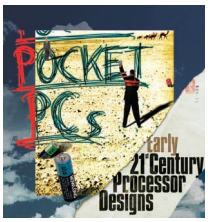

Power: A First-Class Architectural Design Constraint

Power is a design constraint not only for portable computers and mobile communication devices but also for high-end systems, and the design process should not subordinate it to performance.

Trevor Mudge
University of Michigan

Limiting power consumption presents a critical issue in computing, particularly in portable and mobile platforms such as laptop computers and cell phones. Limiting power in other computer settings—such as server farms—warehouse-sized buildings filled with Internet service providers' servers—is also important. A recent analysis by Deo Singh and Vivek Tawair of Intel showed that a 25,000-square-foot server farm with approximately 8,000 servers consumes 2 megawatts and that power consumption either directly or indirectly accounts for 25 percent of the cost for managing such a facility (see the proceedings of the Cool Chips Tutorial at http://www.eecs.umich.edu/~tnm/cool.pdf).

Internet use is growing exponentially, requiring server farms to match the accompanying demand for power. A *Financial Times* article in Fall 2000 noted that information technology (IT) consumes about 8 percent of power in the US. If this component continues to grow exponentially without check, IT will soon require more power than all other uses combined.

Table 1 presents information to better understand current processor power consumption trends (see the Berkeley CPU information center at http://bwrc.eecs.berkeley.edu/CIC/summary/local/summary.pdf). The rapid growth in power consumption is obvious. Equally alarming is the growth in the chip die's power density, which increases linearly. Alpha model 21364's power density has reached approximately 30 watts per square centimeter, which is three times that of a typical hot plate. This growth occurred despite process and circuit improvements. Trading high power for high performance cannot continue, and containing the growth in power requires adding architectural improvements to process and circuit improvements. The only exceptions will be one-of-a-kind supercomputers built for special tasks like weather modeling.

POWER EQUATIONS FOR CMOS LOGIC

Three equations provide a model of the power-performance trade-offs for complementary metal-oxide semiconductor logic circuits. The CMOS literature on low power frequently uses these equations. They are simplifications that capture the essentials for logic designers, architects, and systems builders. Here I focus on CMOS because it will likely remain the dominant technology for the next five to seven years. Besides applying directly to processor logic and caches, the equations have relevance for some aspects of DRAM chips. The first equation defines power consumption:

$$P = ACV^2 f + \tau AVI_{short} f + VI_{leak} \quad (1)$$

This equation has three components. The first component measures the dynamic power consumption caused by the charging and discharging of the capacitive load on each gate's output. It is proportional to the frequency of the system's operation, f, the activity of the gates in the system, A (some gates do not switch every clock), the total capacitance seen by the gate's outputs, C, and the square of the supply voltage, V. The second component captures the power expended as a result of the short-circuit current, I_{short}, which momentarily, τ, flows between the supply voltage and

0018-9162/01/$10.00 © 2001 IEEE

Table 1. Compaq Alpha power trends.

Alpha model	Peak power (W)	Frequency (MHz)	Die size (mm^2)	Supply voltage
21064	30	200	234	3.3
21164	50	300	299	3.3
21264	72	667	302	2.0
21364	100	1,000	350	1.5

ground when a CMOS logic gate's output switches. The third component measures the power lost from the leakage current regardless of the gate's state.

In today's circuits, the first term dominates, which immediately suggests that reducing the supply voltage, V, is the most effective way to reduce power consumption. The quadratic dependence on V means that the savings can be significant: Halving the voltage reduces the power consumption to one-fourth its original value. Unfortunately, this savings comes at the expense of performance, or, more accurately, maximum-operating frequency, as the next equation shows:

$$f_{max} \propto (V - V_{threshold})^2 / V \quad (2)$$

The maximum frequency of operation is roughly linear in V. Reducing it limits the circuit to a lower frequency. Reducing the power to one-fourth its original value only halves the maximum frequency. Equations 1 and 2 have an important corollary: Parallel processing, which involves splitting a computation in two and running it as two parallel independent tasks, has the potential to cut the power in half without slowing the computation.

Reducing the voltage, V, in Equation 2 requires a reduction in $V_{threshold}$. This reduction must occur so that low-voltage logic circuits can properly operate. Unfortunately, reducing $V_{threshold}$ increases the leakage current, as Equation 3 shows:

$$I_{leak} \propto \exp(-qV_{threshold} / (kT)) \quad (3)$$

Thus, this is a limited option for countering the effect of reducing V, because it makes the leakage term in the first equation appreciable.

