cadence®

A Better Tool for Functional Verification of Low-Power Designs with IEEE 1801 UPF

By Mehran Ramezani, Engineering Consultant, and Chul Choi, Cadence

To examine simulation and emulation technologies for a thorough, yet faster functional verification of low-power systems on chip (SoCs), this paper first reviews the fundamental sources and reduction techniques of power drain. Then, by exemplifying an MP3 design, it demonstrates the need for co-verification/co-debug of hardware and "close-to-actual" software for better coverage and lower power-consumption verification, while meeting the high-performance requirements. Having made a case for employing close-to-actual software for verification and the need for high visibility into hardware for debugging, the search narrows to Cadence® Palladium® XP Series emulation tools.

Contents

Introduction

Energy conservation has become the fundamental area of concern in system designs, especially in battery-powered devices. Consumers no longer talk about tradeoffs between performance and power conservation. They demand the lowest power consumption combined with the best performance in smallest packages. Additionally, there seems to be no end to their appetite for the latest hardware and software technology. These demands mean that the upcoming low-power designs must meet the complex functional and power requirements, in shorter development cycles.

Minimization of power consumption is one of the most important requirements of any ASIC. ASIC architects and designers use a variety of methods and measures to curtail the power waste and increase efficiency of design. To better appreciate these measures and methods, one needs to consider that the energy in ASICs is either wasted or is consumed to perform functions. The wasted energy is most often attributed to leakage current in the fundamental components of ASICs and is referred to as "static power", while the energy required for performing a function is called "dynamic power".

Static-Power Dissipation and Leakage Current

Due to the inherent characteristics of semiconductor elements in ASICs, any powered circuit in ASIC will lose energy through leakage current. Even if a circuit is idle and not performing a task, it is consuming energy. Leakage currents are the major components of static power dissipation during idle mode, and are increasing dramatically in sub-100nm processes. Sub-threshold leakage rises due to threshold voltage scaling while gate leakage current increases due to scaling of oxide thickness. To reduce the overall leakage

current, the designer must optimize the supply voltage for the lowest level required for the proper functionality of the circuit for the required speed. To eliminate the idle time leakage current, the power to the circuit must be shut down.

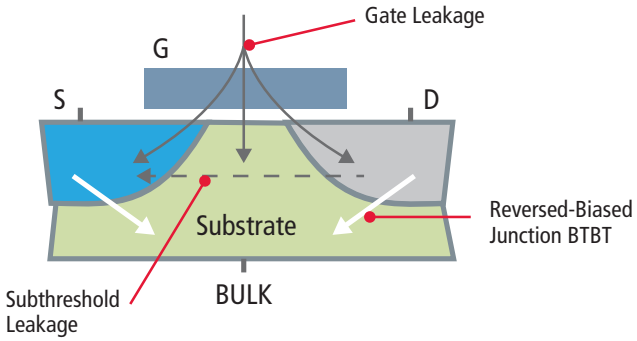

Figure 1. Transistor leakage current

A semiconductor digital circuit consumes energy while it is transitioning from one state to another. As apparent from the formula ($P=C_L.V_{dd}^2.f$), the amount of this energy is mostly influenced by the supply voltage and somewhat related to the characteristics of the semiconductor building block of the ASIC. Considering time in the equation, the faster the circuit is required to switch states, the higher supply voltage and therefore the more power (dynamic power) consumed. Therefore, dynamic power can be optimized by adjusting the frequency at which the digital state is switched and supplying the minimum voltage (V_{dd}) that is required by this frequency. The dynamic power can be eliminated by stopping the logic from switching states.

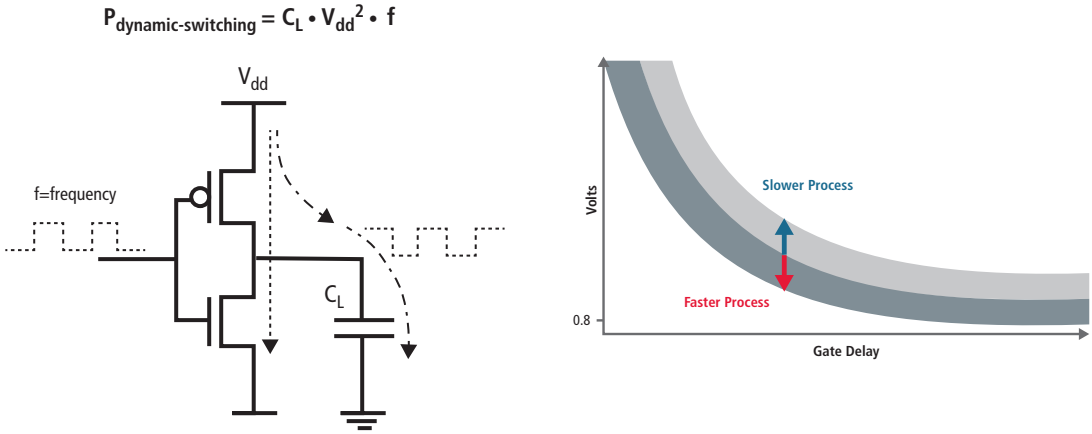

Figure 2. Dynamic-power dissipation

Measures for Minimizing Power Consumption

Methods such as multi-voltage, power gating, clock gating, dynamic voltage, and frequency scaling are widely used to reduce power consumption in SoCs.

