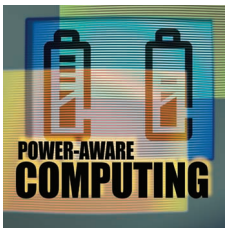
Leakage Current: Moore's Law Meets Static Power



Microprocessor design has traditionally focused on dynamic power consumption as a limiting factor in system integration. As feature sizes shrink below 0.1 micron, static power is posing new low-power design challenges.

Nam Sung Kim
Todd Austin
David Blaauw
Trevor Mudge
University of Michigan, Ann Arbor

Krisztián Flautner
ARM Ltd.

Jie S. Hu
Mary Jane Irwin
Mahmut Kandemir
Vijaykrishnan Narayanan
Pennsylvania State University

Power consumption is now the major technical problem facing the semiconductor industry. In comments on this problem at the 2002 International Electron Devices Meeting, Intel chairman Andrew Grove cited off-state current leakage in particular as a limiting factor in future microprocessor integration.[1]

Off-state leakage is *static power*, current that leaks through transistors even when they are turned off. It is one of two principal sources of power dissipation in today's microprocessors. The other is *dynamic power*, which arises from the repeated capacitance charge and discharge on the output of the hundreds of millions of gates in today's chips.

Until very recently, only dynamic power has been a significant source of power consumption, and Moore's law has helped to control it. Shrinking processor technology has allowed and, below 100 nanometers, actually required reducing the supply voltage. Dynamic power is proportional to the square of supply voltage, so reducing the voltage significantly reduces power consumption.

Unfortunately, smaller geometries exacerbate leakage, so static power begins to dominate the power consumption equation in microprocessor design.

PROCESS TRENDS

Historically, complementary metal-oxide semiconductor technology has dissipated much less power than earlier technologies such as transistor-transistor and emitter-coupled logic. In fact, when not switching, CMOS transistors lost negligible power. However, the power they consume has increased dramatically with increases in device speed and chip density.

The research community has recognized the significance of this increase for some time. Figure 1 shows total chip dynamic and static power consumption trends based on 2002 statistics normalized to the *2001 International Technology Roadmap for Semiconductors*.[2] The ITRS projects a decrease in dynamic power per device over time. However, if we assume a doubling of on-chip devices every two years, total dynamic power will increase on a per-chip basis. Packaging and cooling costs as well as the limited power capacity of batteries make this trend unsustainable.

Figure 1 also shows exponential increases projected for the two principal components of static power consumption:

- subthreshold leakage, a weak inversion current across the device; and
- gate leakage, a tunneling current through the gate oxide insulation.

The ITRS expects the rate of these increases to level out in 2005 but to remain substantial nonetheless. Even today, total power dissipation from chip leakage is approaching the total from dynamic power, and the projected increases in off-state subthreshold leakage show it exceeding total dynamic power consumption as technology drops below the 65-nm feature size.

Published by the IEEE Computer Society
0018-9162/03/$17.00 © 2003 IEEE

If they reach mainstream production, emerging techniques to moderate the gate-oxide tunneling effect—primarily by using high-*k* dielectrics to better insulate the gate from the channel—could bring gate leakage under control by 2010.

As leakage current becomes the major contributor to power consumption, the industry must reconsider the power equation that limits system performance, chip size, and cost.

POWER BASICS

Five equations model the power-performance tradeoffs for CMOS logic circuits. We present them here in simplifications that capture the basics for logic designers, architects, and system builders. The first three are common in the low-power literature.[3] The last two model subthreshold and gate-oxide leakage.

Operating frequency and voltage

The first relation shows the dependency of operating frequency on supply voltage:

$$f \propto (V - V_{th})^{\alpha} / V \qquad (1)$$

where V is the transistor's supply voltage, V_{th} is its threshold or switching voltage, and the exponent α is an experimentally derived constant that, for current technology, is approximately 1.3.

