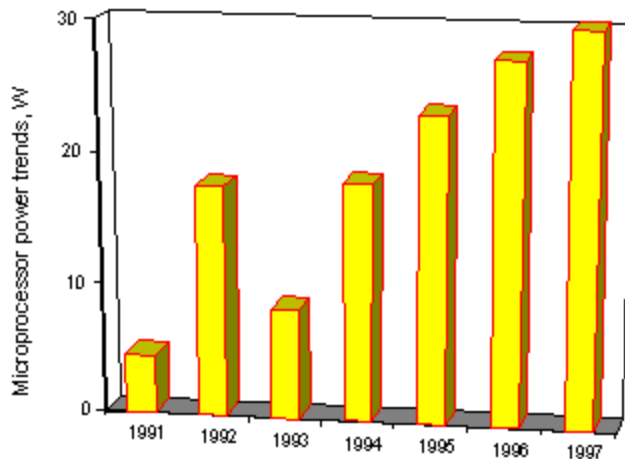


From IEEE Spectrum (Volume 35, Number 2, February 1998)

A multi-level approach to low-power IC design

by Jerry Frenkil Sequence Design, Inc., Santa Clara, CA

The demand for battery-powered products is arousing immense interest in energy-efficient devices. Meanwhile, integrated-circuit densities and operating speeds have continued to climb, proving Moore's law again and again. But chips cannot get more complex, faster, and larger without their hunger for power growing as well [Fig.1]. In fact, with pleas for portability adding to the usual clamor for more features and faster operation, power consumption has in many cases become the limiting factor in fulfilling market demand.



[Fig.1] Microprocessor power needs are climbing, despite lower supply voltages and low-power design techniques. This chart was compiled by surveying annual releases in the major architectures, from Alpha to x86, choosing the biggest power consumer from each, and averaging them. Data came from vendors and from information released at technical conferences.

The concept of power-efficient design is certainly not new. Low power design techniques have been employed for more than 25 years, particularly in the areas of watch circuits and calculators. What is new, however, is the requirement of both very high performance and low power at the same time. In some cases these requirements are motivated simply by the need to extend battery life, while in others they are driven by the need

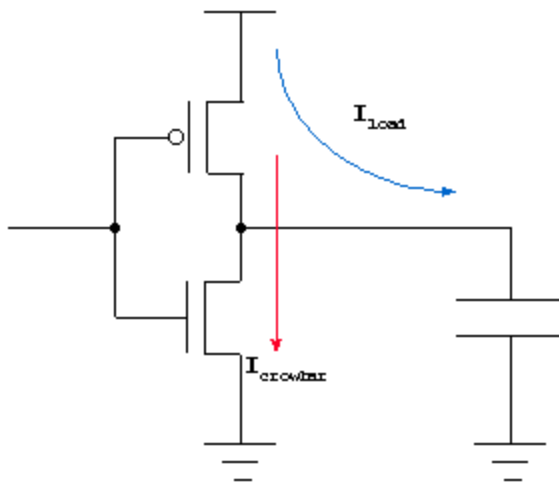
for thermal management.

The need has not gone unnoticed. Low-power IC design has become an especially vibrant area of research and development, resulting in advances in low-power fabrication processes and circuit techniques, dynamically programmable power supplies, and power efficient microprocessors. A new generation of computer-aided design tools developed especially for power efficient design is now available to help engineers optimize power at most stages of the circuit-development process.

Power primer

An IC consumes two types of power: static and dynamic power. The main difference between them is that dynamic power is frequency dependent, while static is not. Static power is defined as the product of the power supply voltage and a static, or dc, current, which itself has two components: leakage current and through-current. Leakage currents are parasitic effects common to all bulk MOS devices, and are slight enough to be ignored except in battery powered applications with long standby or sleep times. Through-currents are orders of magnitude larger, usually in the microampere to milliampere range; they occur in circuits designed with analog techniques or those that use resistive pull-up devices.