Power domains, multi-voltage, and power gating

Not all of the SoC circuits require the same level of supply voltage at all times. Although it might be possible to apply the same level of supply voltage to different circuits, to save power the SoC is partitioned in different voltage regions (power domains). The supply of power to each power domain can, independently from the other regions, be controlled. Supplying different levels of voltage to different power domains is called the multi-voltage strategy, and it is used to reduce dynamic and static power consumption by limiting the exposure of higher levels of supply voltages to the circuits that require it. To totally eliminate static power consumption, the idle power domains can be shut down, which is referred to as power gating.

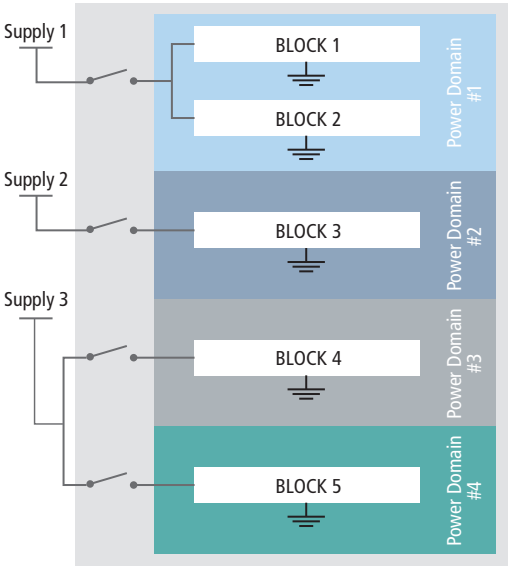

Figure 3. Block power gating

Clock gating and auto gating

Clock signals in digital designs are the major contributor to dynamic power consumption while in idle mode. Clock gating refers to the act of turning off the clocks to idle circuits. In some designs, a circuit might detect a period of inactivity and decide to request for its clock to be turned off, except for a small logic that will be on the watch for upcoming activity. Upon sensing a pending activity, this logic triggers the circuit to request its clock to be turned back on.

Dynamic voltage and frequency scaling

Regularly, the designers enable their circuits to dynamically operate at a higher frequency to meet an occasional demand for higher performance. For example, a processor that normally operates at relatively low clock frequency will temporarily operate at much higher frequencies to handle the extra workload. Unfortunately, an increase of clock frequency might have to be accompanied with an increase in supply voltage. This dynamic scaling of clock frequency and supply voltage allows for a balance between performance and power consumption.

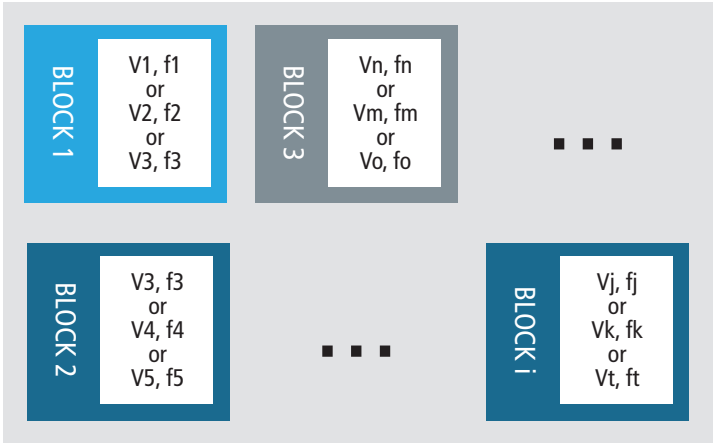

Figure 4. Dynamic voltage and frequency scaling

Most often, the SoC architect defines different levels of performance (e.g., economy, normal, and turbo) for different blocks within SoC. At any given point in time during the SoC operation, different blocks can operate at different performance levels.

Power Management Design and Verification Considerations

Isloation cells, level shifters, retention registers, and power switches are logics used to control the power domain of an IP block. Verifying that these logics are instrumented correctly by the synthesis tool and they are correctly controlled by a power management block is paramount in power management design.

Isolation cells

Isolation cells are placed between two blocks that can independently be powered off. When an IP block is powered off, the isolation cell clamps its output to a predetermined value. This feature is usually used to protect the active block from receiving spurious and unintended interrupts and signals from the powered-off block. The verification of the predetermined clamp values are a very important part of power-aware tests. These predetermined values are usually where the power intent and functional intent overlap and are often a source of trouble.