We can use this relation to develop an equation relating frequency and supply voltage. First, consider an operating voltage V_{norm} and frequency f_{norm}, which are normalized to the maximum operating voltage V_{max} and frequency f_{max}. Then approximate a linear relationship of frequency to voltage with the following equation:

$$V_{norm} = \beta_1 + \beta_2 \cdot f_{norm} \qquad (2)$$

where the constants $\beta_1 = V_{th} / V_{max}$ and $\beta_2 = 1 - \beta_1$. From Equation 1 we see that $f = 0$ corresponds to $V_{norm} = V_{th} / V_{max}$, which for today's technology is approximately 0.3. The simple relationship that Equation 2 expresses closely matches recent industrial data.[4]

Note that f_{max} corresponds to V_{max} and that, as the relation in Equation 1 specifies, the frequency drops to zero when V is reduced to V_{th}. Equation 2 also indicates that reducing the operating frequency by a particular percentage from f_{max} will reduce the operating voltage by a smaller percentage. For instance, if we assume $\beta_1 = 0.3$, reducing the frequency by 50 percent ($f_{norm} = 0.5$) will reduce the operating voltage by 35 percent ($V_{norm} = 0.65$). Conversely, reducing the voltage by half ($V_{norm} = 0.5$) will reduce the operating frequency by more than half ($f_{norm} \approx 0.3$).

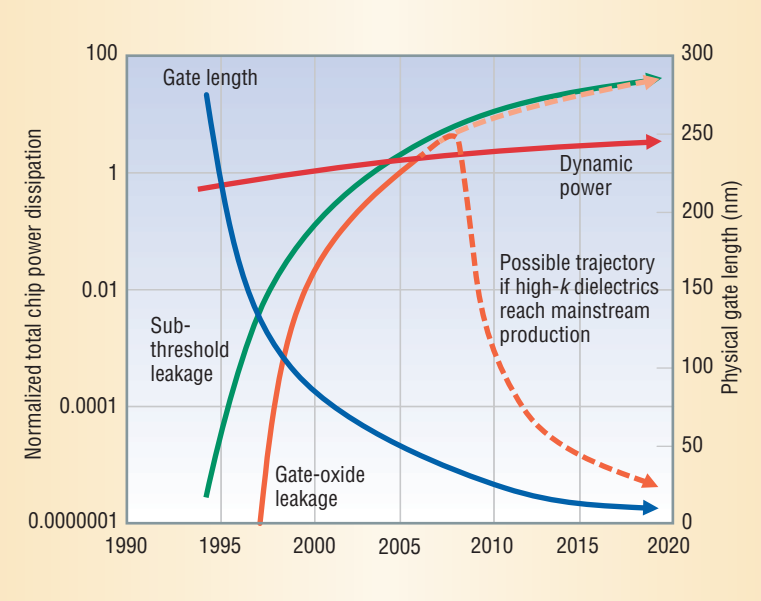


Figure 1. Total chip dynamic and static power dissipation trends based on the* International Technology Roadmap for Semiconductors. *The two power plots for static power represent the 2002 ITRS projections normalized to those for 2001. The dynamic power increase assumes a doubling of on-chip devices every two years.

Overall power consumption

The third equation defines overall power consumption as the sum of dynamic and static power:

$$P = ACV^2f + VI_{leak} \qquad (3)$$

The first term is the dynamic power lost from charging and discharging the processor's capacitive loads: A is the fraction of gates actively switching and C is the total capacitance load of all gates. The second term models the static power lost due to leakage current, I_{leak}.

We have ignored power lost to the momentary short circuit at a gate's output whenever the gate switches. The loss is relatively small; it contributes to dynamic power loss, and the equation's first term can absorb it, if necessary.

When dynamic power is the dominant source of power consumption—as it has been and as it remains today in many less aggressive fabrication technologies—we can approximate Equation 3 with just the first term. Its V^2 factor suggests reducing supply voltage as the most effective way to decrease power consumption. In fact, halving the voltage will reduce the power consumption by a factor of four. But Equation 2 shows that halving the voltage will reduce the processor's maximum operating frequency by more than half.

To compensate for this performance loss, we can use either parallel or pipelined implementations. If the implementation runs the original serial computation as two parallel subtasks or as two pipelined subtasks, the dynamic power consumption can decrease by more than a factor of two compared to the serial case. Of course, the reduction depends on significant parallelism being pre-

Pipelining is the low-power architectural solution.

sent in the computation, but many important code classes approximate this condition, including digital signal processing and image processing.