We can summarize the three essential points from this model as follows:

- Reducing voltage has a significant effect on power consumption: $P \propto V^2$.
- Reducing activity—most simply by turning off the computer's unused parts—also affects power consumption.
- Parallel processing can reduce power consumption if done efficiently—ideally by applying it to independent tasks.

Although reducing the supply voltage can decrease power consumption and result in considerable savings with only a modest impact on performance, the accompanying increase in leakage current limits how far this technique can be taken.

Other figures of merit related to power

Equation 1 represents an average value that gives a somewhat one-dimensional view. In many cases, at least two other power quantities are important:

- *Peak power.* Typically, systems have an upper limit that, if exceeded, leads to some damage.
- *Dynamic power.* Sharp changes in power consumption can result in ground bounce or *di/dt noise* that upsets logic voltage levels, causing erroneous circuit behavior.

Often, we use the term *power* to refer to quantities that are not really power. For example, in portable devices, the amount of energy needed to perform a computation may be a more useful measure because a battery stores a fixed amount of energy, not power. To say that one processor is lower power than another may be misleading. If it takes longer to perform a given computation, the total energy expended may be the same—the battery will run down by the same amount in both cases. This comparison leads to the idea of an energy/operation ratio. A processor with a low energy/operation ratio is sometimes incorrectly referred to as a low-power processor. Engineers frequently use the inverse of this measure, MIPS/W or million instructions per second per watt, as a figure of merit for processors intended for mobile applications.

Reducing the power consumption by half decreases the frequency of operation by much less because of the quadratic term in Equation 1. Ricardo Gonzalez and Mark Horowitz[1] proposed a third figure of merit: energy × delay. This measure takes into account that, in systems in which power is modeled by Equation 1, we can trade a decrease in speed for higher MIPS/W. Most of the literature uses MIPS/W or simply watts, so we will continue this convention, recognizing that occasionally it can suggest misleading trade-offs where "quadratic" devices like CMOS are concerned. Finally, if the computation under consideration must finish by

Computer architecture research has pursued two themes: exploiting parallelism and using speculation.

a deadline, slowing the operation may not be an option. In these cases, a measure that combines total energy with a deadline is more appropriate.

REDUCING POWER CONSUMPTION

Systems designers have developed several techniques to save power at the logic, architecture, and operating systems levels.

Logic

A number of techniques can save power at the logic level. The clock tree can consume 30 percent of a processor's power; the early Alpha model 21064 exceeded even this figure. Therefore, it is not surprising that engineers have developed several power-saving techniques at this level.

Clock gating. This technique is widely used to turn off clock tree branches to latches or flip-flops whenever they are not used. Until recently, developers considered gated clocks to be a poor design practice because the clock tree gates can exacerbate clock skew. However, more accurate timing analyzers and more flexible design tools have made it possible for developers to produce reliable designs with gated clocks.

Half-frequency and half-swing clocks. A half-frequency clock uses both edges of the clock to synchronize events. The clock then runs at half the frequency of a conventional clock. The drawbacks are that the latches are more complex and occupy more area, and that the clock's requirements are more stringent.

The half-swing clock swings only half of V. It increases the latch design's requirements and is difficult to use in systems with low V. However, lowering clock swing usually produces greater gains than clocking on both edges.

Asynchronous logic. Asynchronous logic proponents have pointed out that because their systems do not have a clock, they save the considerable power that a clock tree requires. However, asynchronous logic design suffers from the drawback of needing to generate completion signals. This requirement means that additional logic must be used at each register transfer—in some cases, a double-rail implementation, which can increase the amount of logic and wiring. Other drawbacks include testing difficulty and an absence of design tools.

Several projects have attempted to demonstrate the power savings possible with asynchronous systems. The Amulet, an asynchronous implementation of the ARM instruction-set architecture, is one of the most successful projects (see the Power-Driven Microarchitecture Workshop at http://www.cs.colorado.edu/~grunwald/LowPowerWorkshop/agenda.html). Drawing definitive conclusions from such projects is difficult because it requires comparing designs that use the same technologies. Further, the asynchronous designer works at a disadvantage because today's design tools are geared for synchronous design. Ultimately, asynchronous design does not offer sufficient advantages to merit a wholesale switch from synchronous designs.