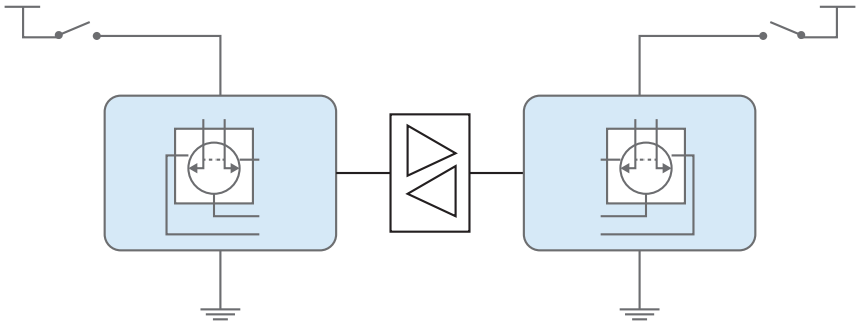

Figure 5. Isolation cells

Level shifters

Level shifters convert the voltage range of an output signal to a different voltage range suitable for the input logic of the receiving block. Verifying the correct operation of the interaction between two power domains when level shifters are involved is difficult and is often neglected.

Retention registers

Retention registers are used to preserve the state of the circuit from before power-down to after the power domain is powered up. Any memory element that is not initialized/reset after a power-on, and does not retain its previous state, will have an unpredictable initial value. These unpredictable initial values are easily represented in simulation as ‘X’ as opposed to ‘0’ or ‘1’. However in emulation, they must be randomly represented as ‘0’ or ‘1’ at each different run of the test.

Power switches

Power switches control the flow of electrical current from the supply to the power domains. They are managed through signals sourced from a power management authority within the SoC. It should be noted that power switches establish the flow of current almost immediately after receiving a request, but the beneficiary power domain might not become operational so quickly. Depending on the electrical characteristics of the beneficiary power domain, there is a finite amount of time before the input supply voltage reaches an optimum operational threshold. Care must be taken when accessing the newly powered-on power domain.

Power supply network

The concept of power domains and multi-voltage creates a need for a network and hierarchy of power lines, switches, and on-chip and off-chip power regulators. Since some of the power lines can be controlled to supply different voltage levels at different times, care must be taken when coordinating the voltage levels with clock frequencies for a given power state.

Memory

Memory blocks, arrays, and FIFOs spend energy through leakage current. Therefore, powering off memory blocks is sometimes one of the methods utilized for conserving power. However, the content of the memory blocks, arrays, and FIFOs are lost during the power off, and upon the return of the power, the contents are unknown. As with the retention registers, the content of the memory after the power-on needs to be randomized in emulation.

IEEE 1801 (aka UPF)

Hardware description language (HDL) semantics do not consider power. Therefore the traditional flows lack the ability to incorporate the power intent of a design and do not accommodate the power-aware verification. As a result, the validation of power-minimization strategies was left to measurements and tests on the physical device. In 2007, Accellera introduced a Tcl[1] -based way of defining the power intent of a design, named Unified Power Format (UPF.) Later, IEEE 1801™-2009[2], a standard format for defining power intent, was introduced to the ASIC industry. Today, IEEE 1801 (UPF) makes it possible to design low-power integrated circuits, catch the functional errors that might have been induced by power-management strategies, and verify a low-power operation during simulation and/or emulation.

The role of IEEE 1801 (UPF) is to incorporate the power-management constructs such as power switches, isolation cells, retention registers, etc. in the design as early as possible, so that verification and debug of the SoC can be started earlier in the timeline and potentially save it from mistakes, big or small. Doing so enables the verification engineers to test the SoC with power-management considerations.

Power-Intent and Functional-Intent Verification

While functional intent is expressed in RTL and defines the architecture, application, and usage of standard interfaces for the design, power intent is captured in IEEE 1801 and defines the power domains, supply rails, power state strategy, operating voltages, isolation cells, level shifters, power switches, and retention registers.

The power intent has a great influence on the functionality of the SoC and as such should be defined alongside the functional intent. Perhaps most, if not all, of the functional-verification tests should become power aware.

Without incorporating IEEE 1801 into the flow, a great many features of power-management measures will remain un-tested and potentially leave the resultant silicon SoC with sometimes disastrous bugs, such as:

- The ROM content (e.g., BOOTROM firmware) neglects to turn on a power domain, but since simulation or emulation of RTL without UPF does not have any concept of power, the firmware goes on to access the domain without any problems
- A wrong clamping value on an isolation cell will not be observed without powering down the domain sourcing the signal
- Neglecting to enable a register to retain its value (retention register) during a power down will cause unpredictable and hard-to-find system bugs

SoC-Level Power-Intent and Functional-Intent Verification

Block-level verification, especially when performed with randomly generated stimuli, is extremely useful. However, IP blocks that are tested in isolation might not function properly when connected to other IP blocks to form a SoC. Plenty of problems can be introduced when connecting IP blocks together. Additionally, some IP block design flaws might be revealed only after the IP is configured and connected to other IP blocks. In particular, some power-management techniques such as clock auto gating, which play a big role in the SoC-level functionality, tend to reveal their design flaws only when tested with the actual application software in silicon.

1 Tool Command Language (conventionally spelled "Tcl" rather than "TCL") is a scripting language

2 Donated by Accelera and developed by IEEE

"Hindsight is 20/20 and Foresight is Myopic"

The design flaws of the past and sometimes-incomplete interpretations of the design specification are inadequate to uncover design flaws.