Dynamic power also has two components. The first arises from momentary short-circuit currents, known as crowbar currents, that flow from power to ground when a transistor stack switches state. While the transistor's inputs are changing from low to high or high to low, the p-channel and n-channel transistors are for an instant both "on" in their linear regions. The more dominant component of dynamic power is capacitive power--the product of the load capacitance, the square of the supply voltage, and the toggle frequency [Fig. 2].



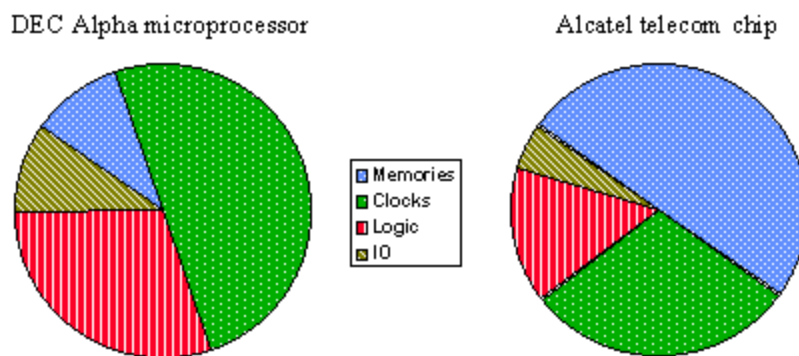
[Fig.2] Dynamic power, which accounts for most of the power dissipated in CMOS ICs, has two major components: crowbar current and load current. Crowbar power is dissipated when the transistor's inputs are changing from low to high or high to low because the p-channel and the n-channel devices are both in their linear regions at the same instant, creating a low-resistance path to ground.

Alternatively, from a cell-based viewpoint, dynamic power can be partitioned into whatever is consumed internally by the cell and whatever is consumed in driving the load. Cell power is the power used internally by a cell or module primitive, for example a NAND gate or flip-flop, including crowbar power and the power required to charge the capacitances inside the cell. Load power is used in charging the

external loads driven by the cell, including both wiring and fanout capacitances. So the dynamic power for an entire chip would be the sum of the power consumed by all the cells on the chip and the power consumed in driving all the load capacitances.

Low-power design techniques focus on reducing one or more of these components. In the case of dynamic power, reducing the supply voltage is usually the first choice since the power goes up with the square of the voltage. By lowering the supply voltage also lowers transistor output currents, lengthening signal delays and degrading performance. For this reason, most low-power design efforts focus on reducing both capacitance and signal frequencies, as well as shunning circuits that consume static power. This strategy is applied globally, to the entire chip, and locally, to individual modules, as different chips vary widely in how much power they

consume and in what circuits are the biggest power consumers [Fig. 3].



[Fig. 3] Power Consumption characteristics of ICs vary tremendously from design to design. The power consumed by the first Alpha micro-processor from Digital Equipment Corp. was dominated by the clocks, while an Alcatel telecommunications chip memories use the most.

Low-power VLSI design

A variety of techniques exist for keeping power consumption to a minimum. For any given design, the choice of which to employ is determined by several factors such as available supply voltages, preferred circuit design styles, and semiconductor fabrication technology.

The greatest impact on power is made by the supply voltage. Aside from such non-trivial issues as supply voltage standards and I/O signaling standards, the key concern is the performance degradation just mentioned.

One way to address this concern is to use several supply voltages: the highest voltage for the logic with the greatest need for speed, and lower voltages for those circuits with less demanding speed requirements.

Another method is to alter the fabrication process to support very low voltages. Jim Burr and John Shott at Stanford University, Stanford, California, demonstrated this approach with an ultralow-power CMOS technology having a supply voltage of only 200 mV. Their process featured device thresholds near 0 V, and used a substrate bias to adjust the thresholds to account for temperature and processing variations. Although the static leakage currents were raised substantially by the extremely low device thresholds, the drop in dynamic power more than offset the leakage power: the power-delay product of a seven-stage oscillator improved by a factor of 625 compared with the same circuit fabricated in 5-V CMOS.

Unfortunately, this is a very expensive solution for commercial applications, because the required process modifications are usually available only to semiconductor companies. Circuit design is more complicated as well, for lack of margins adequate to deal with such difficult issues as noise and soft errors.