However, asynchronous techniques can play an important role in globally asynchronous, locally synchronous systems.[2] Such systems reduce clock power and help with the growing problem of clock skew across a large chip, while allowing the use of conventional design techniques for most of the chip.

Architecture

Computer architecture research, typified by the work presented at the International Symposia on Computer Architecture and the International Symposia on Microarchitecture, focuses on high performance. This research pursues two important themes: exploiting parallelism and using speculation. Parallelism can reduce power, whereas speculation can permit computations to proceed beyond dependent instructions that may not have completed. If the speculation is wrong, executing useless instructions can waste energy. But this is not necessarily so. Branch prediction is perhaps the best-known example of speculation. If the predictors are accurate, it can increase the MIPS/W figure.[3]

New architectural ideas can contribute most profitably to reducing the dynamic power consumption term, specifically the activity factor, A, in Equation 1.

Memory systems. The memory system consumes a significant amount of power. In systems with unsophisticated processors, cache memory can dominate the chip area. Memory systems have two sources of power loss. First, the frequency of memory access causes dynamic power loss, as the first term in Equation 1 models. Second, leakage current contributes to power loss, as the third term in Equation 1 models.

Organizing memory so that an access activates only parts of it can help to limit dynamic memory power loss. Placing a small cache in front of the L1 cache creates a filter cache that intercepts signals intended for the main cache.[4] Even if the filter cache is hit only 50 percent of the time, the power saved is half the difference between activating the main cache and the filter cache, which can be significant.

Memory banking, currently used in some low-power designs, splits the memory into banks and activates only the bank presently in use. Because it relies on the reference pattern having a lot of spatial locality, memory banking is more suitable for instruction-cache organization.

The architect or systems designer can do little to limit leakage except shut down the memory. This is only practical if the memory will remain unused for a long time because memory will lose state and, therefore, requires disk backup. The operating system usually

Microarchitecture Uses a Low-Power Core

Mike Morrow, Intel

The Intel XScale Core (http://developer.intel.com/design/intelxscale/) uses advanced circuit, process, and microarchitectural techniques to facilitate low-power operation.

A processor's active power dissipation is proportional to the frequency and square of the voltage. In the V^2 term of the active power equation, voltage to the core is scalable from 0.75 V to 1.65 V. The core also incorporates back-bias circuitry to minimize leakage current during inactivity, and the core's static design—the clock can be reduced to dc—contributes linearly to active power savings.

Intel's 80200 (http://developer.intel.com/design/iio/manuals/273411.htm)—the first processor to incorporate Intel XScale microarchitecture—offers additional power-saving opportunities by enabling rapid frequency and no-stall voltage changes and by providing a memory bus that runs asynchronously.

Reaching a lower-power state may not be worthwhile if it requires a long time. Because the 80200 uses a PLL with a fast resynch time, it can switch frequencies or power modes in less than 20 µs. This enables system techniques in which the processor, while providing good interactive response, is usually asleep or operating at low frequency. For example, the processor can be in "drowsy" mode (core dissipation less than 1 mW) until an event awakens it; then it processes the event at several hundred megahertz and drops back into drowsy mode. Even the fastest touch-typist would not detect any performance degradation.

For voltage scaling to be effective, the processor must not spend excessive time waiting for the voltage to change. A shorter processor idle period translates into accomplishing more work. The 80200's core can run through a voltage change without the processor idling during the transition. Also, the voltage can follow a natural discharge curve rather than being stepped or precipitously dropped for the core, which avoids wastefully forcing the voltage to a lower level and perhaps enables the use of a simpler power supply.

The 80200's bus interface runs asynchronously—the core can run at a frequency unrelated to the bus clock. In fact, two clock signals are input to the device, allowing system software to change core frequency as needed without concern for bus operation. If the bus frequency is a fixed fraction of the core frequency, as is often the case, a change in core frequency might necessitate waiting for the bus to clear.

The asynchronous relationship of the bus and core clocks also enables static or dynamic optimization of the bus frequency. A RAM region with a 25-ns access time requires three 100-MHz clock cycles per access, or one 33-MHz clock cycle. A system with predictable accesses to this memory might choose to drop the bus frequency and realize a linear improvement in power without incurring any loss in throughput.

