functional verification of the DUV. Unlike behavioral simulation, it requires accurate models of every component. The simulation can include the entire SoC, or it could focus on major sub-units, such as those containing power domains or power management logic. Simulation engines model the power structures defined in the CPF file. These engines typically include RTL simulation and/or hardware emulation systems.

Block-level simulation looks at the individual blocks that make up a power domain and its corresponding control logic. In addition to RTL simulation, verification at this level may involve formal property checking. One goal of this level of simulation is to verify that an individual domain's low power features are verified in isolation.

Since equivalence checking can ensure that a gate-level netlist matches the original specification, gate-level simulation is typically used for only for small, targeted tests. Still, many design teams run some gate-level simulation for low-power designs. By simulating the logical netlist that appears after synthesis, engineers can verify low-power structures such as isolation, state retention and level shifting.

The physical netlist resulting after placement adds power and ground nets, pins, and switches. Physical netlist simulation typically verifies physical connections or checks for dynamic hazards in the design. Tools such as Encounter Conformal Low Power and Cadence® VoltageStorm® Power Verification can generally handle these concerns better.

Embedded software verification

Embedded software plays a crucial role in low-power design, and should be included in the verification process. In most ICs, power modes are primarily defined and controlled by software. Moreover, software development is based on architectural choices that impact power, such as the size of caches, the use of internal or external memory, and the amount of I/O traffic.

Incisive Software Extensions gives the verification testbench access to software executing on processor models. It lets the verification team use the Incisive debugger to control and verify system-level behavior including software processes, function calls, and variables, along with processor and register behavior. Incisive Software Extensions brings constrained-random stimulus generation and functional coverage metrics into the software development world. It also permits execution of a single verification plan including both hardware and software.

Emulation

The enlarged verification space caused by low-power design techniques makes hardware emulation an attractive option for power-aware verification. Emulation greatly speeds verification, allowing engineers to process far more clock cycles than would be possible using software simulation alone. Further, emulation provides a virtual model of the design that can give software engineers a head start in developing firmware and applications software, which, as noted above, affects the power consumption of the SoC. The speed of emulation lets engineers run real-world software applications as part of the verification process. This is critical for low power design, since the time span for low power events is typically very large and often requires simulating real world applications.

Cadence® Incisive® Palladium Dynamic Power Analysis lets users capture and analyze power switching activities for peak and average SoC power consumption, including the effects of embedded software and other real-world stimuli. This can help determine, for example, whether and how often a given block is powered down. The emulator can also pinpoint "hot spots" that show areas of high power consumption, and return the associated activity to the software simulator for a more accurate analysis.

Formal property checking

A formal property checking tool such as Incisive Formal Verifier can exhaustively prove that assertions are correct, without requiring the user to generate testbenches. For low power design, formal property checking can:

- Verify user-generated assertions for power management
- Verify the assertions generated on the power control module associated with each power domain
- Verify that design assertions with low-power intent are correctly modeled

In the case of power shutoff, Incisive Formal Verifier can be run in two different modes. First, it is run to ensure that all the assertions in the RTL are valid in both the powered on and powered off states in the design. Secondly, it is used to formally verify the generated assertions on the power control module.

Assertion-based verification

Many design and verification teams use assertions to drive formal tools, or to monitor specific test cases and ensure specific properties during simulation. Assertions also provide coverage data that supplements the data obtained from cover groups.

Assertions are a powerful tool for low-power verification. They can be used to validate power control logic, system-level power management, and specific requirements met before and after power mode transitions. But assertions must be used properly, especially when power shutdown techniques are used. Further, engineers must make sure that "traditional" (non power-related) assertions, such as those for bus protocols, hold in all power modes and during all power mode transitions.

A number of assertions can be automatically generated from CPF specifications. Such assertions monitor and check transitions in the power control signals to verify legal and correct low-power behavior. These assertions remain active, yet dormant, for powered-down modules. They can also provide coverage metrics.

Power-aware assertions go beyond the items that can be inferred by the CPF file, and may include the following:

- The finite state machine (FSM) for a design must be idle before this design/block can be powered down.
- During power mode A, configuration register X should be a specific value.
- Bus Y should be idle for N cycles before power-down.

The Property Specification Language (PSL) assertion segment in Figure 7 shows a power cycle with errors. The assertions flag incorrect power shutoff (PSO) behavior during both power-down and power-up sequences.

Figure 7 – PSL-based assertions for low-power control checks

As noted in the next section, it is important to understand the potential impact of PSO on pre-existing assertions, because assertions may fire inappropriately during power shutoff.

