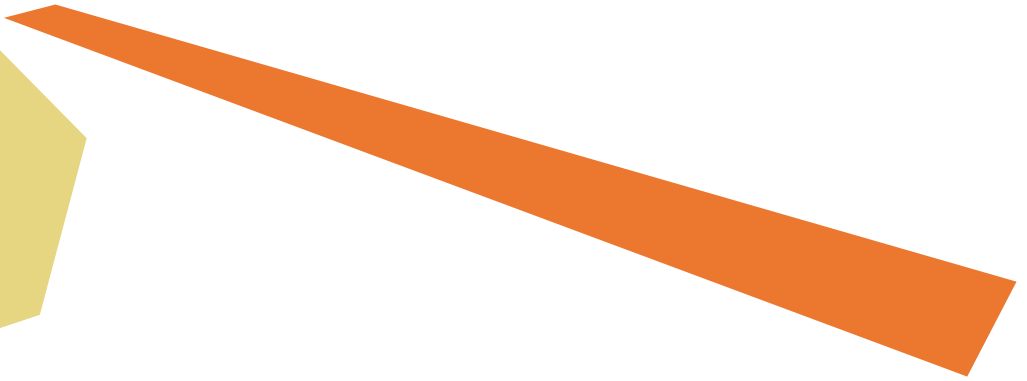
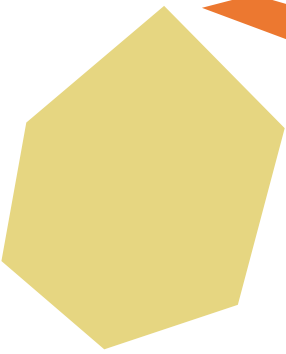


WHITE PAPER

ARCHITECTING, DESIGNING,
IMPLEMENTING, AND VERIFYING
LOW-POWER DIGITAL
INTEGRATED CIRCUITS



INTRODUCTION

In recent years, power consumption has moved to the forefront of digital integrated circuit (IC) development concerns. The combination of higher clock speeds, greater functional integration, and smaller process geometries has contributed to significant growth in power density. Furthermore, with every new process generation, leakage power consumption increases at an exponential rate.

It is common to think of low-power designs only in the context of handheld, battery-powered devices such as personal digital assistants (PDAs) and cell phones. And it is certainly fair to say that this class of device is at the top of low-power development concerns. In reality, however, power consumption (and corresponding heat generation) is also of significant interest to semiconductor segments with fixed installations, such as networking, set-top boxes, and computing devices. For example, the InformationWeek "Power Surge" article on 27 February 2006 reported that data center electricity costs are now in the range of US\$3.3 billion annually, and it can cost more to cool a data center than it does to lease the floor space in which to house it. Additionally, consumers increasingly demand quieter devices for their living rooms and desktops, and low-power designs help manufacturers eliminate noisy cooling fans from set-top boxes and other products.

Over recent years, a wide variety of techniques have been developed to address the various aspects of the power problem and to meet evermore-aggressive power specifications. These include (but are not limited to) the use of clock gating, multi-switching threshold (multi-Vt) transistors, multi-supply multi-voltage (MSMV), substrate biasing, dynamic voltage and frequency scaling (DVFS), and power shut-off (PSO). *Figure 1* illustrates the power, timing, and area tradeoffs among the various power management techniques.

Figure 1:
Power tradeoffs

Power Reduction Technique	Power Benefit	Timing Penalty	Area Penalty	Methodology Impact			
				Architecture	Design	Verification	Implementation
Multi-Vt Optimization	Medium	Little	Little	Low	Low	None	Low
Clock Gating	Medium	Little	Little	Low	Low	None	Low
Multi-Supply Voltage	Large	Some	Little	High	Medium	Low	Medium
Power Shutoff	Huge	Some	Some	High	High	High	High
Dynamic and Adaptive Voltage Frequency Scaling	Large	Some	Some	High	High	High	High
Substrate Biasing	Large	Some	Some	Medium	None	None	High

As expected, the use of more advanced approaches—such as MSMV and PSO—reduces power consumption, but at the same time increase the complexity associated with design, verification, and implementation methodologies. Although using a single technique in isolation could be relatively simple, often a combination of these techniques must be used to meet the required timing and power targets. Using multiple techniques concurrently could result in an extremely complex design flow. A key requirement is consistency throughout the flow such that the use of one technique preserves any gains from other techniques. This also requires a good understanding of the various low-power techniques and their respective tradeoffs, as highlighted in *Figure 1*.

