Dynamic Zero-Sensitivity Scheme for Low-Power Cache Memories

A low-power cache has become essential in many applications, but cache accesses contribute significantly to a chip's total power consumption. Because most bit values read from the cache are 0s, the authors introduce a dynamic zero-sensitivity (DZS) scheme that reduces average cache power consumption by preventing bitlines from discharging in reading a 0.

Yen-Jen Chang
National Chung-Hsing University

Feipei Lai
National Taiwan University

Power consumption is an increasingly important consideration in modern system design, especially for advanced microprocessors and battery-powered portable devices. On-chip caches effectively reduce the performance gap between the processor and main memory, but they contribute significantly to the chip's total power consumption. For example, caches consume about 25 percent of total chip power in the Alpha 21164 processor[1] and 43 percent in the StrongARM processor (SA-110), which targets low-power applications.[2] Clearly, caches are the most attractive targets for power reduction. Because most cache accesses are reads, especially accesses to the instruction cache, our work focuses on reducing cache power dissipated in reading.

Traditionally, for large SRAM caches, designers choose differential bitlines, which provide good noise immunity and fast sensing. In a differential-bitline design, one of the two bitlines must be discharged for each read regardless of the stored bit value, which implies that the power dissipated in reading a 0 is the same as that dissipated in reading a 1. Consequently, the most important feature of a conventional cache is that, for each read access, power consumption is fixed and independent of the read data value.

By examining the read accesses of benchmark programs, we found that an overwhelming majority of read bits are 0s for both instruction and data caches; in other words, the cache read data's bit distribution is highly skewed toward 0. This observation led to our dynamic zero-sensitivity (DZS) scheme, which exploits the prevalence of 0 bits to reduce average cache read power. In contrast to a conventional cache in which the power dissipated in reading a 1 and a 0 are the same, the DZS scheme prevents the bitlines from discharging in reading a 0, so that the power dissipated in reading a 0 is far less than that dissipated in reading a 1. Therefore, unlike a conventional cache in which read power is insensitive to the percentage of 0s in the read data, the DZS cache's power consumption for each read access varies with

Published by the IEEE Computer Society

Related work

To reduce cache power consumption, researchers have proposed many techniques based on the observation that the cache access stream exhibits a strong bias toward 0 at the bit level. Tseng and Asanovic propose a single-ended read bitline to minimize the number of bitline transitions, while modifying register file bit cells so that reading a 0 causes no bitline discharge.[1] The major difference between their work and ours is that their single-ended bitline design is only suitable for register cells in which the bitlines are shorter than the cache bitlines. Park, Chang, and Kyung propose a scheme for ROMs and small RAMs, using single-ended bitlines to reduce the total number of bitline discharges by conditionally inverting stored words.[2]

Villa, Zhang, and Asanovic propose the dynamic zero compression scheme to reduce the energy required for cache accesses by writing and reading only a single bit for every 0-value byte.[3] The DZC method adds a 0 indicator bit (ZIB) to each byte, which indicates whether the byte contains all 0 bits. On a read access, only the ZIB is read if the byte is 0; otherwise, both the data bits and the ZIB are read. DZC's major disadvantage is that the cluster of 0 bits limits power reduction. This is especially unfavorable for instructions due to the field limitation of the instruction format. In contrast, our DZS scheme effectively reduces cache power consumption without requiring the 0 bit cluster. Two examples show our method is superior to DZC in reducing bitline discharges: 1) If a byte is 00000001, we cannot apply DZC to reduce power consumption. In addition, we would have to pay the additional power penalty incurred by the ZIB. In the same case, our scheme prevents seven bitlines from being discharged, reducing cache read power. 2) If a byte is 00000000, DZC can work well, but the ZIB's read power is still needed. In contrast, because our method consumes almost no power in reading a 0 and needs no additional bit, it consumes less read power.

The preceding techniques mainly reduce the cache's dynamic power. In contrast, Azizi et al. propose an asymmetric SRAM cell in which selected transistors are implemented with high V_T (threshold voltage) to reduce leakage power when the cell is storing a 0 (the common case).[4,5] This technique maintains the traditional SRAM architecture to reduce leakage power consumption in storing a 0. In contrast, our technique modifies the traditional SRAM architecture to reduce dynamic power consumption in reading a 0.