Leakage current

But how useful are parallelism and pipelining for reducing power when static power consumption becomes a major component? As noted, leakage current, the source of static power consumption, is a combination of subthreshold and gate-oxide leakage: $I_{leak} = I_{sub} + I_{ox}$.

Subthreshold power leakage. An equation that Anantha Chandrakasan, William Bowhill, and Frank Fox[5] present shows how subthreshold leakage current depends on threshold voltage and supply voltage:

$$I_{sub} = K_1 W e^{-V_{th}/nV_\theta}\left(1 - e^{-V/V_\theta}\right) \quad (4)$$

K_1 and n are experimentally derived, W is the gate width, and $V\theta$ in the exponents is the thermal voltage. At room temperature, $V\theta$ is about 25 mV; it increases linearly as temperature increases. If I_{sub} grows enough to build up heat, $V\theta$ will also start to rise, further increasing I_{sub} and possibly causing thermal runaway.

Equation 4 suggests two ways to reduce I_{sub}. First, we could turn off the supply voltage—that is, set V to zero so that the factor in parentheses also becomes zero. Second, we could increase the threshold voltage, which—because it appears as a negative exponent—can have a dramatic effect in even small increments. On the other hand, we know from Equation 1 that increasing V_{th} will reduce speed. The problem with the first approach is loss of state; the problem with the second approach is the loss of performance.

Gate width W is the other contributor to subthreshold leakage in a particular transistor. Designers often use the combined widths of all the processor's transistors as a convenient measure of total subthreshold leakage.

Gate-oxide power leakage. Gate-oxide leakage is less well understood than subthreshold leakage. For our purposes, a simplification of equations from Chandrakasan, Bowhill, and Fox[5] is sufficient to illustrate the key factors:

$$I_{ox} = K_2 W \left(\frac{V}{T_{ox}}\right)^2 e^{-\alpha T_{ox}/V} \quad (5)$$

K_2 and α are experimentally derived. The term of interest is oxide thickness, T_{ox}. Clearly, increasing T_{ox} will reduce gate leakage. Unfortunately, it also degrades the transistor's effectiveness because T_{ox} must decrease proportionally with process scaling to avoid short channel effects. Therefore, increasing T_{ox} is not an option. The research community is instead pursuing the development of high-k dielectric gate insulators.

As with subthreshold leakage, a die's combined gate width is a convenient measure of total oxide leakage.

Low-power architectural options

Because subthreshold and oxide leakage both depend on total gate width or, approximately, gate count, a pipelined implementation's contribution to leakage is comparable to the simple serial case, apart from the extra gates that latches introduce to separate the pipe stages. Pipelined implementations can run at a lower voltage, which can reduce power consumption for both dynamic and static power compared to the serial case.

Parallel implementations can also run at a lower voltage, but only by roughly doubling the amount of hardware. Thus, depending on some of the equations' experimental constants, the parallel case could leak more power than the serial case—even to the point of offsetting any savings in dynamic power.

Pipelining is therefore the low-power solution. It will always leak less power than the parallel case because it has about half the hardware, and it will leak less power than the serial case because it runs at a lower voltage. In fact, pipelining's combined dynamic and static power leakage will be less than that of the serial case.

REDUCING STATIC POWER CONSUMPTION

Unlike dynamic power, leakage is not activity based, so reducing node switching when there is no work does not help reduce power consumption. Shutting off the inactive part of the system does help, but it results in loss of state.

Retention flip-flops

When a device is inactive for a long period of time, a "snooze" mode may help if the save/restore cost is small compared to the power that the snooze time conserves. For shorter inactive periods, researchers have developed "balloon" logic, also called retention flip-flops. The idea is to use high-V_{th} latches to duplicate those latches that must preserve state. As Equation 4 shows, the high-V_{th} latches have a dramatically reduced subthreshold leakage. In snooze mode, the control logic copies

the main latch's state to the retention latch and turns off the main latch to save energy.