Furthermore, “low power” isn’t just something that is “bolted” on at the end of the development process. To meet aggressive design schedules, it is no longer sufficient to consider power only in the implementation phase of the design. The size and complexity of today’s ICs makes it imperative to consider power throughout the design process, from the chip/system architectural phase; through the implementation architecture phase; through design (including micro-architecture decisions); and all the way to implementation with power-aware synthesis, placement, and routing. Similarly, to prevent functional issues from surfacing in the final silicon, power-aware verification must be performed throughout the development process.

A key enabler of a modern power-aware design flow is the ability to capture and preserve the intent of the chip architects and designers throughout the design flow. This requires a common specification format that can be used and shared across the entire design chain, from architectural specification (“this block has three power modes”) to verification (“will the chip recover if these blocks are put to sleep in this order?”).

The current state-of-the-art in such a specification is the Common Power Format (CPF), which is managed under the auspices of the Silicon Integration Initiative (Si2) consortium’s Low Power Coalition. CPF is a new design language that addresses limitations in the design automation tool flow. It provides a mechanism to capture architects’ and designers’ intent for power management and it enables the automation of advanced low-power design techniques. CPF allows all design, implementation, verification, and technology-related power objectives to be captured in a single file and then applies that data across the design flow, providing a consistent reference point for design development and production.

There are three major benefits of using CPF to drive the design, verification, and implementation steps of the development flow. First, it helps designers achieve the required chip specs by driving the implementation tools to achieve superior tradeoff among timing, power, and area. Second, by integrating and automating the design flow, it increases designer productivity and improves the cycle time. Third, by eliminating the need for manual intervention and replacing *ad hoc* verification methodologies, it reduces the risk of silicon failure due to inadequate functional or structural verification.

This paper introduces the various aspects of power-aware design during chip/system architectural specification, power architecture definition, design, physical implementation, and verification.

CHIP/SYSTEM ARCHITECTURAL SPECIFICATION

Power-aware design starts with the architectural specification of the chip/system. This is where the most significant tradeoffs can be made. To define the most appropriate architecture for a design, it is necessary to understand the algorithms being used and the “space” of the application. To address this, it is now common practice to model the system at a high level of abstraction and to use this model to determine bandwidths (how much data needs to be moved around) and how much processing power is required to achieve these bandwidths.

One critical task is to partition the system into its hardware and software components. Hardware implementations are fast and consume relatively little power, but they are “frozen in silicon” and cannot be easily modified to address changes in the standards or the protocols. By comparison, software implementations are slow and consume a relatively large amount of power, but they are extremely versatile and can be modified long after the chip has gone into production.

Another area that is receiving increasing attention is the creation of the software compiler so as to generate power-tuned machine code (e.g., generating code that occupies memory in a non-sequential manner such that only a single address bit is toggled when accessing “adjacent” words in the machine code).

Regarding the hardware portions of the design, “block” partitioning should be defined in the context of the types and quantities of data that are flowing through the system. The goals here should be to minimize both inter-block communication and the frequency at which the majority of signals will be switching.

Also at the architectural stage, evaluations should be made as to which blocks are not performance-critical, so that they could potentially be run at a lower voltage and/or frequency to conserve power. In some designs, certain blocks may be suitable candidates to utilize voltage-frequency scaling techniques, in which case it is necessary to determine (and document) how the performance-voltage-frequency feedback mechanism will function. Similarly, in some designs, certain blocks may be suitable candidates for “sleep mode” or completely shut down to conserve power when they are inactive. This means that the architects will define different “modes” and then specify which blocks will be on or off (or asleep) in each mode, or even blocks that have different power/performance requirements for the different modes.