References

1. J.H. Tseng and K. Asanovic, "Energy-Efficient Register Access," *Proc. 13th Symp. Integrated Circuits and Systems Design*, IEEE Press, 2000, pp. 377-382.
2. B.K. Park, Y.S. Chang, and C.M. Kyung, "Confirming Inverted Data Store for Low Power Memory," *Proc. Int'l Symp. Low Power Electronics and Design* (ISLPED 99), IEEE Press, 1999, pp. 91-93.
3. L. Villa, M. Zhang, and K. Asanovic, "Dynamic Zero Compression for Cache Energy Reduction," *Proc. 33rd Int'l Symp. Microarchitecture* (Micro 33), IEEE Press, 2000, pp. 214-220.
4. N. Azizi et al., *Asymmetric-Cell Caches: Exploiting Bit Value Biases to Reduce Leakage Power in Deep-Submicron, High-Performance Caches*, tech. report TR-01-01-02, ECE Computer Group, Univ. of Toronto.
5. N. Azizi, A. Moshovos, and F.N. Najm, "Low-Leakage Asymmetric-Cell SRAM," *Proc. 35th Int'l Symp. Low Power Electronics and Design* (ISLPED 02), IEEE Press, 2002, pp. 48-51.

the read data's 0 percentage. The more 0 bits the read data contains, the more power the DZS scheme can save.

We present two alternate implementations of the DZS scheme. One is a traditional differential-bitline design called the DZS_D, and the other is a single-bitline design called the DZS_S, which we developed to relieve the DZS_D cache's area penalty. For both implementations, a sense amplifier modification is the most critical adaptation for ensuring that the DZS scheme performs well. Besides sensing differential voltage to V_{DD}, the amplifier must be adapted to sense the signal of no voltage difference as 0. Compared with a conventional cache, area overhead and stability degradation are the disadvantages of the DZS_D and DZS_S caches, respectively.

We evaluated the read data's 0 and 1 distribution from the SPEC2000 (http://www.spec.org) and MediaBench[3] benchmarks, and we obtained all power consumption data from an HSpice simulation of the extracted layout in Taiwan Semiconductor Manufacturing Company's (TSMC) 0.18-micron technology with a 1.8-V power supply. Our experimental results show that by minimizing the power dissipated in reading 0, the DZS scheme drastically reduces average cache read power.

Motivation

We analyzed power consumption during a read in a conventional cache and described the 0 and 1 bit distribution of instruction and data references. It is the extremely asymmetric distribution of 0 and 1 bits that motivated our DZS scheme. The "Related work" sidebar briefly reviews other techniques and compares them with ours.