Also in the processing domain, developers are working to minimize capacitances. Low-capacitance interconnects are desirable for high performance as well as for power reduction. Two are of particular interest: copper interconnects, and insulators with low dielectric constants. The first, recently commercialized by IBM Corp. and Motorola Inc., holds out the prospect of lower capacitances and higher resistance to electromigration. Added levels of interconnects also help, since the upper layers are less capacitive than the lower ones and so are the preferred routing layers layer for highly active signals such as clocks. Silicon-on-insulator

technologies seem worthy candidates for low-power applications because their junction capacitances are much lower than in standard bulk CMOS. But once more, cost is a factor. These solutions entail new processing techniques and in the case of silicon-on-insulator, a completely different type of substrate.

The sheer expense of modifying the process to cut back on power is an inducement to turn to design-oriented solutions. The three basic options are reducing cell power, cutting interconnect capacitance, and minimizing the average switching frequency.

Power can be lowered by carefully designing and laying out any given cell so as to shrink the transistors and parasitic capacitances within it to a minimum. On a case-by-case basis, logic families---static or domino CMOS for example---can also be chosen to economize on power. To limit interconnect capacitance, wires are made as short as possible and high-frequency signals are preferentially routed on the least capacitive layers.

Frequency reduction techniques focus on minimizing the energy wasted from operations and signal transitions that do no useful work. Examples are glitches--spurious or undesired signal transitions--and the clocking of stale data into registers.

To every level, its own techniques

Contemporary VLSI design occurs at many levels of abstraction, which can be classified, from lowest to highest, as transistor, logic, architecture, and system. The strategies for cutting back on power use differ because at each level because the design task is fundamentally different.

At the transistor level, capacitance reduction is most effective. Global floorplanning and logic partitioning are useful in minimizing wire lengths and hence wiring capacitances. Within cells, transistors are kept as small as possible so as to present the lowest capacitive loads possible. Signals with limited voltage swings serve to lessen the voltage component of dynamic power. (This last approach is usually confined to I/O buffers that drive large capacitive loads, because limited swing drivers and receivers usually require some circuit overhead that consumes static power.) The Gunning transceiver I/O buffer, with an output voltage swing of only 1 V, is an example of a circuit having a limited signal swing.

Note that these techniques--limited-swing drivers and reduced-capacitance transistors--are often also used to design high-speed circuits: signal switching is faster since the signals have to transition across a smaller voltage difference, and smaller capacitances result in faster signal rise times as well as lower power consumption.

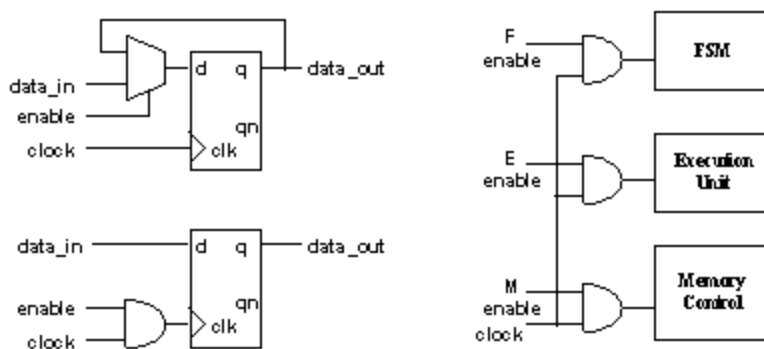
At the logic level, opportunities to economize on power exist in both the capacitance and frequency spaces, beginning with the choice of cell library. Standard cells have lower input capacitances than gate arrays because they use a variety of transistor sizes. For the same reason, the cells themselves consume less power when switching. Capacitance can also be reduced by using libraries designed for low power. These libraries contain cells that have low-power microarchitectures or operate at very low voltages. Some of the leading application-specific IC (ASIC) vendors are providing such libraries today, and many captive design groups are producing specialized libraries for low-power applications. But no matter which type of library is utilized, the logic designer can minimize the power used by each cell instance by paying careful attention to the transition times of input and output signals. Long rise and fall times should be avoided in order to minimize the crowbar current component of the cell power.