Example 1

Consider the block diagram of a SoC consisting of a processor and a memory controller (Figure 6). The memory controller is designed to monitor the data path and automatically turn off (auto gate) the clock to its internal circuitry after detecting a period of inactivity. It is also designed to turn the clocks back on again, if a transaction destined for the external memory appears on the data path. Since the controller is only monitoring the data path, any access to the registers of the controller (control path) while the clocks are off will cause problems.

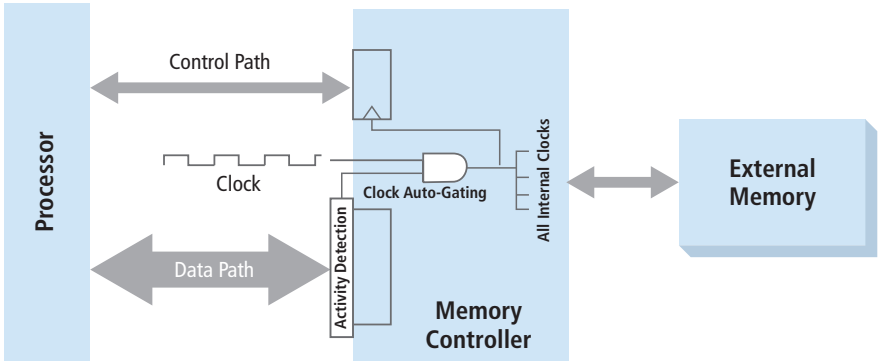

Figure 6. Memory controller example

To further complicate the situation, the access through the control path while the clocks are off might be considered unreasonable and neglected[3] in the block-level testbench, as memory controller registers are normally only initialized once and left alone for the duration of the operation. However, the modern memory controllers offer quality of service (QoS) and input/output memory management unit (IOMMU) features that are sometimes adjusted by the application software (e.g., operating system) for a variety of reasons during the normal operation of the SoC. Depending on when the processor accesses the QoS or IOMMU registers of the memory controller, the access might end up with a bus error or even worst, a hang.

Example 2

Consider the block diagram of a SoC that controls the clock gating through the processor (Figure 7). The processor writes a register inside the slave block and turns the clock off or on. As a consequence of SoC-level (top-level) architectural design, the control path (register access) might have a longer latency than the data path.

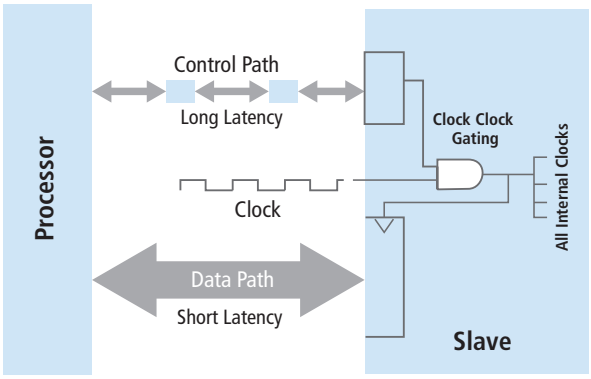

Figure 7. Clock gating example

A bus error, hang, or unexpected reaction can result, depending on how fast the processor accesses the slave through the data path after enabling the clocks by writing into a clock gate control register.

3 By constraining the generation of random stimuli, to avoid error messages

A SoC, whether partially or fully represented by RTL, can be verified through simulation testbenches[4]. Or, it can be verified in simulation or emulation through the execution of test software directly on the SoC's embedded processor.

SoC-level simulation test benches come short of properly testing the interaction between the embedded SoC processors and the rest of the SoC. Additionally, as large as some of the SoCs are and as many I/O interfaces that they have, it will be nearly impossible to provide the proper constraints for randomly generated stimuli for the inputs of the SoC while coordinating with the bus functional models (BFMs) substituting the embedded processors.

Hence, the SoC-level verification (functional or power) is best performed by executing code on the embedded processors within the SoC that is connected to external peripheral models.