A transistor's threshold voltage depends on its design and process technology. Typical values are 450 mV in 180-nm processes and 350 mV in 130-nm technologies. Using doping techniques or applying a bias voltage to the substrate can increase threshold voltage by 100 mV. This in turn reduces leakage by a factor of about 10, but it increases switching time by about 15 percent. Thus, low-leakage retention flops are only useful in saving state energy efficiently—their use on the processor's critical path would slow it down. They also incur die area increases for the duplicated latches, which can be significant: 20 percent of the logic size and 10 percent of total chip area are not unexpected for small embedded cores.

Controlling memory leakage

On-chip caches constitute the major portion of the processor's transistor budget and account for a significant share of leakage. In fact, leakage is projected to account for 70 percent of the cache power budget in 70-nm technology.[6]

Figure 2 illustrates the various leakage current paths in a typical memory cell. The current through the access transistor *N*3 from the bitline is referred to as *bitline leakage*, while the current flowing through transistors *P*1 and *N*2 is *cell leakage*.

Both bitline and cell leakage result from subthreshold conduction—current flowing from the source to drain even when gate-source voltage is below the threshold voltage. In addition, gate-oxide leakage current is flowing through the transistor gates.

Circuit techniques. Two broad categories of circuit techniques aim to reduce leakage: state-destructive and state-preserving.

State-destructive techniques use ground gating, also called gated-V_{dd}. Ground gating adds an NMOS (*n*-channel metal-oxide semiconductor) sleep transistor to connect the memory storage cell and the power supply's ground.[7-9] Turning a cache line off saves maximum leakage power, but the loss of state exposes the system to incorrect turn-off decisions. Such decisions can in turn induce significant power and performance overhead by causing additional cache misses that off-chip memories must satisfy.

State-preserving techniques vary. Drowsy caches multiplex supply voltages according to the state of each cache line or block. The caches use a low-retention voltage level for drowsy mode, retaining the data in a cache region and requiring a high voltage level to access it.[10] Waking up the drowsy cache lines is treated as a pseudo cache miss and incurs one additional cycle overhead.

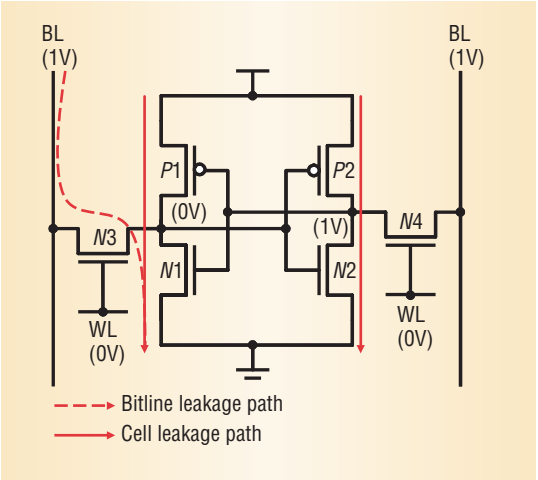


Figure 2. Leakage current paths in a memory cell. The bitline leakage current flows through the access transistor N3 from the bitline, while the cell leakage flows through transistors P1 and N2.

Other proposed state-preserving techniques include gradually decreasing threshold voltages[11] and using preferred data values.[12] All these techniques reduce leakage less than turning a cache line off completely, but accessing the low-leakage state incurs much less penalty. Moreover, while state-preserving techniques can only reduce leakage by about a factor of 10, compared to more than a factor of 1,000 for destructive techniques, the net difference in power consumed by the two techniques is less than 10 percent. When the reduced wake-up time is factored into overall program runtime, state-preserving techniques usually perform better. They have the additional benefit of not requiring an L2 cache.

Control techniques. There are two broad categories of control techniques for leakage-saving features:

- application-sensitive controls, based on runtime performance feedback,[9,13] and
- application-insensitive controls, which periodically turn off cache lines.[6,7,10]

The degree of control varies significantly within each group. For example, an application-sensitive technique[9] may indicate that the processor can turn off 25 percent of the cache because of a very high hit rate, but it provides no guidance about which 75 percent of the cache lines will be used in the near future. On the other hand, an insensitive technique like the cache-decay algorithm[7] keeps track of statistics for each cache line and thus may provide better predictive behavior.