Mixed-signal simulation

Most low-power ASICs and SoCs contain some analog or mixed-signal circuitry, and this needs to be included in simulation. When a digital component powers off or changes voltage, it may impact analog components. Digital and analog simulation need to be aware of this change and be able to track its effects.

```
//Isolation must occur before Power-Off
PwrAsrt_ISO_before_PSO: assert always ({!iso; iso}|-> (!pso before !iso));

//Isolation must hold steady during Power-Off
PwrAsrt_hold_ISO_in_PSO: assert always ({!iso; iso[*]; !pso} |-> (iso until pso));

//Full power cycle
PwrAsrt_Full_PwrCycle: assert always ({!iso; iso[*2:5]; !pso[*]; pso} |-> (iso[*]; !iso});
```

Mixed-signal simulations can take advantage of features in the Incisive Unified Simulator that allow low-power effects to be propagated to analog blocks that interface directly or indirectly with powered-down digital logic. The simulator identifies the CPF influence on the analog blocks, and inserts Power-Smart Connect Modules (PSCM) that help propagate the effect of power shutoff onto the analog blocks. PSCMs can also back-trace the digital drivers to get power domain and state information, and can perform digital-to-analog value conversion based on the information obtained.

Analog components used for power management in an SoC can be efficiently modeled using a technique called Real Valued Modeling (RVM). This technique bridges the gap between the performance requirements for a full-chip simulation and the accuracy limitations of a mixed signal design. A significant speedup in simulation performance can be achieved by replacing the analog portions of the SoC with functionally equivalent real number models that do not require the analog simulation engine. While the RVM models trade off accuracy for speed, they are not affected by typical analog simulation problems like convergence.

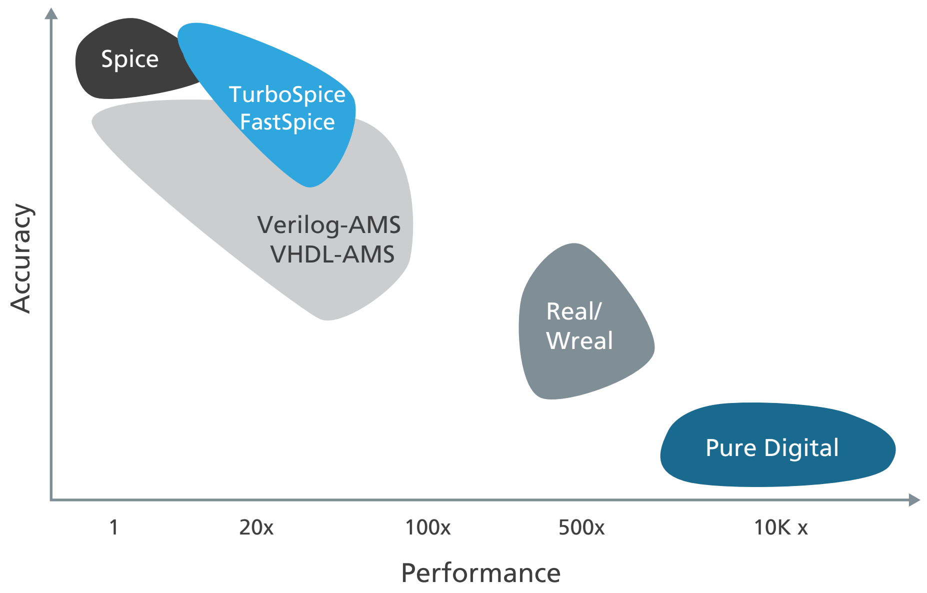


Figure 8 – Speed versus accuracy tradeoff for mixed-signal simulation.

In applications where power-shutoff (PSO) is not applicable, power management is often achieved through DVFS, which dynamically scales the operating frequency and voltage of the target design. The verification of DVFS is often a very difficult and delicate task that involves tremendous interaction between the digital and the analog domains.

Beyond CPF, DVFS verification requires Wreal models for voltage sources and regulators like low-dropout regulators (LDOs), as well as clock generators such as voltage-controlled oscillators (VCOs) and phase-lock loops (PLLs). The various nominal voltages for each power domain need to be generated accurately and to be constantly monitored throughout the simulation. Additional checks need to be performed to keep track of complex interactions between the digital and analog domains throughout the voltage scaling process. Any errors in voltage levels and sequences of transitions need to be detected, and the corresponding voltage domains corrupted.