***Mike Morrow** is a processor architect at Intel. Contact him at michael.morrow@intel.com.*

handles this type of shutdown, which we often refer to as "sleep mode."

Buses. Buses are a significant source of power loss, especially interchip buses, which are often very wide. The standard PC memory bus includes 64 data lines and 32 address lines, and each line requires substantial drivers. A chip can expend 15 percent to 20 percent of its power on these interchip drivers.

One approach to limiting this swing is to encode the address lines into a Gray code because address changes, particularly from cache refills, are often sequential, and counting in Gray code switches the least number of signals.[5]

Adapting other ideas to this problem is straightforward. Transmitting the difference between successive address values achieves a result similar to the Gray code. Compressing the information in address lines further reduces them.[6] These techniques are best suited to interchip signaling because designers can integrate the encoding into the bus controllers.

Code compression results in significant instruction-memory savings if the system stores the program in compressed form and decompresses it on the fly, typically on a cache miss.[7] Reducing memory size translates to power savings. It also reduces code overlays—a technique still used in many digital-signal processing (DSP) systems—which are another source of power loss.

Parallel processing and pipelining. A corollary of our power model is that parallel processing can be an important technique for reducing power consumption in CMOS systems. Pipelining does not share this advantage because it achieves concurrency by increasing the clock frequency, which limits the ability to scale the voltage, as Equation 2 demonstrates. This is an interesting reversal because pipelining is simpler than parallel processing; therefore, pipelining is traditionally the first choice to speed up execution. In practice, the trade-off between pipelining and parallelism is not so distinct: Replicating function units rather than pipelining them has the negative effect of increasing area and wiring, which in turn can increase power consumption.

The degree to which designers can parallelize computations varies widely. Although some computations are "embarrassingly parallel," they are usually characterized by identical operations on array data structures.

Despite their lower clock rates, DSPs can achieve high MIPS ratings because of their parallelism and direct support for a multiply-accumulate operation.

However, for general-purpose computations typified by the System Performance Evaluation Cooperative benchmark suite, designers have made little progress in discovering parallelism. Successful general-purpose microprocessors rarely issue more than three or four instructions at once. Increasing instruction-level parallelism is unlikely to offset the loss caused by inefficient parallel execution. Future desktop architectures, however, will likely have shorter pipes.

In contrast, common signal-processing algorithms often possess a significant degree of parallelism. DSP chip architecture, which is notably different from desktop or workstation architectures, reflects this difference. DSPs typically run at lower frequencies and exploit a higher degree of parallelism. Despite their lower clock rates, DSPs can achieve high MIPS ratings because of their parallelism and direct support for a multiply-accumulate operation, which occurs with considerable frequency in signal-processing algorithms. An example is the Analog Device 21160 SHARC DSP, which uses only 2 watts to achieve 600 Mflops on some DSP kernels.

Operating system

The quadratic voltage term in Equation 1 shows that reducing voltage offers a significant power-savings benefit. It is not necessary for a processor to run constantly at maximum frequency to accomplish its work. If we know a computation's deadline, we can adjust the processor's frequency and reduce the supply voltage. For example, a simple MPEG decode runs at a fixed rate determined by the screen refresh, usually once every 1/30th of a second. Therefore, we can adjust the processor to run so that it does not finish its work ahead of schedule and waste power.

There are two approaches to scaling voltage using the operating system. The first approach provides an interface that allows the operating system to directly set the voltage—often simply by writing a register. The application uses these operating system functions to schedule its voltage needs.[8] The second approach uses a similar interface, but the operating system automatically detects when to scale back the voltage during the application. An advantage of automatic detection is that applications do not require modifications to perform voltage scheduling. A disadvantage is that detecting the best timing for scaling back voltage is difficult, but important steps toward solving this problem are under way (see Kris Flautner's thesis at http://www.eecs.umich.edu/~tnm/theses/krisf.pdf).

Both approaches offer effective power-saving methods that work directly on the quadratic term in Equation 1. Voltage scaling has already received support in Intel's next-generation StrongARM microprocessor, the XScale.

WHAT CAN WE DO WITH A HIGH MIPS/W DEVICE?