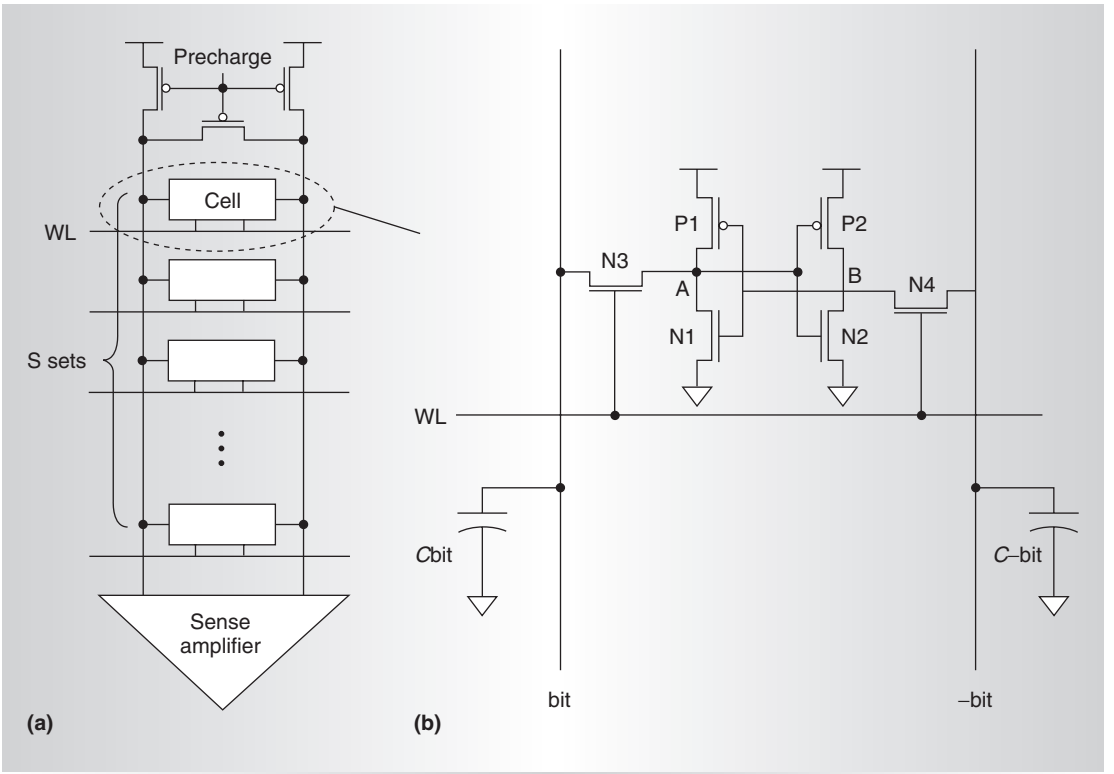

Figure 1. Column circuit for a conventional cache (a); dynamic logic design (b).

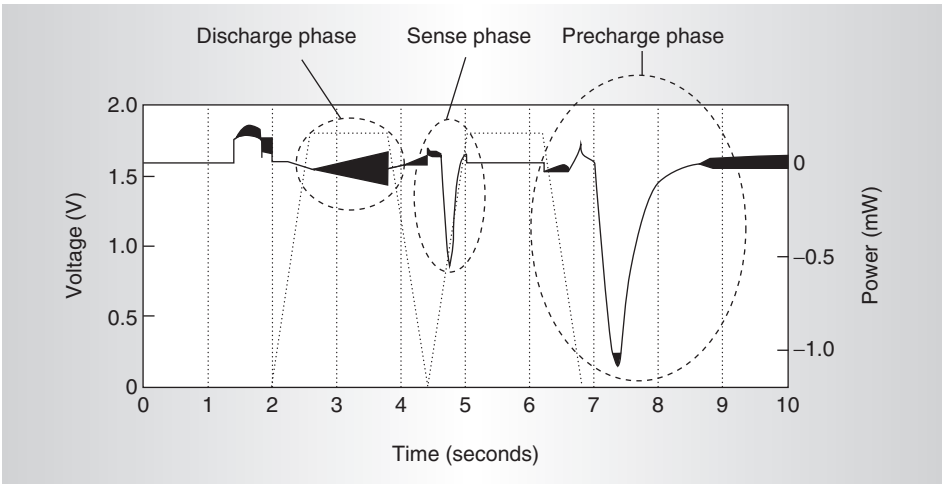

Figure 2. Column power distribution during a read.

Cache power consumption during a read

Figure 1 shows one column circuit of a conventional cache, which consists of two bitlines (*bit* and *–bit*), *S* memory cells, and a sense amplifier, where *S* is the number of sets that depend on the cache configuration. In the dynamic logic design, precharging and equilibrating the bitlines to V_{DD} initiates a read operation. When the wordline (*WL*) signal is asserted, access transistors N3 and N4 connect the bitlines to the cell. Depending on the stored value, one bitline is pulled down and the other remains high. Thus, a differential voltage is set up across the bitlines. The sense amplifier detects the difference between *bit* and *–bit* and then amplifies the differential to the logic level.

Traditionally, employing a pulsed-wordline technique to deassert the wordline signal when a sufficient voltage differential has developed on the bitlines reduces a cache read's power consumption significantly.[4] To analyze cache power consumption during a read, we used TSMC 0.18-micron technology to implement a cache column with 128 cells. As Figure 2 shows, a read has three power-consuming phases:

- bitline discharge, in which one of the two bitlines is discharged through the accessed cell;
- sense, in which the sense amplifier is enabled to detect and amplify the differential between the two bitlines; and
- precharge, in which the two bitlines are precharged to V_{DD} before the next read.