Application-insensitive algorithms do not optimize for the specific workload. Instead, they aim for good average behavior and minimize the downside of misprediction. Periodically turning off a cache line is an example of such a scheme.[10] Its success depends on how well the selected period reflects the rate at which the instruction or data working set changes. Specifically, the optimum

Hotspots and Code Sequentiality

Researchers at the Pennsylvania State University have developed a scheme to manage instruction cache leakage that is sensitive to changes in temporal and spatial locality during program execution.[1]

The scheme builds on drowsy cache techniques. It associates each cache line with a mode bit that controls whether the line is awake and accessible or in low-leakage drowsy mode. Periodically, a global sleep signal resets all the mode bits to put all the cache lines in drowsy mode.

This approach is based on the notion that working sets change periodically. In reality, the change is gradual, so asserting the sleep signal will unnecessarily put some lines that are part of the working set into drowsy mode.

Identifying hotspots

One improvement on this basic idea prevents inadvertent mode transitions by augmenting each cache line with a local voltage-control-mask bit. When set, the VCM masks the influence of the global sleep signal and prevents mode transition. The VCM bits are set based on information from an enhanced branch target buffer. The BTB monitors how often an application accesses the different basic blocks and uses this information to identify whether they belong to a program hotspot. Once the BTB determines that a program is within a hotspot, the processor sets a global mask bit, then sets the VCM bits of all accessed cache lines to indicate the program hotspot. The processor updates the BTB access-frequency counters and the VCM bits periodically to reflect the change in program phase.

Predicting transitions

A second improvement focuses on predictively transitioning the cache lines that an application program will access next from sleep to normal mode. The predictive strategy avoids the performance and associated leakage penalty incurred from accessing a cache line in sleep mode. Since sequentiality is the norm in code execution, the program counter predictively transitions the next cache line to the normal mode when it accesses the current line. This technique is referred to as *just-in-time activation*.

Tracking access moves

Finally, asserting the global sleep signal when accesses move from one cache sub-bank to another enables the processor to identify opportunities for sleep mode transition when spatial locality changes. This differs from asserting the global sleep signal periodically to capture temporal locality changes.

Performance improvements

When averaged across 14 SPEC2000 benchmarks, these three improvements provide an average leakage energy savings of 63 percent in the instruction cache compared to using no leakage management, 49 percent compared to the bank-based turnoff scheme, and 29 percent over the compiler-based turnoff scheme.

Reference

1. J. Hu et al., "Exploiting Program Hotspots and Code Sequentiality for Instruction Cache Leakage Management," *Proc. Int'l Symp. Low-Power Electronics and Design* (ISLPED 03), ACM Press, 2003, pp. 402-407.

period may change not only across applications but also within the different phases of one application.

In such cases, the algorithm must either keep the cache lines awake longer than necessary or turn off the lines that hold the current instruction working set, which slows performance and wastes energy. Addressing the first problem by decreasing the period will exacerbate the second problem. On the plus side, this approach is simple, has little implementation overhead, and works well for data caches.

Application-insensitive algorithms can also exhibit pathologically bad behavior on certain code types. For example, one technique[6] wakes up (and puts to sleep) cache subbanks as execution moves between them. If the first part of the loop is in one bank and the latter part in another, the algorithm can cause frequent drowsy-awake transitions that negatively impact the loop's performance. Moreover, the leakage current dissipated by lines that are awake and not accessed can waste significant energy.

Compiler techniques

Using compiler directives might make it possible to keep some loops within bank boundaries, assuming that the compiler knows the bank structure. However, a typical large application will likely have to divide some loops across banks.

The compiler can also provide application-sensitive leakage control. For example, a program's source code could include explicit loop-level cache line turn-off instructions.[13] However, this scheme demands sophisticated program analysis and modification support as well as modifications to the instruction set architecture.

The "Hotspots and Code Sequentiality" sidebar describes a leakage management technique that exploits two main characteristics of instruction access patterns: the confinement of program execution mainly to program hotspots and the sequential access pattern that instructions exhibit.