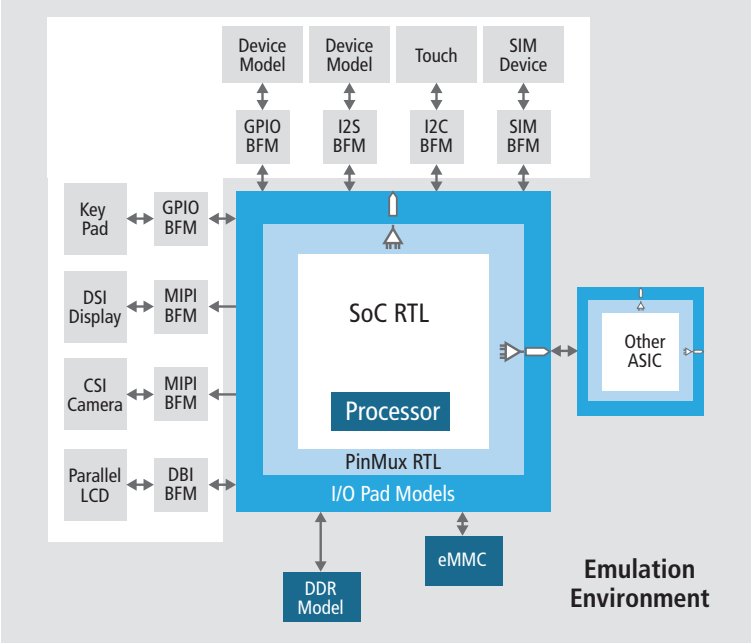

Figure 8. Emulation environment

Verification Test Software for Functional- and Power-Intent Verification

Verification engineers normally create a number of individual software tests, each representing a test case for an identifiable corner case and/or feature appearing in the functional and power intents. These test cases are executed after a long and exhaustive setup and configuration. In many instances, the duration of the test case itself is much shorter than the duration of setup required for test. The verification team can elect to shorten this long setup and/or configuration period by utilizing special techniques in simulation and/or emulation. The most common of these special techniques are Tcl scripts for initializing memory segments (e.g., zero initialization, MMU tables) and serving as logging mechanism for test results. These aforementioned techniques are not necessarily needed[5] by emulation platforms and might not even be practical for FPGA-based emulation platforms. Other techniques not practiced by all verification teams, involve "FORCE," a method through which the setup and configuration is bypassed, and the state of the SoC is forced to a known value.

The above mentioned test cases are specifically designed to only interact with as few hardware resources as necessary for the specific test. They specifically avoid utilization of operating systems, which might take the focus away from the targeted test by performing non-related tasks. Although not widely practiced, it is very possible to execute the same software test in simulation, emulation, and the resultant silicon SoC.

4 Replacing the embedded processors with BFMs

5 Emulation speed is fast enough to make these Tcl scripts unnecessary.

Role of application software in verification

It is impractical to manually create test cases for all possible corner cases identifiable for all of the SoC's features in functional and power intents. The impracticality is partially due to the fact that most often in some corner cases and during the occurrence of certain events, the use of a feature is not obvious and remains unknown until the silicon SoC is under the control of the application software. To ensure that the neglected non-obvious corner cases do not show up later, some verification teams use the application software[6] as the ultimate test software for the power-intent and functional-intent verification and as a complementary test to the verification test software. However, a few problematic factors might prevent the verification engineers from using or delay the use of the actual application software for verification:

- Due to the simulation speed, the execution of the application software in simulation is impractical. However, it becomes much more practical at emulation speeds. Especially if the application software is stripped of some of its features that might not have an influence on the usage of hardware. For example, the decompression of software or data before loading into memory can be avoided by storing it as decompressed image.
- The actual application software might not be available at the time of RTL verification, usually due to unavailability of drivers for the new hardware. Sometimes, the basic framework of an older version of the application software combined with the drivers for the new hardware will be a very close fit.
- Unavailability of peripheral models that are capable of interacting with the application software. Fortunately, this big task can be overcome because it is not a technical issue. The good news is that, when available, the models can be reused for other projects.

Close-to-actual application software

In lieu of the actual application software, a reduced functionality application software, called "close to actual," can be used as a complement of functional-intent-driven verification tests.

Utilizing Assertion in Power-Intent Verification

Assertions are very useful in SoC-level verification. They can be utilized to monitor and check aspects of the SoC's design that might not be directly verifiable. The following two examples of power-aware tests demonstrate the merit of assertion in monitoring conditions that would have been ignored otherwise:

- If a clock signal was really turned off during a desired time window
- If the power to a power domain was actually turned off

In conjunction with Tcl scripts and GPIO pins on the SoC, the assertions can be coordinated with the verification test software running on the embedded processor to provide an insight to whether the clock or power domain were turned off at the right time.

MP3 Player: Making a Case for Co-Verification of RTL and Software

This example illustrates the importance of verifying low-power management logic by running real software with the hardware. Subtle problems can be easily missed by just verifying hardware alone without software.

SoC architecture and application software as a pair have a huge influence on power

Figure 9 illustrates an SoC performing MP3 player functionality. This example demonstrates that the emulation of the MP3 player's RTL, while executing the real MP3 player software, enables the designers and software engineers to achieve a lower power design by adjusting the hardware and software architecture as a pair.