For each column, one of the two bitlines is discharged toward 0 and then precharged to V_{DD} during a read operation. In the conventional cache design, because such a bitline swing is independent of the value stored in the accessed cell, the power dissipated in reading a 1 and a 0 are the same. As measured in our simulation, the column power consumption in reading either a 1 or a 0 is 6.357×10^{-2}mW.

Distribution of 0s and 1s

We used SimpleScalar[5] to investigate the 0 and 1 bit distribution of cache references. By default, the Portable Instruction Set Architecture (PISA) instructions used in SimpleScalar consist of a 64-bit encoding to facilitate instruction set research. To make the results convincing, we enabled the *–icompress* option to remap the 64-bit instructions to 32-bit equivalents in all simulations.

Figure 3 shows the proportion of 0 bits to total reference bits (called the zero rate), which we examined from execution traces of parts of the SPEC2000 benchmarks. The figure shows that more than 45 percent of the instruction references and about 75 percent of the data references are 0 bits. For instruction caches, there are several reasons for this phenomenon: First, in the instruction format, some fields must be reserved for augmentation, and the width of some fields is overestimated for rare cases. Second, immediate values, address displacement offsets, and branch displacements are often small integers.[6] Finally, modern instruction sets usually use a sparse encoding of opcodes to accelerate decoding.[7] Similarly, for data caches, there are two underlying reasons that most read bits are 0s: In the general programming style, zero and relatively small positive integers are commonly used and are often stored as words, such as flags, loop iteration counts, and array indexes. Second, heap objects are always zeroed at allocation time, leading to a large fraction of 0s in the cache block, and linked data structures often have 0s in the upper pointer address bits because a heap grows downward.[7]

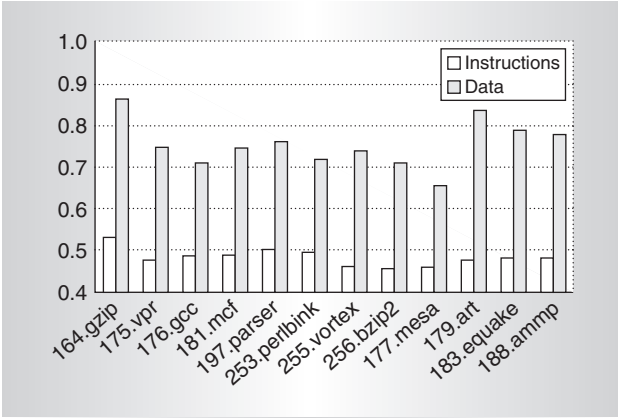

Figure 3. Zero rates of instruction and data references for SPEC2000 benchmarks.

DZS_D cache implementation

Because such a design provides better noise immunity, we first developed a differential-bitline implementation of the DZS cache called the DZS_D. To read and write information from and to multiple locations without address or data contention, we use a cell model with split one read port and one write port.

DZS_D SRAM cell

To implement the DZS_D cache, we adapted the memory cell to zero sensitivity. *Zero sensitivity* means that the bitline discharge depends on whether the value stored in the accessed cell is a 0 or not. If the stored value is a 0, both bitlines should be prevented from discharging. Otherwise, one bitline must discharge as a typical read operation. Figure 4 shows the schematic of the DZS_D cell, which dynamically prevents the *rbit* bitline from discharging. Compared with a traditional SRAM cell, an additional NMOS pass transistor N7 in the read port of the DZS_D cell connects read wordline signal *RWL* to access transistor N3.