TECHNOLOGY TRENDS AND CHALLENGES

Equation 4 shows that subthreshold leakage exhibits a strong dependence on temperature. Therefore, one approach to reducing subthreshold leakage is to actively refrigerate the chip. While this option seems promising for controlling the subthreshold leakage point, it does not address gate-oxide leakage. Further, its practical application faces significant technological and cost challenges.

Multiple threshold voltages

A more promising approach in terms of sub-

threshold current employs multiple threshold voltage technologies. Today's processes typically offer two threshold voltages. Designers assign a low threshold voltage to a few performance-critical transistors and a high threshold voltage to the majority of less timing-critical transistors. This approach incurs a high subthreshold leakage current for the performance-critical transistors, but it can significantly reduce the overall leakage.

Further, future technologies are likely to offer three threshold voltages—low, high, and extra high—or even more. This opens the way to new leakage optimizations within different portions of a cache or at different levels of its hierarchy. For instance, the address-decoder and bus-driver circuits in a cache consume a significant portion of total access time, so a designer could construct them from high-V_{th} transistors, while constructing the more numerous bit cells from extra-high-V_{th} devices and reserving low-V_{th} devices for speed-critical parts of the processor core. Furthermore, new tradeoffs become possible between the cache size and its threshold voltage.

Another possible power optimization is to significantly reduce leakage by increasing the L2 cache's threshold voltage. To compensate for the increased L2 access time, a designer could increase the L1 cache size or use faster (and leakier) transistors to construct it. Trading off cache size and threshold voltages at different levels of the on-chip cache hierarchy can reduce the total leakage without sacrificing overall performance.

These techniques do not address gate-oxide leakage.

Gate length

As CMOS technology is scaled, variations in gate length, oxide thickness, and doping concentrations are becoming more significant—particularly for intradie variations that occur among the devices of a single die.

Figure 3a shows the impact of gate length variations on leakage current. Short-channel effects, such as drain-induced barrier lowering, give subthreshold leakage current an exponential dependence on the gate length. Thus, even a 10 percent variation from the nominal length can change the leakage current by a factor of three.

Figure 3b shows the expected probability distribution of the leakage current for devices on a chip, given a Gaussian distribution of gate lengths with a standard deviation equal to 5 percent of mean for a 180-nm process. The exponential increase in leakage current generates a log-normal distribution; the long "tail" for high-leakage currents corresponds to devices with small gate lengths. This distribution implies that a small set of devices experience significantly more subthreshold leakage current than the average device. Since caches comprise a large number of devices, they have a high probability of containing a few "extremely leaky" gates.

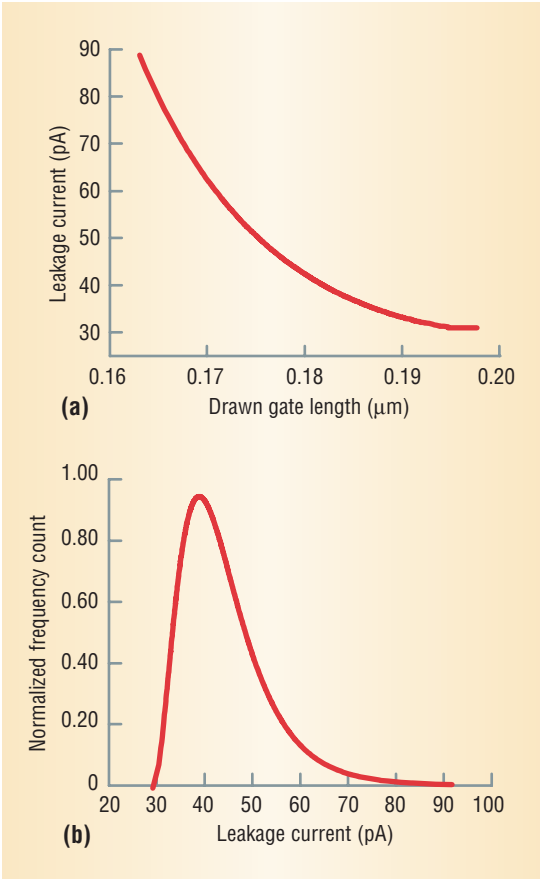


Figure 3. Leakage current and gate length. (a) Impact of gate-length on leakage current, and (b) expected probability distribution of the leakage current for devices on a chip.