Tough Challenge #1 – Verifying Power Shutoff

Understanding how to deal with power shutoff (PSO) is one of the most critical aspects of power-aware verification. PSO has a significant impact on simulation, assertions, coverage, verification IP, and modeling. The first point to understand is what not to do. If PSO is defined in the CPF file, the user-defined verification environment should not attempt to model power shutoff corruption, state retention, or isolation inside the DUV. Doing so actually increases the chances of failure. The only exception to this statement is that some behavioral models, like RAM models, may require a signal that defines when power is shut off to the module containing the RAM.

Simulation

Constructs such as power shutoff corruption, power switches, save/restore logic, and isolation should be described in the CPF file, not the RTL. The simulator needs to model the behaviors described in the CPF file during power-down and power-up sequences.

During power-down, the simulator needs to consider isolation and state retention. During power-off, the simulator models the corruption of data internal to the power domain by setting all the internal values to Xs. During power-up, the simulator models the restoration of power to the domain. Incisive Unified Simulator 8.2 lets users specify the time it should take for power-up to occur.

Assertions

In many cases, assertions will not be valid during a power-down phase. Any unclocked assertion that requires specific signal values will probably fail during power shutoff, because the signals will have gone to X states. To avoid this problem, assertions can be made power-aware, but this makes them dependent on the power architecture and harder to reuse.

One way to assure that assertions will work during power shutoff is to use the CPF command "create_assertion_control." This command lets users disable an assertion, or set of assertions, during the power shutdown sequence. The user has detailed control over when the assertion is valid, and can specify whether an assertion should be reset or suspended during power shutoff.

Gated clocks can eliminate the unwanted triggering of clocked assertions. If the clock is gated during power shutoff, any clocked assertion will be suspended.

Incisive Unified Simulator can automatically generate assertions to check PSO and isolation. Examples of assertions for shutoff and isolation are shown in the diagram below.

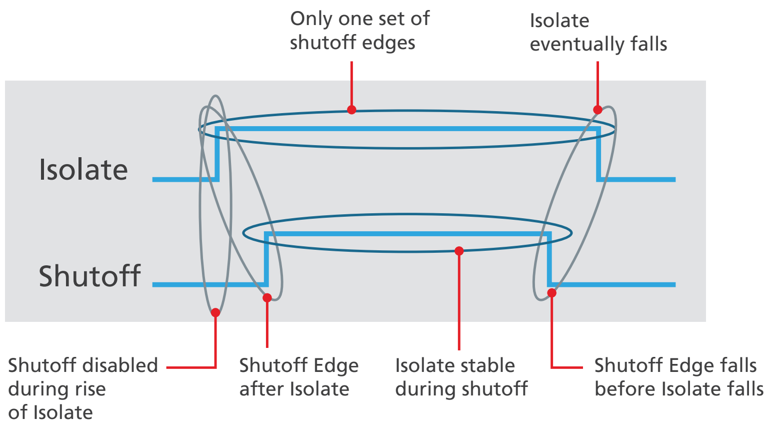

Only one set of shutoff edges
Isolate eventually falls
Isolate
Shutoff
Shutoff disabled during rise of Isolate
Shutoff Edge after Isolate
Isolate stable during shutoff
Shutoff Edge falls before Isolate falls

Figure 9 – Sample assertions for shutoff and isolation

Coverage

Coverage of power control and management is very important to the verification of low power design. The coverage items that are internal to a power shutoff domain are most often not very interesting during power shutoff. In many cases these are automatically ignored during power shutoff -- signals transition to "X," and clock gating during shutoff prevents most coverage events from triggering. In other cases, such as those involving immediate assertions, the user may need to disable assertions manually using the CPF create_assertion_control command.

PSO may require changes to the coverage model in two ways:

- Coverage on a power domain input from outside the power domain. Because inputs will be corrupted inside the domain, verification IP that monitors the inputs of a power domain will see the signals toggling. Designers need to decide whether this coverage is desired.
- Internal coverage points during a multi-cycle state retention. State retention may take multiple clock cycles, creating coverage events as data is scanned through the system. Designers need to decide whether these events should be covered.

Scoreboards

A scoreboard monitors the inputs and/or outputs of a module, and checks the traffic that passes through. When power is shut off, the verification team needs to decide the appropriate action for the scoreboard. In many cases no action is required -- checkers on the interface of a domain often will continue to work throughout the power shutoff cycle. In other cases, sequences may need to be reset, or shadow registers in the verification code will need to be synchronized with the DUV.