Depending on the stored value, the DZS_D cell has two states during a read: locked and unlocked. If the stored value is a 0, the cell is in locked state. In this state, node A is held at 0, which turns off N7 to disconnect *RWL* from N3. Even if *RWL* were asserted, the *rbit* bitline is prevented from discharging

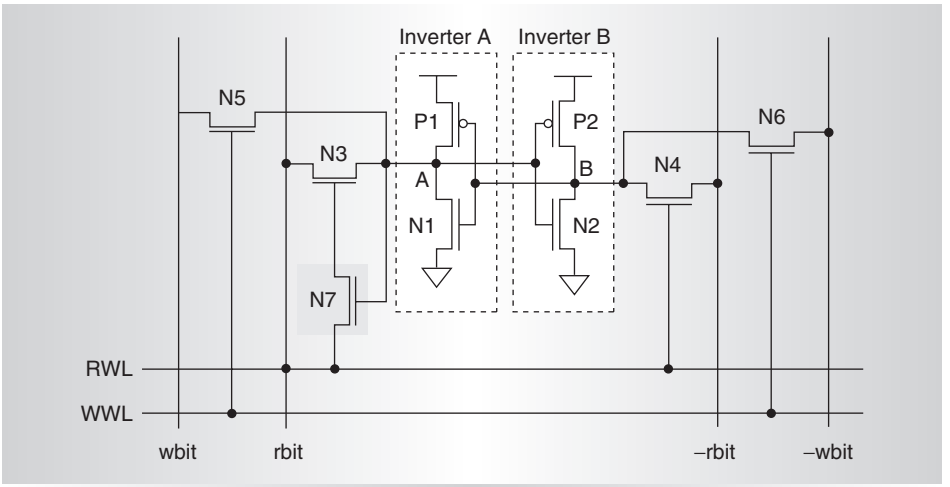

Figure 4. DZS_D SRAM cell. Read wordline (RWL) and write wordline (WWL) signals select a cell for reading and writing, respectively.

from the write port in our baseline cache design, the write port's access transistors (N5 and N6) are different from the read port's access transistors (N3 and N4). Thus, the cell state (locked or unlocked) is determined after a write operation and doesn't interfere with it.

DZS_D sense amplifier

Figure 5a shows our baseline sense amplifier design. It is a conventional latch sense amplifier with isolated bitlines. In the DZS_D cache, if the accessed cell is in locked state (that is, the stored value is a 0), the DZS_D cell disables the *rbit* bitline discharge so that there is no voltage difference across the two bitlines. To avoid having the sense amplifier hang and burn static current or swing due to threshold voltage mismatches or noise, we modify the conventional latch sense amplifier to sense 0 if there is no voltage difference between the two bitlines. Figure 5b shows the DZS_D sense amplifier, which we adapted for sensing the signal of no voltage differential as 0. Compared with the conventional sense amplifier, it contains three additional NMOS transistors (N2, N3, and N4). During the sensing phase, N4 is turned on, and N2's function is to connect node *–out*

to low through N3 and N1; that is, the two read bitlines *rbit* and *–rbit* would maintain a high voltage as in the precharge phase. Thus, during a read access, if the accessed cell is in locked state, there is no power consumption due to bitline discharge.

If the stored value is a 1, the cell is in unlocked state. In this case, node A is high, which turns on pass transistor N7. When *RWL* is asserted, access transistors N3 and N4 conduct, and then the *–rbit* bitline discharges to low through N4 and N2 as a typical read 1 operation.

The DZS_D cell is used only in the read-only port. Because the read port is separate