Another key consideration is IP selection, which relates to both internal and third-party IP. In the case of one real-world design that exceeded its power budget, an unexpected amount of leakage was tracked down to the memory IP obtained from a third party. Thus, in addition to its core functionality, an IP block should be evaluated in the context of the device architecture’s low-power requirements. Is it required for this block to be capable of being placed in sleep mode and/or completely shut down, for example? And if so, does this IP block support the required functionality?

Clock gating also must be planned during the architectural phase. This technique can significantly reduce the design’s dynamic power consumption since the clock trees can account for one-third to one-half of a chip’s dynamic power consumption.

Another consideration for today’s multi-million-gate designs is the interconnect mechanism used to link the large numbers of components. For instance, conventional synchronous bus architectures constantly burn power, even if they aren’t actually moving any data around. One solution is to move to a globally asynchronous locally synchronous (GALS) architecture. In this case, data flows as fast as possible through the self-timed (asynchronous) interconnect network because there is no waiting for clock edges, and the power consumed by the buses is dictated by their traffic loads. Furthermore, the clocks associated with the synchronous blocks can be stopped (or gated) when those blocks are not being used.

The problem is that implementing any low-power technique involves time, engineering resources, and risk, and all of this is extremely design dependent. What will be the return of using each technique? A particular technique may convey tremendous power-reduction benefits with one design and negligible payback with another design, while adding the same amount of risk to both. Furthermore, one technique may impact the effects of another technique unless both approaches are considered “holistically.” Thus,

it is extremely important to have a good understanding of the various low-power techniques and their respective tradeoffs. Also, it is necessary to have a good understanding of any impacts each power-reduction technique will have on other chip goals such as timing and area.

These considerations are extremely complex and require the design team to adopt advanced low-power techniques without compromising their productivity or increasing the risk associated with the project. Thus, a key requirement is the ability to capture any power-related architectural decisions in a form that can be accessed and used throughout the remainder of the flow. This requires the use of a common specification format—such as CPF—that can be used and shared across the entire design chain, from architectural specification (“this block has three different power modes”) to verification (“will the chip recover if these blocks are put to sleep in this order?”).

Keeping the power profiles of the various blocks in the CPF and separate from the RTL is a key consideration when re-using blocks (in the form of RTL code) in different applications—or even different parts of the same design—where power requirements may differ.

POWER ARCHITECTURE

Following the definition of the chip/system architectural specification, the next step in the development process is to refine the power architecture. For example, the architectural specification may specify that a certain block should be implemented in such a way that it is capable of being completely powered down. In the power architecture portion of the process, the team will determine just how *often* this block is to be shut down, and also any interdependencies among this block and other blocks and modes.

To place this in perspective, before a certain block is powered down, it may be necessary to first power down one or more other blocks in a specific order, and to ensure that each block is completely powered down before a subsequent block is powered down. Similarly, when this portion of the design is restored to its operating condition, it will be necessary to ensure that the various blocks are powered up in a specific order.

Compounding the problem is the fact that different blocks may be powered down in different combinations and in different sequences while the device is in different operating modes. Furthermore, the power architecture has to be capable of handling a reversal of the power-up or power-down processes in mid-sequence. Following some period of inactivity, for example, the system may decide to initiate a power-down sequence. While in the middle of this sequence, however, an external stimulus may be detected (say, an incoming signal on a wireless device) that requires this portion of the design to be functioning. What should happen in this case? Will the system be forced to complete the entire power-down sequence, or can it abort the sequence and immediately commence to power blocks up again (in the defined order, of course).