The obvious applications for processors with a high MIPS/W lie in mobile computing. Futuristic third-generation mobile phones will require remarkable processing capabilities. These 3G phones will communicate over a packet-switched wireless link at up to 2 Mbits/sec. The link will support both voice and data and will be connected all the time. Vendors plan to support MPEG4 video transmission and other data-intensive applications.

In contrast, manufacturers build today's cell phones around two processors: a general-purpose computer and a DSP engine. Both processors require low power, the lower the better. A common solution uses an ARM processor for the general-purpose machine and a Texas Instruments DSP chip. For 3G systems, both processors will need to be more powerful without sacrificing battery life. Given the power constraints, the processing requirements exceed current technology.

One of the next major challenges for computer architects is designing systems in which power is a first-class design constraint. Considering the hundreds of millions of cell phones in use today and the projected sales of billions more, mobile phones will surpass the desktop as the defining application environment for computing, further underlining the need to consider power early in the design process.

An inelegant solution

The cell phone's two-processor configuration arose from the need for a low-power system that performs significant amounts of signal processing and possesses general-purpose functionality for low-resolution display support, simple database functions, and protocols for cell-to-phone communications. From an architectural perspective, this solution lacks elegance. A "convergent" architecture that handles both signal-processing and general-purpose computing requirements offers a better alternative. However, it may be easier to manage the power to separate components so that users can easily turn off the power to either one. More research on such trade-offs is necessary.

Applications where power is key

While the cell phone and its derivatives will become the leading users of power-efficient systems, this is by no means the only application where power is key. For example, a server farm has a workload of independent programs. Thus, it uses parallelism without the inefficiencies that intraprogram communication and synchronization often introduces—making multiprocessors an attractive solution.

A typical front-end server that handles mail, Web pages, and news has an Intel-compatible processor, 32 Mbytes of memory, and an 8-Gbyte disk, and requires

about 30 watts of power. Assume that the processor is an AMD Mobile K6, with a total of 64 Kbytes of cache running at 400 MHz, and is rated at 12 watts. Compare this processor with the XScale, which has the same total cache size but consumes only 450 milliwatts at 600 MHz. The processor can actually run from about 1 GHz to less than 100 MHz; at 150 MHz, it consumes 40 milliwatts. Replacing the K6 with 24 XScales does not increase power consumption. To process as many jobs as the 24-headed multiprocessor, the K6 will require an architectural efficiency about 24 times that of an XScale.

Some may disagree with this analysis. For example, it does not account for the more complex processor-memory interconnect that the multiprocessor requires, and it doesn't consider the fact that the individual jobs can have an unacceptable response time. However, the analysis shows that if power is the chief design constraint, a low-power, nontrivial processor like the XScale can introduce a new perspective into computer architecture. If we replaced the K6 with a 100-watt Compaq 21364, it would need to be *200 times* as efficient as the XScale.

FUTURE CHALLENGES

It is remarkable how quickly some microprocessor manufacturers have already deployed some ideas for lowering power consumption: The XScale provides but one example.

However, if power consumption is to continue to be reduced, an important problem requires attention in the near term, that is, the leakage current. As devices shrink to submicron dimensions, the supply voltage must be reduced to avoid damaging electric fields. This development, in turn, requires a reduced threshold voltage. In Equation 3, leakage current increased exponentially with a decrease in $V_{threshold}$. In fact, a 10 percent to 15 percent reduction can cause a two-fold increase in I_{leak}. In increasingly smaller devices, leakage will become the dominant source of power consumption. Further, leakage occurs as long as power flows through the circuit. This constant current can produce an increase in the chip temperature, which in turn causes an increase in the thermal voltage, leading to a further increase in leakage current. In some cases, this vicious circle results in unconstrained *thermal runaway*.

Because they are keenly aware of this pitfall, circuit designers have proposed several techniques to handle it. The present popular solution employs two types of field-effect transistors: low $V_{threshold}$ devices for the high-speed paths and high $V_{threshold}$ devices for paths that are not critical, as addressed in Equation 2. Such *voltage clustering*[9] makes additional demands on computer-aided design tools. Future circuits must operate properly with several supply-voltage levels and thresholds.

At the architectural level, some techniques for controlling dynamic power will also help to reduce leakage effects. In many cases, components that were shielded from unnecessary activity, such as static random-access memory, are also candidates for implementation with lower-power-threshold devices and can tolerate the performance hit.