Methods to reduce activity at the logic level usually take the form of blocking unnecessary signal transitions--for example, by latching the inputs to a multiplier so that data enters the multiplier only when multiplication results are desired. Similarly, decoders can be designed with enables so that the inputs are decoded only when necessary, thus preventing unwanted and thereby wasted output transitions.

Several techniques can be applied to optimize the logic itself. Logic restructuring is used to prevent high switching frequencies from propagating through the logic when their values are unwanted, and to choose the most power-efficient local logic structures. Low-power state encoding is employed to choose the states that minimize the dynamic power of finite-state machines.

Several power minimization techniques work especially well at the architectural level. Most of them rely on switching frequency. The best example of which is the use of clock gating. In clock gating, a control signal enables a clock signal so that the clock toggles only when the enable signal is true, and is held steady when the enable signal is false. Gated clocks are used, in power management, to shut down portions of the chip, large and small, that are inactive. This saves on clock power, because the local clock line is not toggling all the time.

Consider the case of a data bus input register [Fig. 4]. With conventional scheme, the register is clocked all the time, whether new data is to be captured or not. If the register must hold the old state, its output is fed back into the data input through a multiplexer whose enable line controls whether the register clocks in new data or recycles the existing data. With a gated clock, the signal that would otherwise control the select line on the multiplexer now controls the gate. The result is that the power consumed in driving the register's clock input is reduced in proportion to the decrease in average local clock frequency. The two circuits function identically, but utilization of the gated clock reduces the power consumption.



[Fig. 4] Clock gating is highly effective at reducing dynamic power, because it prevents the unnecessary clocking of storage elements. In conventional schemes [top left], the register is clocked all the time, whether or not new data awaits. If the register must hold old data, its output is fed back into the data input through a multiplexer whose enable line controls whether the register clocks in new data or recycles existing data. With a gated clock [bottom left], the signal that had controlled the enable line now controls the gate. The lower the average local clock frequency, the less power is spent driving the register's clock input. Thus, the two circuits function identically, but use of the gated clock saves power. Clocks can be gated locally [bottom left] or else at a more global level [right] for even greater power reductions.

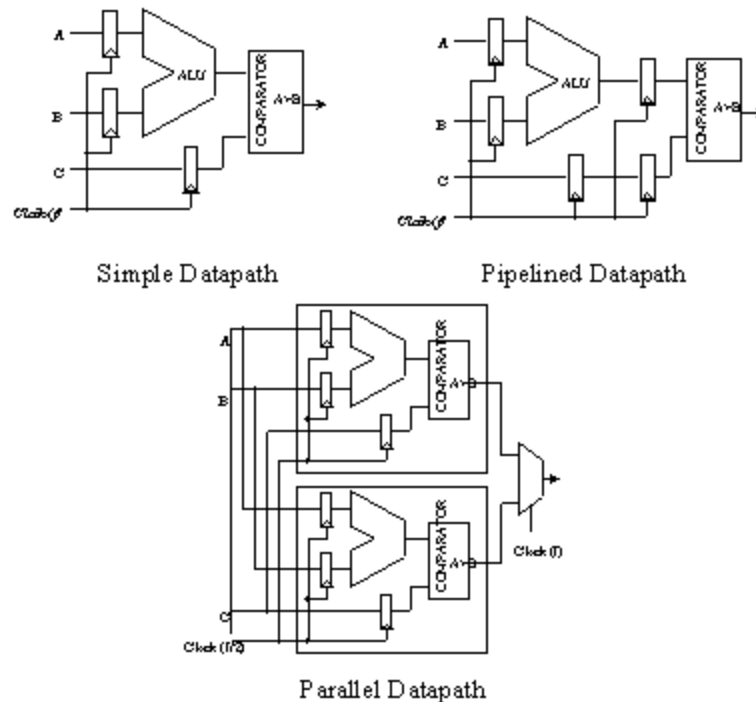
The gated clock scheme also reduces cell power within each register. Many register, or flip-flop, designs utilize local clock buffering where the clock signal is inverted and buffered to produce an inverted and buffered true clock signal internally. Thus when the input clock switches, the internal

buffers switch, whether or not the flip-flop's output changes state.