The power architecture phase relies on system-level switching activity and profiling data captured during the chip/system architecture process. In addition to powering down blocks and/or placing them in sleep modes, other aspects of the power architecture include the following:

- Multi-supply multi-voltage (MSMV)
- Power shut-off (PSO)
- Substrate biasing
- Dynamic voltage and frequency scaling (DVFS)
- Clock gating

Applying any of these techniques requires a significant amount of effort and involves a certain amount of risk. In the case of MSMV implementations, for example, it will be necessary to consider low-power cells, level-shifter cells, and so forth. Similarly, in the case of PSO, designers have to choose between “simple power shut-off” where everything in the block is powered down, and “state retention power shut-off,” in which the bulk of the logic is powered down but key register elements remain “alive.” This latter technique can significantly reduce the subsequent boot-up time, but state-retention registers consume power and also have an impact on silicon real-estate utilization.

Substrate biasing is typically applied only to portions of the design. The idea here is that a functional block typically doesn’t need to run at top speed for the majority of the time, in which case substrate biasing can be applied, which causes that block to run at a slower speed but with significantly reduced leakage power.

DVFS is used to optimize the tradeoff between frequency and power by varying the voltage or frequency in relatively large discrete “chunks.” For example, the nominal frequency may be doubled to satisfy short bursts of high-performance requirements or halved during times of relatively low activity. Similarly, a nominal voltage of 1.0V may be boosted to 1.2V to improve the performance, or reduced to 0.8V to reduce the power dissipation. Each of these scenarios has to be tested in the context of surrounding blocks, which may themselves switch from one mode to another.

Even the well-understood technique of clock gating—which was originally planned in the chip/system architectural exploration phase and now has to be followed up in the power architecture phase—can pose a complex problem. For example, should clock gating be performed only at the bottom of the tree (the leaf nodes), at the top of the tree, in the middle of the branches, or as a mixture of all of these cases? There are tools to make this more efficient by moving the clock-gating structures upstream and/or downstream and performing splitting and cloning, but all of this adds substantially to the task of physically implementing the clock tree and also to the verification problem.

As market requirements continue to impose stringent performance, area, and power requirements, the majority of designs will require a combination of power management techniques. However, things that would probably work without any problems if only a single technique were used become much more complicated when multiple techniques are used concurrently.

The end result is that the power architecture phase requires access to a variety of exploration and “what-if” analysis tools, and all of these tools need to be able to account for effects such as MSMV and substrate biasing. For example, timing analysis and power estimation engines should be able to accommodate “what-if” scenarios such as “what happens if the voltage on Block A is lowered to 0.8V at the same time as substrate biasing is applied to Block B?”

As the power architecture team evaluates different scenarios and makes decisions—such as which blocks will be running at which voltage and/or frequency and with what biasing (if any) in which operating modes—these decisions need to be captured in the Common Power Format (CPF) file to be used in the downstream design, implementation, and verification portions of the flow. Furthermore, the CPF facilitates performing “what-if” analysis, because it allows engineers to easily and quickly make a change in one place that will subsequently “ripple” throughout all of the design, analysis, implementation, and verification tools.

POWER-AWARE DESIGN

In this context, “design” refers to the portion of the flow where—taking the results from the power architecture phase—the design engineers capture the RTL descriptions for the various blocks forming the design. The designers associated with each block are responsible for ensuring that block will meet its functional, timing, and power requirements while using the minimum silicon real estate.

This means the designers need to know the target voltage level(s) for their block so that they can understand the various timing and power challenges. If the system as a whole employs multiple voltage levels, level shifters must be inserted between domains, and the designers must know if these are present in their timing paths because these cells could result in significant cell delays.

Designers need the ability to quickly and easily explore and evaluate alternative micro-architectures, such as unraveling loops, resource sharing, varying the number of pipeline stages (more stages result in higher performance at the expense of increased power consumption, area utilization, and latency), using hardware accelerator(s) versus microcode, and so forth. With regard to power-centric design, this involves the ability to capture toggle values to gain an accurate estimation of power—first with the RTL representations and later at the gate level.

Designers also need to know if their block will be a power-shutdown block, as this will affect design and implementation. For example, how quickly is it required for the block to enter into—and recover from—the shutdown mode? Using state retention registers offers the fastest recovery, but these registers require more area than their non-retention counterparts do. If state retention is to be used, will all of the registers in the block use them, or will it be used only for a subset of the registers? It is important for designers to be able to perform quick “what-if” analyses to determine the optimum number of state retention registers. Also, using state retention registers will require a special state retention power grid, which is likely to add some area overhead to their block.