Although the contributions of process engineers and circuit designers to power-aware designs are crucial, architects and systems-level designers also need to contribute a twofold improvement in power consumption per design generation.

Elevating power to a first-class constraint must be a priority early in the design stage when designers make architectural trade-offs as they perform cycle-accurate simulation. This presents a problem because designers can make accurate power determination only after they perform chip layout. However, designers can usually accept approximate values early in the design flow, provided they accurately reflect trends. For example, if the architecture changes, the approximate power figure should reflect a change in power in the correct direction.

Several research efforts are under way to insert power estimators into cycle-level simulators. Event counters obtain frequency measures for architectural components such as adders, caches, decoders, and buses to estimate power. Researchers at Intel,[10] Princeton,[11] and Penn State University[12] have developed power estimators based on the SimpleScalar simulator widely used in academe (see http://www.simplescalar.org). A fourth effort, PowerAnalyzer, which I am developing with Todd Austin and Dirk Grunwald, is expanding on previous work to provide estimates for multiple thresholds, di/dt noise, and peak power (see http://www.eecs. umich.edu/~tnm/power/power.html).

Acknowledgment

This work was supported, in part, by contract F33615-00-C-1678 from the Power Aware Computing and Communications program at DARPA.

References

1. R. Gonzalez and M. Horowitz, "Energy Dissipation in General-Purpose Microprocessors," *IEEE J. Solid-State Circuits*, IEEE Press, Piscataway, N.J., 1996, pp. 1277-1284.

2. S. Moore et al., "Self-Calibrating Clocks for Globally Asynchronous Locally Synchronous Systems," *Proc. Int'l Conf. Computer Design*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 73-78.
3. S. Manne, A. Klauser, and D. Grunwald, "Pipeline Gating: Speculation Control for Energy Reduction," *Proc. 25th Int'l Symp. Computer Architecture*, IEEE Press, Piscataway, N.J., 1998, pp. 132-141.
4. M. Johnson, M. Gupta, and W. Mangione-Smith, "Filtering Memory References to Increase Energy Efficiency," *IEEE Trans. Computers*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 1-15.
5. L. Benini et al., "Address Bus Encoding Techniques for System-Level Power Optimization," *Proc. 1998 Design Automation and Test in Europe* (DATE 98), IEEE CS Press, Los Alamitos, Calif., 1998, pp. 861-866.
6. A. Park, M. Farrens, and G. Tyson, "Modifying VM Hardware to Reduce Address Pin Requirements," *Proc. 25th Int'l Symp. Computer Architecture*, IEEE Press, Piscataway, N.J., 1992, pp. 1-4.
7. C. Lefurgy, E. Piccininni, and T. Mudge, "Reducing Code Size with Run-Time Decompression," *Proc. 6th Int'l Symp. High-Performance Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 218-227.
8. T. Pering, T. Burd, and R. Brodersen, "Voltage Scheduling in the lpARM Microprocessor System," *Proc. 2000 Int'l Symp. Low Power Electronics and Design*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 96-101.
9. K. Usami and M. Horowitz, "Clustered Voltage Scaling Technique for Low-Power Design," *Int'l Symp. Low-Power Design*, ACM Press, New York, 1995, pp. 3-8.
10. G. Cai and C. Lim, "Architectural Level Power/Performance Optimization and Dynamic Power Estimation," *Cool Chips Tutorial: An Industrial Perspective on Low-Power Processor Design*, T. Mudge, S. Manne, and D. Grunwald, eds., IEEE CS Press, Los Alamitos, Calif., 1999, pp. 90-113; http://www.eecs.umich.edu/~tnm/cool.html.
11. D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proc. 27th Int'l Symp. Computer Architecture*, IEEE Press, Piscataway, N.J., 2000, pp. 83-94.
12. N. Vijaykrishnan et al., "Energy-Driven Integrated Hardware-Software Optimizations Using SimplePower," *Proc. 27th Int'l Symp. Computer Architecture*, IEEE Press, Piscataway, N.J., 2000, pp. 95-106.

Trevor Mudge *is a professor in the Department of Computer Science and Electrical Engineering at the University of Michigan, Ann Arbor. His research interests include computer architecture, very large scale integration design, and compilers. Mudge received a PhD in computer science from the University of Illinois at Urbana-Champaign. Contact him at tnm@eecs.umich.edu.*