Clock gating can be implemented locally by gating the clocks to individual registers, or globally, by building the gating structures into the overall architecture to turn off large functional modules. While both

techniques are effective at reducing power, global gating results in much larger power reductions and is often used in implementing power-down and power-management modes.

A more aggressive architectural approach, known as Architecture Based Voltage Scaling, was developed at the University of California at Berkeley by Anantha Chandrakasan and Robert Brodersen. The idea is to lower the supply voltage, V_{dd} , and accept lower transistor performance, but compensate for it with a higher-throughput architecture. A simple datapath might, for example, perform an addition and a comparison [Fig. 5]. Parallelism is used to compensate for the loss in throughput due to voltage reduction.



[Fig. 5] When voltages are lowered, the performance of an IC can be maintained if the slower transistor switching is offset by parallelism and pipelining. (Granted, this remedy increases IC area.) The simple datapath [top left] performs an addition and a comparison. Pipelining [top right] and parallelism [bottom] compensate for the loss in throughput due to voltage reduction. The use of parallelism and pipelining to maintain performance at lower voltages was developed by Anantha Chandrakasan and Robert Brodersen at the University of California at Berkeley.

Performance can be further boosted by adding pipelining. *Table 1* summarizes the results of this approach. Its obvious drawback is the increase in chip area, but the resulting power reduction is significant. In this example, all four architectures achieve the identical throughput. However, the parallel, pipelined, and pipelined-parallel versions consume less power than the simple datapath since they each require a lower supply voltage to reach the same throughput as the simple architecture.

Table 1: Effects of architecture-based voltage scaling			
Architecture	Voltage,V	Area (normalized)	Power (normalized)
Simple	5.0 V	1.0	1.00
Parallel	2.9 V	3.4	0.36
Pipelined	2.9 V	1.3	0.39
Pipelined-Parallel	2.0 V	3.7	0.20

Attacking frequency reduction at the system level can also result in large power savings, especially for processors in portable systems. The PowerPC603, for example, contains three power management modes--doze, nap, and sleep--that are controlled by the operating system and cut power use overall when the processor is idle for any extended period of time. With these modes, chip power can go from 2.2 W in active mode to 358 mW in doze, 126 mW in nap, and as low as 1.8 mW in sleep.

A thriftier use of power by way of improved application software also augurs well. Such optimizations address the effect on a processor's power consumption of specific instruction sequences. Recent work at Princeton University, New Jersey, by Vivek Tiwari, Sharad Malik, and their colleagues has shown that with this the approach, a particular Fujitsu digital signal processor could run on 26 percent to 73 percent less power--with no hardware changes. They did it by, among other things, carefully assigning data to specific memory banks and by using packed, instead of unpacked instructions. In the former technique, if operands will be needed together for computations, they are assigned to the same memory bank to take advantage of a double-operand move operation out of the one memory. Since the double-operand move takes only a single cycle, instead of two cycles for two single-operand moves, the access draws less power. For the same reason, using packed, instead of unpacked, instructions also consumes less power: instructions are chosen that reduce the number of execution cycles and so are fundamentally more efficient.

An example of a design that used many of these techniques is the StrongARM 110 microprocessor, designed by Digital Equipment Corp. and Advanced RISC Machines, Ltd. Introduced in 1996, this processor produces 183 Dhrystone MIPS at 160 MHz, while operating off of a 1.5 volt supply and consuming only 500 mW.

Tools for low power design

As with all other aspects of VLSI design, electronic design automation (EDA) tools are essential for designing, analyzing, and optimizing integrated circuits for power efficiency. In fact, many sophisticated tools are already in existence. *Table 2* lists most of the leading tools for low-power IC design.