6 The software that eventually will be running on the physical silicon

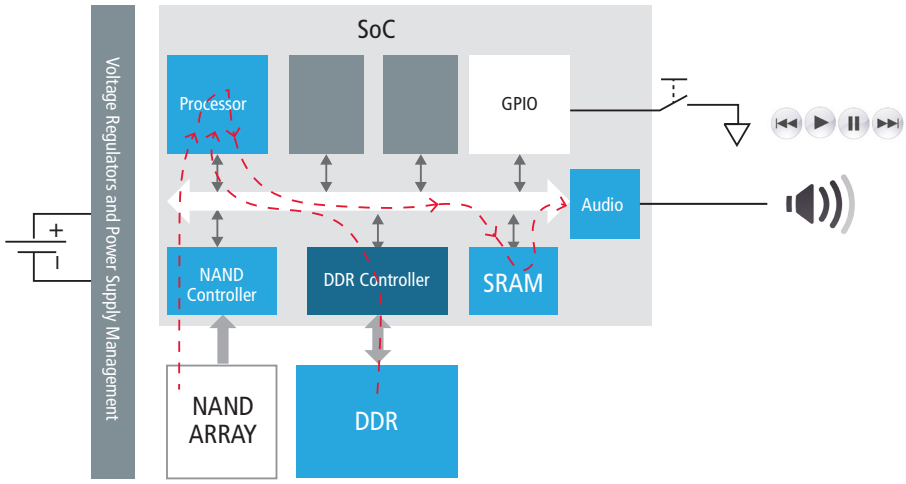

Figure 9. MP3 player example

MP3 player power-management strategy

The pertinent power management aspects of the SoC include:

- A single processor, implemented in its own power domain. The processor can be programmed to operate at three different clock frequencies, the highest of which requires voltage increase on the processor power.
- GPIO pins are supported within a single power domain that is always on. The GPIO pins are connected to the play, pause, forward, and reverse buttons.
- Audio (a mixed-signal block) is partially represented by RTL for verification. It is contained in a power domain and can be turned on when the audio playback is activated and turned off when one of the pause, forward, or reverse buttons is pressed.
- NAND and DDR controllers are housed in their own respective power domains and can be independently clock gated or turned on or off.
- SRAM (internal) is housed in own power domain and is kept on for the duration. The supply voltage to SRAM can be reduced from operation mode to retention mode (no activity but the contents are retained).
- The NAND array and the external DDR memory are supplied by individually controllable power supplies. They can be turned on or off by the external voltage regulator and power-supply management unit.
- The external voltage regulator and power supply management unit (PMU) is activated into a default supply configuration upon power-on reset. However, to control the regulators, it does obey commands from the SoC through a serial link.

Changes to the SoC after analyzing the MP3 player's estimated power consumption

There are few tools for both simulation and emulation platforms that help in estimating power consumption of a SoC prior to tapeout. The traditional simulation-based power-analysis tools provide their estimates for a short duration of activity and only at IP block level. On the other hand, the emulation-based power-estimation analysis tools allow power analysis of the entire SoC (depending on the capacity of the emulation technology) for much longer periods of time.

Using a tool to estimate the dynamic-power consumption of the MP3 player SoC at different phases of the its operation, while executing the actual MP3 player software, enables the designers to arrive at the following architectural and design decisions:

- Make the SRAM large enough for at least four frames of decoded MP3 audio[7], so that the audio block can stream audio data from the SRAM rather than from DDR. This way, the DDR can self-refresh while the processor sleeps for a longer period of time.
- The processor will use the same amount of dynamic power for decoding of the MP3 frames at the lowest and mid frequencies (same supply voltage and number of toggles,) and somewhat more for the highest frequency[8]. It is more efficient to operate the processor at the mid-frequency and let it decode the MP3 faster and subsequently go sleep sooner and for longer periods of time[9].

Utilizing a simulation-based static-power analysis tool, and by augmenting the resultant numbers for each block with the dynamic-power numbers for each phase of the operation, the pattern shown in FIgure 10 emerges.

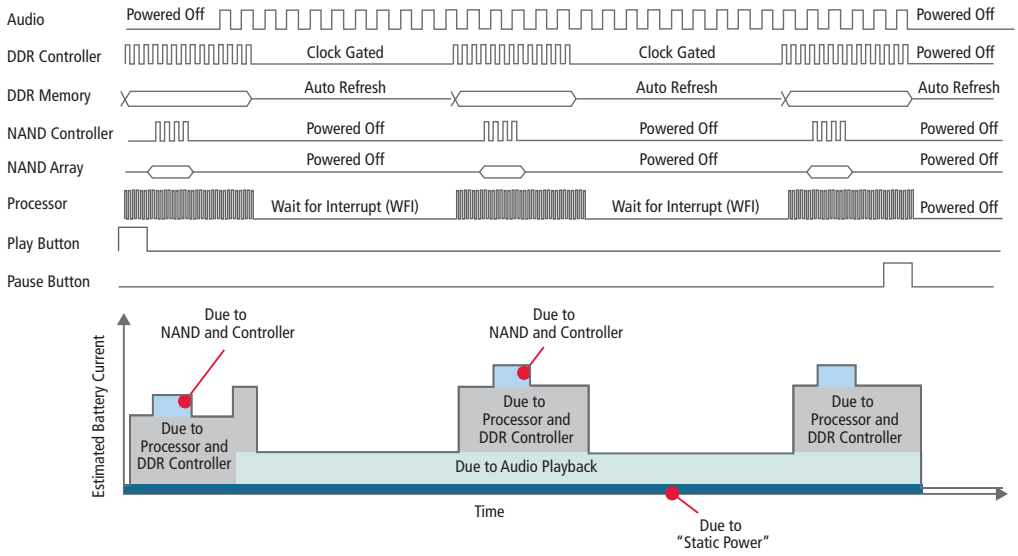

Figure 10. Power consumption

Co-verification of RTL and software requires visibility into RTL

The speed and the interactive nature of emulation platforms make them ideal for co-verification of RTL and application software. However, the ability to examine any given node in the RTL is by far the most useful feature that an emulation platform can offer for co-verification.