Even for a common technique, such as clock gating, there are multiple alternative scenarios (e.g., some design teams code the clock-gating structures directly into their RTL, while others implement clock gating using synthesis technology during the implementation phase, and some teams use a mixture of both these techniques).

Formal analysis can play a very important role in verifying the power-centric functional aspects of the design. As each block is coded, power-centric assertions can be associated with that block; for example: “when this block is instructed to enter its power shutdown mode, the ‘isolation’ signal should go active first followed—after some minimum delay—by the power shutdown signal going active. Also, assertions will be associated with both clusters of blocks and at the full-chip level to ensure that blocks power down and power up in specific sequences depending on the current operating mode.” Formal analysis is of particular interest when identifying potential livelock/deadlock conditions that result from blocks being powered up and/or powered down, thereby causing states to appear between blocks that would not appear during the normal operation of the device. This type of issue cannot be exhaustively checked by any means other than formal verification.

One important consideration is that every power-centric aspect of a design would traditionally have been hard-coded into the RTL. This can cause problems, because the same block can be used in multiple implementations with different power intent. For example, a block may be used at a certain voltage level and/or have a certain power shut-down profile in one instantiation in the design, and a different power profile in another instantiation in the same design. And, of course, blocks of RTL are often re-used in multiple chips whose power requirements may be vastly different.

Conventionally, implementation-specific intent of this type has been captured using constraints and tool scripts, or by being hand-coded directly into the RTL implementation. However, this approach is prone to error, especially as things may change during the design process. This is a major advantage of the Common Power Format (CPF), which captures design intent that can be shared across the design, implementation, and verification tools throughout the flow.

POWER-AWARE IMPLEMENTATION

The implementation phase is where all of the hard work performed during the chip/system architecture, power architecture, and design phases comes to fruition with the aid of power-aware engines for logic synthesis, clock gating, placement, clock-tree synthesis, routing, and so forth. In addition, during the implementation phase, the logical and physical structures needed for the various power techniques are created. These include power grid synthesis, power plane implementation, and insertion of level shifters, switch cells, isolation cells, and state-retention cells.

Top-down multi-dimensional optimization: One key requirement for low-power design is the ability of logic and physical synthesis engines to concurrently optimize for timing, area, and power tradeoffs. For example, by swapping low-Vt cells with high-Vt cells, optimization engines can reduce power while reducing performance. However, power cannot be an afterthought to the optimization process where first the timing targets are achieved and later on cell swapping is done to reduce power. This means that right from the logic synthesis stage and all the way to routing, the optimization engines need to create structures that can meet performance targets while minimizing the use of low-Vt cells.

Similarly, in the case of MSMV designs, it is important that the optimization engines understand power domains top down so that they have full visibility to optimize across entire timing paths. The top-down optimization approach for MSMV designs leads to superior timing, area, and power tradeoffs. Furthermore, the ability to explore power-timing tradeoffs at the top level eliminates the need for multiple iterations that are common in bottom-up partitioning.

Clock-gating challenges: Though clock gating has been used for some time as an effective technique to reduce dynamic power, today's stringent power specs demand more sophisticated gating techniques. The most sophisticated form of clock gating currently available is multi-stage gating, in which a common enable is split into multiple sub-enables that are active at different times and/or under different operating modes. And, of course, the process is further complicated when it comes to clock trees that pass through different power domains, especially when one or more of those power domains are candidates for power shut-off.

Power-aware design planning: Power domains must be shaped and placed; power pads and switches must be placed and optimized; and power routing must be planned. Silicon virtual prototyping helps the partitioning process by minimizing the wirelength of the high switching probability wires, which lowers the dynamic power. This requires tools that can understand which wires contribute to the most capacitance and are able to find ways to minimize the interconnect capacitance through optimal partitioning and floorplanning.