2. Tools for low-power IC design		
Tool Name	Vendor	Key attributes
Abstraction: architecture		
WattWatcher / Architect	Sente, Inc.	Register transfer level analysis
Abstraction: gate		
WattWatcher / Gate	Sente	Full chip gate-level analysis
POET	Viewlogic, Inc.	Cycle-by-cycle gate-level analysis
Power Compiler	Synopsys, Inc.	Gate-level power optimization
Design Power	Synopsys	Gate-level probabilistic analysis
QuickPower	Mentor Graphics Corp.	Cycle-by-cycle gate level analysis
Power Gate	Epic Design Automation	Cycle-by-cycle gate level analysis
Abstraction: transistor		
PowerMill	Epic	Transistor-level analysis
LSimPower	Mentor	Transistor-level analysis
AMPS	Epic	Transistor-level optimization
RailMill	Epic	Transistor-level electromigration and IR voltage drop analysis
PowerArc	Epic	Circuit characterization
Thunder & Lightning	Simplex Solutions Inc.	Transistor-level electromigration and IR voltage drop analysis
Star-Sim	Avant! Corp.	High-capacity SPICE compatible transistor-level analysis
Star-HSPICE	Avant!	Circuit-characterization
Star-Power	Avant!	Transistor-level electromigration and IR voltage drop analysis

The tools can be differentiated, to a first order, by the level of abstraction on which they operate. Lowest of all are the transistor-level tools. These tools possess the best accuracy, but commensurately require the longest run times and have the smallest capacities--the size of the circuit that can be analyzed. While transistor-level analysis tools can assist with analyses earlier in the design process, they are typically used to characterize cells and modules for use at the higher abstraction levels. In addition, they play a critical role in analog and mixed-signal analysis. Transistor-level power optimization is also available for minimizing power by automatic transistor resizing, so as to trade off delay and size versus power along a logical path.

The next level of abstraction embraces the logic-level power analysis tools. Their function is typically to verify that the power consumption of the gate level design is still within the target specification. They are also good for optimizing certain design parameters, such as signal transition times, which are unknown at the higher levels. Tools at this level tend to be about an order of magnitude faster than the transistor-level tools and can handle much larger circuits. But they are less accurate. While the transistor level tools produce results that are within 5-10 percent of measured results, the gate-level tools are generally accurate to about 10-15 percent.

The highest abstraction level for which power analysis tools exist today is the architectural level. This type of tool analyzes abstract design representations such as Verilog or VHDL RTL code. Because less detailed information is available about the design at this stage, accuracy is not as great as at the lower levels, but is typically within 20 to 25 percent of silicon. However, for the same reason, far larger designs can be analyzed in far less time, making the tool appropriate for use in evaluating architectural design tradeoffs. Optimization at this level is currently an interactive process, consisting of the analysis of various architectures and organizations and the subsequent choice of the least power alternative that satisfies the other project constraints.

A low power design methodology

Given the complexity of VLSI circuits along with the subtle factors that affect many design decisions, tools alone are usually insufficient--a coherent design methodology is required as well. A methodology for low power design has several components: up-front specification, early analysis and optimization, and multi-level verification. A suite of tools to support this methodology would include power analysis tools for each of the architecture, logic, and transistor levels. This software should have extensive reporting mechanisms so that the designer can easily understand the power consumed in each portion of the design. This methodology is possible because by the existance of power models for such library primitives as gates and flip-flops, as well as more complex functions such as I/Os, phase-locked loops, memories, and controllers.

The design methodology begins with an understanding of the power consumption target. As in designing for performance, where early architectural decisions influence the resulting IC throughput, these same decisions play a major role in achieving the power specification. While early analysis and optimization are perhaps the most important part of the methodology, they are by no means the only parts.