Intradie process variations will require new approaches to reducing cache leakage, specifically targeting bit cells with high-leakage currents. One approach would be to identify such cells at test time and swap them with redundant cells.

Oxide tunneling

The growing significance of oxide-tunneling current poses another challenge to reducing cache leakage. Process scaling has consistently reduced the gate-oxide layer's thickness to provide sufficient current drive at reduced voltage supplies. The resulting gate-tunneling leakage current is significant, as Equation 5 shows. I_{ox} arises from the finite (nonzero) probability of an electron directly tunneling through the insulating silicon oxide layer.

Figure 1 shows that gate-oxide current leakage will catch up to subthreshold leakage in magnitude—and in many cases, it already has. The main approach to reduce gate-oxide leakage applies aggressive high-k materials with dielectric constants of 25 to 50, such as hafnium oxide. These materials greatly diminish gate-oxide leakage, but

they also pose numerous process integration problems. The ITRS does not project them reaching mainstream production until 2010.[4]

The emerging importance of gate-oxide leakage raises new challenges for leakage reduction in on-chip caches. One possible approach employs technologies with dual-oxide thicknesses. Designers could use thick-oxide transistors for cache bit cells, for example, reducing overall gate-oxide leakage in the same way high threshold voltage devices can reduce subthreshold leakage.

Other methods of reducing subthreshold leakage may apply to gate-tunneling current as well, but their effects require further research. For example, gate-oxide leakage—unlike subthreshold leakage—has a very weak dependence on temperature. Thus, as subthreshold leakage decreases with decreases in operating temperature, gate-oxide leakage becomes more dominant. This will require technology and engineering that specifically target gate-oxide reduction in standby mode.

Power consumption has become a primary constraint in microprocessor design, along with performance, clock frequency, and die size. Researchers in both industry and academia are focusing on ways to reduce it. This community has become adept at overcoming seemingly insurmountable barriers—for example, the supposed feature size limits for optical lithography dictated by the wavelength of visible light. We can expect the community to meet the power challenge in the next few years as well. ■

Acknowledgments

This work was supported by ARM, an Intel Graduate Fellowship, the US Defense Advanced Research Projects Agency, the Semiconductor Research Corporation, the Gigascale Silicon Research Center, and the National Science Foundation.

References

1. R. Wilson and D. Lammers, "Grove Calls Leakage Chip Designers' Top Problem," *EE Times*, 13 Dec. 2002; www.eetimes.com/story/OEG20021213S0040.
2. Semiconductor Industry Assoc., *International Technology Roadmap for Semiconductors, 2002 Update*; http://public.itrs.net.
3. T. Mudge, "Power: A First-Class Architectural Design Constraint," *Computer*, Apr. 2001, pp. 52-58.
4. K. Nowka et al., "A 0.9V to 1.95V Dynamic Voltage-Scalable and Frequency-Scalable 32b PowerPC Processor," *Proc. Int'l Solid-State Circuits Conf.* (ISSCC), IEEE Press, 2002, pp. 340-341.
5. A. Chandrakasan, W. Bowhill, and F. Fox, *Design of High-Performance Microprocessor Circuits*, IEEE Press, 2001.
6. N. Kim et al., "Drowsy Instruction Caches: Leakage Power Reduction Using Dynamic Voltage Scaling and Cache Sub-bank Prediction," *Proc. 35th Ann. Int'l Symp. Microarchitecture* (MICRO-35), IEEE CS Press, 2002, pp. 219-230.
7. S. Kaxiras, Z. Hu, and M. Martonosi, "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power," *Proc. 28th Int'l Symp. Computer Architecture* (ISCA 28), IEEE CS Press, 2001, pp. 240-251.
8. M. Powell et al., "Gated-V_{dd}: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," *Proc. Int'l Symp. Low-Power Electronics and Design* (ISLPED 00), ACM Press, 2000, pp. 90-95.
9. M. Powell et al., "Reducing Leakage in a High-Performance Deep-Submicron Instruction Cache," *IEEE Trans. VLSI*, Feb. 2001, pp. 77-89.
10. K. Flautner et al., "Drowsy Caches: Simple Techniques for Reducing Leakage Power," *Proc. 29th Ann. Int'l Symp. Computer Architecture* (ISCA 29), IEEE CS Press, 2002, pp. 148-157.
11. H. Kim and K. Roy, "Dynamic Vth SRAMs for Low Leakage," *Proc. Int'l Symp. Low-Power Electronics and Design* (ISLPED 02), ACM Press, 2002, pp. 251-254.
12. N. Azizi, A. Moshovos, and F.N. Najm, "Low-Leakage Asymmetric-Cell SRAM," *Proc. Int'l Symp. Low-Power Electronics and Design* (ISLPED 02), ACM Press, 2002, pp. 48-51.
13. W. Zhang et al., "Compiler-Directed Instruction Cache Leakage Optimization," *Proc. 35th Ann. Int'l Symp. Microarchitecture* (MICRO-35), IEEE CS Press, 2002, pp. 208-218.