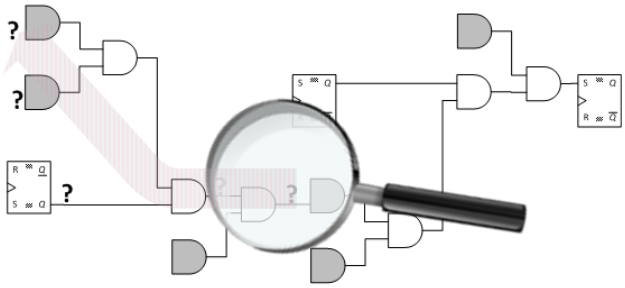

Figure 11. SoC logic visibility

7 The rate at which the audio stream is played back is constant and much slower than the rate at which the processor can read the coded data from the NAND and decodes it. Therefore by reading and decoding more audio frames from the NAND and back-filling the audio block, the more time all blocks except for audio can rest and save power.

8 The MP3 decode algorithm is the same for all operating frequencies, therefore the total number of state transitions for the processor as an isolated unit remains the same regardless of the rate of transitions (i.e., frequency). However, since the supply voltage to the processor (Vp) should be increased (ΔV) for the highest frequency, each state transition will have additional $C_l.\Delta V(2.V_p+\Delta V)$ dynamic power consumed.

9 After the processor has read the data from the NAND, it can turn off the NAND and, a bit later, the NAND controller. The processor then decodes the MP3 data and fills up the SRAM to be played by the audio block. After the SRAM has received all of the decoded audio stream, the processor goes to sleep (in wait for interrupt (WFI mode), which in turn causes the DDR controller to put the external DDR into self-refresh mode and drop its request for clock (clock gate.) The audio block will wake the processor up via an interrupt, just before it runs out of buffered data.

A Better Tool for Co-Verification of Low-Power Designs with IEEE 1801

An ideal platform for co-verification of low-power designs with IEEE 1801 and application software is the Cadence Palladium XP II platform. It offers design, verification, and software engineers the flexibility to achieve fast bring-up, low-power features, clock/auto gating and frequency scaling, memory and FIFO randomization, Dynamic Power Analysis (DPA), and IEEE 1801 (UPF) support.

Fast bring up

The Palladium XP II platform can compile at very fast speeds from RTL to downloadable database (DB) in about an hour in a single Linux workstation. For a 100+ million-gate SoC design, design, verification, and software engineers have a platform with full visibility and controllability running at tens of thousands times faster than software simulation, in just a few hours. During the early phase of the project, new RTL drop can be downloaded and is ready to be tested in the Palladium XP II platform in a matter of hours, not days. Implementing low-power logic in the Palladium XP II platform is simple. During the Palladium XP II platform's compile step, the designer just needs to specify the read_power_intent command with low-power intent files.

read_power_intent <1801 files>

Figure 12 illustrates a comparison between a normal RTL compile flow and the low-power instrumentation of the RTL compile flow.

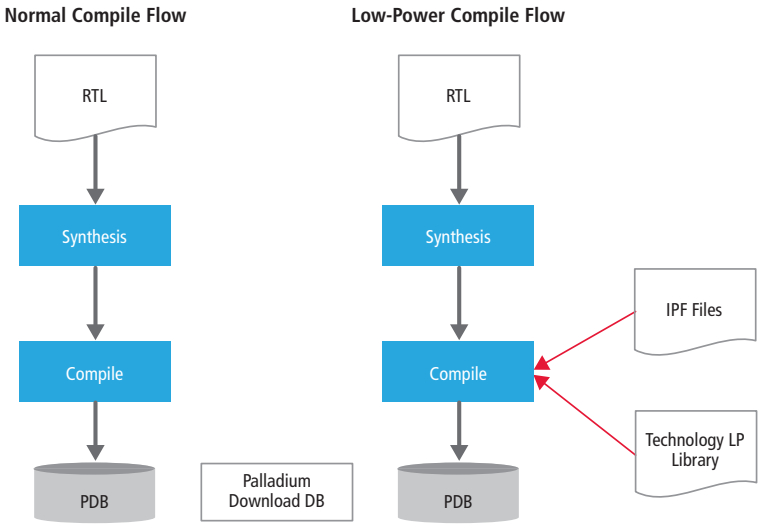

Figure 12. Palladium Low-Power Flow diagram

Low-power debug features

Debugging logic functional issues in a multi-million-gate SoC design is hard. Debugging low-power verification issues in a multi-million-gate SoC design is even harder. The problem can be in multiple places, including the power management unit(s), low-power software, low-power architect, and/or IEEE 1801 low-power format files. The Palladium XP II platform has several features specifically for debugging low-power related issues, from syntax checking of IEEE 1801 low-power format files, built-in low-power (lp) debug commands, probe command options for probing low-power control and instrumented signals such as power domain power state, isolation memory retention modeling, retention controls, etc., to a dedicated low-power debug GUI.