In a low-power design methodology, power analysis and verification are pursued at each of the architecture, logic, and transistor levels through the use of analysis tools appropriate to each level. In all phases of the project, various design options are analyzed with a view to picking the

lowest-power alternative that fits within the other constraints (cost, size, performance, and time to market). Power consumption is thus tracked throughout as the design becomes ever more detailed and ever less abstract. At the beginning of the project, the biggest issue is the design of an RTL architecture that can meet the power specification targeted. Architecture-level analysis tools are employed to evaluate different architectures and design options in terms of their effect on power use. Later in the project, after synthesis and the design's representation by gates or transistors, the power issue swings more to verifying that the early design target has been met. At this point, gate- or transistor-level tools are used to estimate the power, and while run times at this level often take days or even weeks, these lower level verifications are

run much less often than the earlier architectural estimates.

In summary, the methodology applies the appropriate tools to aid in analyzing and optimizing the basic architectural design early in the project. Once the architecture has been refined to the point where it can be shown to meet the power specification, it is fed forward to the logic level. At this point it is converted into a logical netlist either through conventional manual translation or through automatic logic synthesis. At this level the design is again analyzed and optimized, using techniques and transformations appropriate for the level. Once again, when the design has been shown to meet the target specification it is fed forward to the physical level, where it is analyzed, and if necessary, optimized.

Future trends

As the issue of power efficiency becomes even more pervasive, the battle to use the bare minimum of power will be fought on multiple fronts: semiconductor technology, circuit design, system architecture, and design automation tools.

As it happens, many semiconductor companies are already developing manufacturing processes with improved energy-delay characteristics and lower power supply voltages, device thresholds, and interconnect capacitance, among other things.

Circuit design research is investigating lower-power structures, such as true-single-phase clocked (TSPC) flip-flops and latches. These storage elements have no need for local clock buffering, whose removal would reduce total clock power. Limited-swing circuits should also cut power, too, as well as raise speeds. Researchers are also studying techniques to return energy to the supply instead of being dissipating it as heat. While these adiabatic logic techniques have been shown to improve power efficiency at low frequencies, much more work is needed to make them useful for a wide range of applications.

Design tool development will rise to higher levels of abstraction thanks to the dramatic power savings possible through architectural optimizations. The next development is likely to be architecture-level power optimizers in which RTL code is restructured to require less power, followed by analytical tools for higher-abstraction behavioral descriptions.

Expansion of the low-power design "culture" will occur as well, when increasing numbers of designers, project leaders, and managers awake to the many rewards of power-efficient design. Eventually, the concern for low-power design will expand from devices to modules to entire systems, including software. Application software will in some cases be crafted specifically for power efficiency.

While today low-power design is perhaps in its infancy relative to design for performance, it will soon mature into a first-order design issue for most IC developments. Driven by seemingly inexorable advances of circuit miniaturization and burgeoning circuit complexities, design tools and methodologies will be required to manage the task. A low-power future is dawning.

To probe further

Low Power Digital CMOS Design, by Anantha Chandrakasan and Robert Brodersen (Kluwer Academic Press, Norwell, Mass. 1996), presents a comprehensive approach to system-and

circuit-level power minimization. The vehicle for the presentation is the Infopad project at the University of California at Berkeley. Infopad is a wireless multimedia terminal designed for extremely low-power consumption.

"Power Conscious CAD Tools and Methodologies: A Perspective" is a comprehensive overview of the design automation tools and methodologies for low power design, presented in the form of a tutorial. Written by Deo Singh, Jan Rabaey, Massoud Pedram, Francky Catthoor, Suresh Rajgopal, Naresh Sehgal, and Thomas Mozdzen, it appeared in the Proceedings of the IEEE, April 1995, Vol. 83, no. 4, pp. 570-94

Low Power Design Methodologies, edited by Jan Rabaey and Massoud Pedram, deals in depth with design methodologies for all layers of the design abstraction. It was published in 1995 by Kluwer Academic Press, Norwell, Mass.

For a review of the various proposed power estimation techniques for VLSI, read "A Survey of Power Estimation Techniques in VLSI Circuits" by Farid Najm, IEEE Transactions on VLSI Systems, December 1994, Vol. 2, no. 4, pp. 446-55.