Verification IP

Some verification IP (VIP) blocks have internal state elements. It is possible for an internal state to get out of sync with the DUV. For example, a VIP may have an internal state tracking the progress of a transaction in a shut-off module. When powering up, the module resets itself. If the VIP does not monitor that reset signal, its internal state will be different from the DUV. To minimize this possibility, designers should ensure that all resets and clocks in the verification code are identical to those used in the DUV.

Tough Challenge #2 – Verifying DVFS

Static multiple supply voltage (MSV) designs have very little impact on the simulation environment. They can be completely verified with static checks provided by Encounter Conformal Low Power. The checking needs to ensure that the voltage difference between two domains is within a given threshold, and if not, that level shifters are provided. This threshold is technology dependent but is generally less than 100 millivolts.

If the static checks are clean, the different voltage levels will not change the logic values seen in simulation, and these voltage levels place no requirements on the simulation engine.

Dynamic voltage, however, does impact the simulation. First, simulation needs to verify power domains at all legal combinations of voltages. Secondly, the interfaces between power domains need to be checked to verify that a correct voltage is provided across the interfaces. If any problems arise, either isolation or level shifting should be used to protect the interfaces.

Determining when and where to place level shifters can be a difficult process. Using the power modes defined in the CPF file, Encounter Conformal Low Power can determine if the respective power modes require level shifting or not. During simulation, transition times of the power domains will help determine whether level shifters are necessary. During RTL simulation, the transition time is merely an estimate; physical design is needed to get a more accurate answer. Checking the transitions in RTL simulation can be viewed as an early check, but in the end this must be verified at the physical design level with a tool such as VoltageStorm Power Verification.

Standby mode

Standby mode is a special case of DVFS. This mode reduces voltages to a level that's still high enough to retain the states of all state elements in a domain, but too low to calculate any new values. Full power shutoff, in contrast, turns off all dynamic power and all leakage power except for state retention cells. Standby mode provides a good savings in leakage power without the need for state retention. This simplifies the complex problem of determining the exact state retention registers that are required to restore state. Since the full state is retained, the power-up sequence is typically much faster than a PSO cycle, especially if the PSO does a full reset on power up.

From a verification point of view, when an input changes in standby mode, everything downstream from that input is corrupted. Incisive Unified Simulator 8.2 detects standby voltages, automatically checks for unintended input toggles, and automatically propagates X states as required.

IP Design and Reuse for Low Power

The best way to design an IP block that will be used in a low-power environment is to provide a CPF file that defines power behavior for the block. The block should also come with a detailed specification of the requirements needed to use these low-power features, and include assertions and tests to verify the implementation.

Blocks are sometimes used in power environments they were not designed for. In such cases, it is up to the IP integrator to ensure that the block operates properly. Most low-power design techniques won't perturb the block, but with PSO, the user needs to ensure that the state retention and restoration process will work properly, and define the behavior of assertions.

Blocks that will be reused in a low-power environment should come with the following information:

- State retention and restoration policy
- I/O requirements, including input isolation requirements
- Assertion behavior when power is shut off
- Coverage information

Another ideal deliverable is a compliance checklist that includes a verification plan, a set of coverage items and assertions, and a set of test cases that validate whether the part is being used in an expected manner.

Conclusion

Functional verification is a difficult task – and low-power design makes it even more difficult. With today's low power design techniques, ICs may have dozens or hundreds of different power modes and domains. The IC must function correctly in every possible power state, and transitions between power states must be correct. Power shutoff (PSO) and dynamic voltage and frequency scaling (DVFS) are particularly problematic from a verification standpoint.

A plan-to-closure methodology, complemented by power-aware verification tools and a common way of expressing power intent, can go a long ways towards easing the low-power verification challenge. Such a methodology begins with a verification plan based on the low-power architecture. The plan defines verification closure and describes how to measure it through coverage metrics.

The verification plan should describe all of the power modes and domains, and explain the features available in each power mode. Addressing both system-level and domain-level verification, it provides scenarios, tests, transactions and features for each power mode. A Common Power Format (CPF) file, which is separate from the RTL code, then conveys power intent to all of the downstream tools in the verification flow.

Every tool in the verification toolbox can contribute to low-power verification. Static, formal checking can make sure the power intent is complete and correct. Simulators can read the CPF file and verify the power intent. Emulation, formal property checking, and assertion-based verification can all play an important role. But all these tools must be used with an awareness of low-power features such as PSO and DVFS.

Anyone contemplating low-power design must also consider the challenge of low-power verification. Fortunately, help is available in the form of verification planning, a common power format, and power-aware verification tools. A plan-to-closure methodology and an integrated suite of tools can make low-power verification a manageable challenge..