Nam Sung Kim *is a PhD candidate at the University of Michigan. His research interests include low-power and low-complexity microarchitecture design at the circuit and microarchitectural boundary. Kim received an MS in electrical engineering from the Korea Advanced Institute of Science and Technology, Taejon, Korea. He is a student member of the IEEE and the ACM. Contact him at kimns@eecs.umich.edu.*

Todd Austin *is an associate professor of electrical engineering and computer science at the University of Michigan. His research interests include computer architecture, compilers, computer system ver-*

ification, and performance analysis tools and techniques. Austin received a PhD in computer science from the University of Wisconsin. Contact him at taustin@umich.edu.

***David Blaauw** is an associate professor of electrical engineering and computer science at the University of Michigan. His research interests include VLSI design and CAD with emphasis on circuit analysis and optimization problems for high-performance and low-power microprocessor designs. Blaauw received a PhD in computer science from the University of Illinois, Urbana-Champaign. Contact him at blaauw@umich.edu.*

***Trevor Mudge** is the Bredt Professor of Electrical Engineering and Computer Science at the University of Michigan. In addition, he runs Idiot Savants, a chip design consultancy, and advises several venture firms. His research interests include computer architecture, computer-aided design, and compilers. Mudge received a PhD in computer science from the University of Illinois, Urbana-Champaign. He is a Fellow of the IEEE and a member of the ACM, the IEE, and the British Computer Society. Contact him at tnm@eecs.umich.edu.*

***Krisztián Flautner** is a principal researcher at ARM Limited and the architect of ARM's Intelligent Energy Management technology. His research interests address high-performance, low-power processing platforms to support advanced software environments. Flautner received a PhD in computer science and engineering from the University of Michigan. Contact him at krisztian.flautner@arm.com.*

***Jie S. Hu** is a PhD candidate at Pennsylvania State University. His research interests include high-performance, low-power microprocessor design; power-efficient memory architectures; and complexity-effective instruction issue queues. Hu received an ME in signal and information processing from Peking University. He is a student member of the ACM and the IEEE. Contact him at jhu@cse.psu.edu.*

***Mary Jane Irwin** is the A. Robert Noll Chair in Engineering in the Department of Computer Science and Engineering at Pennsylvania State University. Her research interests include computer architecture, embedded and mobile computing systems design, low-power design, and electronic design automation. Irwin received a PhD in computer science from the University of Illinois. She is an IEEE Fellow, an ACM Fellow, and a member of the National Academy of Engineering. Contact her at mji@cse.psu.edu.*

***Mahmut Kandemir** is an assistant professor of computer science and engineering at Pennsylvania State University. His research interests include optimizing compilers, I/O-intensive applications, and power-aware computing. He received a PhD in electrical engineering and computer science from Syracuse University. He is a member of the IEEE and the ACM. Contact him at kandemir@cse.psu.edu.*

***Vijaykrishnan Narayanan** is an associate professor of computer science and engineering at Pennsylvania State University. His research interests are in embedded systems, energy-efficient designs, computer architecture, and VLSI design. Narayanan received a PhD in computer science from the University of South Florida. He has received several awards including the IEEE Computer Society's Richard E. Merwin Award. Contact him at vijay@cse.psu.edu.*