"Power Analysis and Minimization Techniques for Embedded DSP Software", a paper by Mike Tien-Chien Lee, Vivek Tiwari, Sharad Malik, and Masahiro Fujita, presents a systematic approach to analyzing and minimizing software's power-consumption characteristics. It appeared in the IEEE Transactions on VLSI Systems, March 1997, Vol. 5, no. 1, pp. 123-133.

ABOUT THE AUTHOR

Jerry Frenkil (M) is the vice president for Low Power Design at Sente Inc., Acton, Mass. Prior to co-founding Sente, he worked in various capacities for system and semiconductor companies, spending eight years at VLSI Technology Inc., where he managed circuit design. He has several patents and publications.

Further Information from the Author included in this article:

Cutting back on power in a digital signal processor

Minimizing power consumption in an IC is a process of design, analysis, and modification, repeated again and again at every level of abstraction until the power specification is met. The procedure begins at the highest levels of abstraction, where the greatest gains can be made, then moves down to the next lower level.

This is known as feed-forward design: a design does not progress to the lower abstractions through synthesis or subsequent layout until the architecture satisfies the power specification. Early in the process, the object of power analysis is to support design, while later on it becomes verification.

The analysis starts out with two objectives: to estimate the overall power consumption, and then to understand how much power is consumed in different portions of the design--in other words, to identify the "hot" spots and the reasons for the power consumption. Then, the design is modified to address the root causes of the power usage. Addressing the largest consumers of power in his manner helps to ensure that efforts to cut back are focused on the areas that can produce the largest savings.

Consider the design of an application-specific digital signal processor (DSP) slated for use in a wireless telephone.

The first step could be the choice of supply voltage, low as possible within the constraints of battery availability and the voltage required by other chips in the system.

The next step would be to simulate the inner loops of the DSP algorithm that will be the most frequently executed using an architectural level estimator such as Sente Inc.'s Watt Watcher/Architect. This exercise yields a good estimate of the overall power characteristics of the design, and illuminates the portions that warrant the most attention. Likely hot spots are memories and datapaths.

To reduce power in memories, it is necessary to use a low power memory architecture. The memory can be split into two or more banks to facilitate parallel access. Additionally, a signal-processing algorithm that keeps the number of memory accesses to a minimum can be selected.

The datapath should be analyzed and designed to minimize glitching. One way to do this is path balancing, which helps to ensure that all of the signals arrive at a given operator at the same time, eliminating unwanted transitions. Another approach is to reorder functions so that glitchy signals are fed into the datapath as deep as possible in the logic. Architectural analysis should be repeated for each major mode of operation, so global clock gating can be used to shut down large functional blocks when they are idle.

Proceeding in parallel with, or sometimes ahead of, the architecture development is the design of the library macro functions and the datapath cells. Each of these elements should be designed at the transistor level for minimum power and pin capacitance.

The power and timing characteristics of flip-flop, adder, and multiplier microarchitectures should be given special attention. These cells are analyzed and characterized with transistor-level tools, such as Spice for the logic-level primitives and, for the more complex cells, PowerMill, from the Epic Technology Group of Synopsys Inc., in order to generate the power models needed during the architectural analysis. At the transistor level, AMPS, also from Epic, is often used to optimize a macro's power by shrinking transistor sizes wherever slack timing constraints allow.

Once the architecture has been frozen and shown to meet the power specification, the design can be synthesized or converted into a netlist consisting of lower-level primitives. Several logic optimizations can be automatically performed at this level with the aid of a logic-level optimizer such as Synopsys' Power Compiler. Technology mapping, logic restructuring, and signal transition time balancing are some of the possibilities. Post-synthesis and post-optimization power analysis can be performed on the gate level netlist by means of a gate level power estimator such as Mentor Graphics Corp.'s QuickPower.

During full-chip layout, the floorplan is configured to minimize the wiring capacitances on highly active signals. Perhaps the best examples are the clock signals, which can be routed on the least capacitive metal layers. Balanced buffer trees, which reduce wiring capacitance by minimizing wire widths, may be used to distribute the clock.

Sequence acknowledges trademarks or registered trademarks of other organizations for their respective products and services. Copyright © 1998 All rights reserved.