Physical implementation of low-power structures: Multiple power domains in MSMV and PSO techniques require the insertion, placement, and connection of specialized power structures, such as level shifters, power pads, switch cells, isolation cells, and state-retention cells.

In particular, PSO requires a special set of implementation considerations. For example, designers need to make a tradeoff between using fine- and coarse-grained power gating. Fine-grained gating includes a power-gating transistor in every standard cell, which has a large area penalty but eases implementation. Coarse-grained gating reduces the area penalty by using a single gate to cut off power to an entire block, but requires sophisticated analysis to determine the gating current of the switched-off block. In addition, turning on several blocks at once could lead to the problems of ground bounce and latch-up current due to IR drop. This requires dynamic gate power analysis to better understand the turn-on characteristics of a block and minimize the rush current or ground bounce effects.

Signal integrity considerations: With increasing clock frequency and lowering of supply voltages, there is increasing sensitivity to signal integrity (SI) effects such as crosstalk-induced delay changes and functional failures. With the use of advanced power management techniques, SI analysis becomes even more complicated. For example, in a PSO design, a spurious signal caused by an SI aggressor could shut down an entire module. Similarly, the use of multi-Vt cells can lead to the creation of super-aggressors with low-Vt while weakening their high-Vt victims. Overall, multi-Vt, MSV, and PSO techniques require implementation and analysis tools to be power aware so they can account for SI effects.

Low-power formal verification: Low-power designs could fail due to a number of structural errors, such as missing level-shifters/isolation logic; redundant level-shifters/isolation cells; bad power-switch connections; and bad power and ground connections. Some possible functional errors include bad state-retention sleep/wake sequences and bad logic for power gating and isolation. Thus, it is important that designers use robust formal verification techniques to identify and rectify these structural and functional errors.

A major consideration is to ensure that each power-centric implementation technique is executed in a holistic manner, taking account of all of the other techniques being employed on a particular design. Each technique has to be supportive of the other techniques, and any power savings achieved from one technique needs to be maintained and preserved by the other techniques. For example, one of the most basic savings in power is to not use the biggest drivers. A slight reduction in margins will result in a smaller design occupying less area and consuming less power. But if the design environment does not support power-aware timing closure, it may negate any power savings achieved earlier in the process by adding too many cells for delay fixing.

Similarly, the practice of adding extra timing margin early in the process to ensure easy timing closure will negatively impact power consumption. Thus, once again, it is important that all optimization techniques simultaneously consider all of the implementation objectives to achieve the best possible solution.

Using the Common Power Format (CPF) is the key to effective power-aware implementation. The CPF captures the intent of chip architects and designers and enables automatic optimization during the implementation. And again, having this specification separate from RTL enables re-use of a block in different power profiles.

By employing automation tools that understand the designer's power intent through CPF for optimization, design planning, and layout steps, design teams can achieve superior timing and power tradeoff while improving the productivity and cycle time during the implementation phase.

POWER-AWARE VERIFICATION

Verifying today's extremely large and complex designs is both expensive and resource intensive. In fact, verification can account for as much as 70 percent of a digital IC's development time and resources.

Verification commences with the planning process. Once the top-level verification goals have been established, individual items to be tested are detailed, the way in which each item will be verified is defined, and the required coverage metrics for each item are specified.

In a modern design and verification environment, the human and machine-readable verification plan is complemented by a sophisticated and intelligent verification management application. This takes the executable verification plan and automatically deploys the appropriate tools required to perform the various facets of the verification. These tools include formal verification engines, software simulators, hardware accelerators, and emulators. The verification management application automatically balances the jobs for each verification session to make optimal use of all available resources.

Two key attributes of the verification management application are to bring visibility and predictability into the verification process. For example, the executable verification plan can specifically define certain verification metrics (such as coverage goals) that must be achieved by certain dates/milestones. These milestones are used to pace the team and to ensure that commitments to other groups and external customers are met.