The low-power debug GUI assists in four areas of verification. The Runtime tab is for runtime control of the design and verification, and allows the designer to set supply states and voltages of supply nets and supply ports, list isolation strategies for pins and retention strategies for elements, create SDL[10] expressions for IEEE 1801 that can be used either as an SDL trigger expression or as part of an SDL program, and get the current state of and then probe and upload IEEE 1801 objects (supply nets, supply ports, supply sets, power state, power domains, isolation, and

10 State Description Language (SDL), is a run-time debugging feature that allows, based on design signal values, to control several aspects of the run such as when to sop, or when to capture probed data for waveform tracing. It also allows schedule execution of Tcl or Palladium commands during run-time using "EXEC" facility.

retention rules). The Illegal Events tab allows the designer to enable/disable illegal event checking if, for example, retention values get corrupted before they can be restored. The Power Intent tab allows the designer to bring up the original IEEE 1801 file as well as view the IEEE 1801 objects and attributes with their properties in a tabular format or a tree format. The Power Supply tab's graphical view of the power supply network, including supply nets, supply ports, supply switches, and power domains, can be used to view one supply net at a time or all of them together.

Clock gating, auto gating, and frequency scaling

The Palladium XP II platform is a cycle-accurate functional emulation platform. Hence, it supports all clock gating, auto gating, and frequency scaling low-power schemes. It also supports unlimited clocks with unlimited frequencies. This support frees design, verification, and software engineers to explore various clock gating, auto gating, and frequency scaling scenarios. These functionalities can be coupled with DPA to calculate the expected power savings of these low-power schemes.

Memory and FIFO randomization

In a software simulation, when a power domain is shut down, the content of all logics including memories and FIFOs are set to 'X'. Unfortunately, it is very costly to model the 'X' state in a hardware emulation platform. The next-best solution to setting all powered-down logics to 'X' is to set them to some random value. While it is relatively easy to set registers to random values inside a powered-down domain, it is quite difficult to set memories and FIFO contents to random values.

One approach is to load the contents to memories and FIFOs with predefined randomized values, but the load operations can be time consuming due to the large number of memories and FIFOS that are in powered-down domains. Another approach is to create external logic that is triggered when the power domain is in shutdown mode. This external logic will update the memory and FIFO contents at emulation speed. This approach, though re-usable, requires additional hardware resources, requires code development for the emulation platform, and also requires additional debug to make sure that it functions correctly. In the Palladium XP II platform, the software provides a feature using a special array primitive that can randomize arrays in one to two clock cycles with minimal capacity overhead. This feature was implemented to avoid having to create user-defined logic specific for memory randomization and to reduce the performance impact that comes with memory operations.

Randomizing the contents of memories and FIFOs is very important when validating low-power software. Unlike functional problems, when low-power logic does not function (i.e., content of memories and FIFOs remain the same), logic functionality might perform perfectly. The low-power software's initialization function is not tested, so the state of the power domain before shut down is same as just after power up. This "sameness" leads to false confidence on the software engineer's part that the code is fully tested and leads to a long debug cycle during real silicon bring-up. The Randomize Array feature of the Palladium XP II platform allows the software engineer to fully validate the code before real silicon is back.

DPA

The Palladium XP II platform offers the Dynamic Power Analysis (DPA) tool for high-performance, cycle-accurate, and full-system power analysis of a design under test (DUT). It works hand-in-hand with Cadence RTL Compiler, or any other third-party power tool that the designer might be using, to present designers with static power consumption as well as dynamic power consumption numbers. The designer provides Liberty files for a given technology as well as a power consumption table for all macro cells such as memories. When the designer runs the application software in the Palladium XP II platform, the DPA tool can identify hot spots, peak power consumption, and average power consumption. The latest DPA tool also provides both forward SAIF and backward SAIF files, to be further analyzed by backend tools.

IEEE 1801 (UPF) support

The latest version of the Palladium XP II software supports most of UPF 1.0; IEEE 1801, 2009; and IEEE 1801, 2013. In addition, it supports Common Power Format (CPF) 1.0, CPF 1.1, and CPF 2.0. Much of the Palladium XP II platform's IEEE 1801 supports are tightly coordinated with Cadence Incisive® tools to reduce the learning curve and to allow seamless migration from the software simulation world to the hardware emulation world.

Summary

Low-power verification is increasingly a prime area of focus for most designs and the big battleground to differentiate efficient designs from non-performing ones. Various power techniques have been discussed in this paper pertaining to achieving the best results in a complex hardware/software interactive world. IEEE 1801 (UPF) power instrumentation combined with DPA techniques ensure that designs fully comply with the power requirements specified in the architecture. The Palladium XP Series acceleration and emulation platform from Cadence offers a wide range of features that help in fully verifying the design and tapeout well ahead of time.