The verification management application automatically accesses the log files from the various verification engines, parses them, and analyzes the results. It compares the required metrics with the actual results and reacts accordingly by re-deploying resources to address any problem areas. Furthermore, users (managers and engineers) can employ the verification management application to generate frequent, accurate, and concise reports in real time. At the push of a button, a manager can immediately see which portions of the design have been verified and which have not. If things are starting to slip, the team can quickly re-deploy effort to focus on problem areas.

In the past, verification plans of this type were based on various coverage metrics (assertion coverage, code coverage, functional coverage, etc.). In the case of a power-aware design and verification environment, the verification process also needs to take account of any power-centric aspects of the design. For example, before a certain block is powered down, it may be necessary to first power down one or more other blocks in a specific order, and to ensure that each block is completely powered down before a subsequent block is powered down. Similarly, when this portion of the design is restored to its operating condition, it will be necessary to ensure that the various blocks are powered up in a specific order.

To address this, the verification environment should automatically access the information contained within the Common Power Format (CPF) file(s) associated with the design. This is because the CPF captures information on the different power profiles and modes associated with the various blocks forming the design, how these profiles and modes interrelate, and how they are to be tested. The CPF can then be used to control tools such as simulators and/or formal analysis engines to verify functionality, such as power shut-off and state retention, thereby eliminating the need to develop custom RTL, testbench code, and PLI routines to exercise this behavior.

For example, it may be that certain modes are mutually exclusive, in which case the CPF will capture this information, the verification plan will document the appropriated tests, and the verification manager will deploy the appropriate verification engines to ensure such a condition can never occur. Similarly, in

the case of a power shutdown methodology employing state retention, the verification manager can call the appropriate verification engines to ensure that the state is maintained and the system resumes with the same state restored.

Finally, the power control module is a complex state machine that must be verified to ensure that power sequences and mode transitions occur correctly. Corner-case situations—which can have devastating consequences if not caught—are difficult to capture during dynamic simulation, so formal analysis is often used to verify the functionality of this module.

FUTURE CHALLENGES

With the adoption of advanced process nodes, design teams also need to start accounting for probabilistic on-chip variations (OCV) associated with the design processes. The process of manufacturing an IC is inherently imperfect. Slight variations in the duration, temperature, and chemical concentrations at each step result in variations from one wafer to another; between die on the same wafer (inter-die); and between cells and interconnect on the same die (intra-die). This requires the design team to consider analysis and optimization engines that are statistical in nature.

In the case of timing, the industry is moving to the use of statistical static timing analysis (SSTA) engines, in which the timing characteristics of each element in the device are represented as probability distribution functions (PDFs). One major advantage of this technique is that it is not necessary to excessively guard band the design (excessive guard bands are major obstacles for power-efficient design). This means the designers aren't leaving performance "on the table" and no longer have to sacrifice power for performance. In the case of low-power designs at the 65nm technology node and below, statistical techniques have to be extended to cover power and noise as well.

SUMMARY

Design teams across the semiconductor industry are adopting power management techniques to meet their market requirements. In addition to the usual considerations of chip performance, functionality, and cost, these new power-related requirements include improved battery life, reduced system cost, cooler operation and improved reliability.

Although advanced low-power techniques such as MSMV, PSO, and DVFS offer maximum power reduction, these techniques may have profound impact on the chip design, verification, and implementation methodologies. Often, the complexities associated with these advanced techniques result in low-power designs that suffer from sub-optimal timing, power, and area tradeoffs; reduced productivity and turn-around time; and increased risk of silicon failure.

To enable the adoption of advanced low-power techniques by mainstream users, there is a need for a design flow that holistically addresses the architecture, design, verification, and implementation of low-power designs. The Common Power Format (CPF) has emerged as an effective means to capture the design's power intent early on and to communicate this intent throughout the design flow. In addition to the CPF, chip design teams need to adopt innovative automation tools to ensure that the design intent specified through the CPF is preserved across the entire verification and implementation flow.



cādence™

Cadence Design Systems, Inc.

Corporate Headquarters

2655 Seely Avenue San Jose, CA 95134

800.746.6223 / 408.943.1234

www.cadence.com