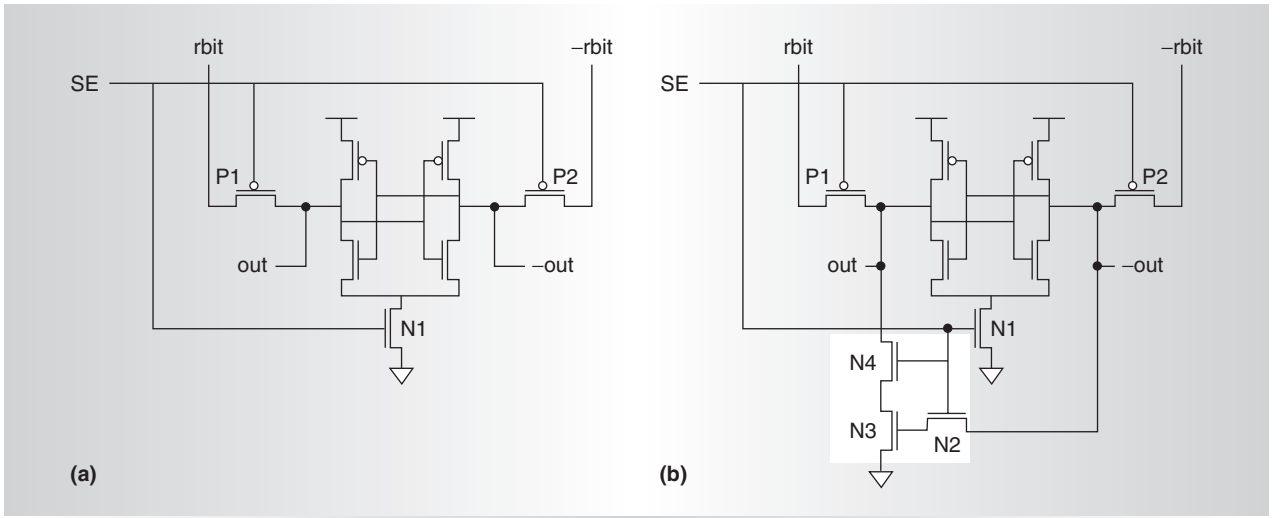

Figure 5. Sense amplifier modification: conventional latch sense amplifier (a); DZS_D sense amplifier (b).

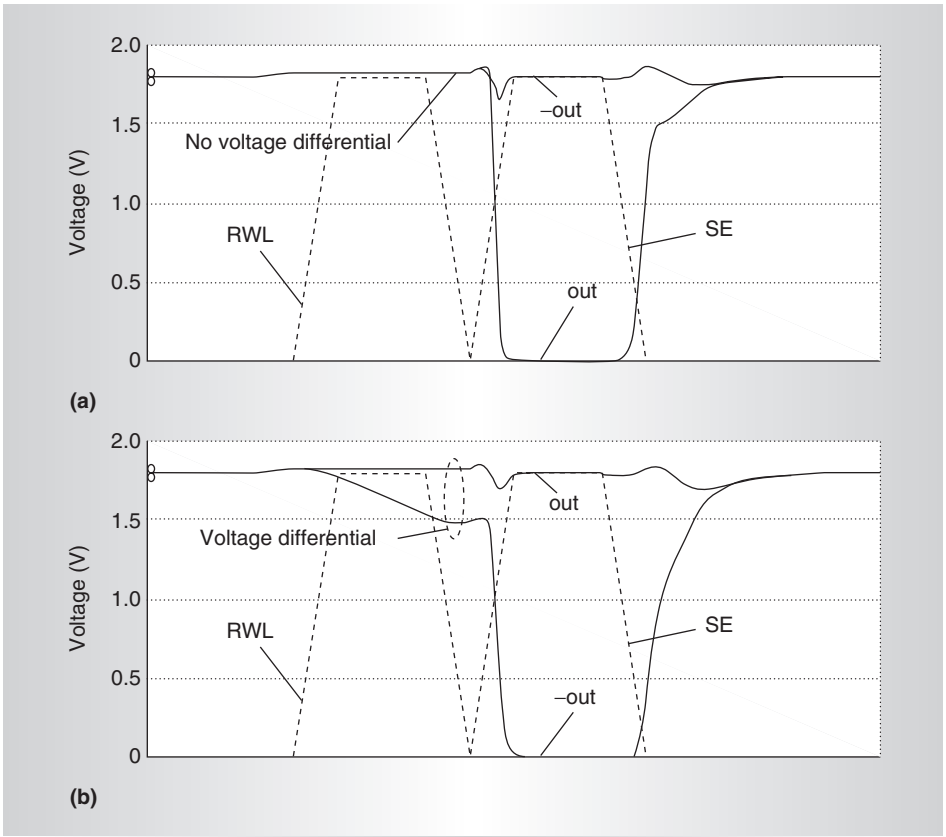

Figure 6. HSpice waveform of DZS_D sense amplifier: nondifferential sensing (a); differential sensing (b).

to the gate of N3 responsible for pulling down the voltage of node *out* if necessary.

Two operations are possible in the DZS_D sense amplifier: differential sensing and nondifferential sensing. If the accessed cell is in locked state, neither of the two bitlines discharges. Because nodes *out* and *–out* follow the values of *rbit* and *–rbit*, which implies *out* = *–out* = V_{DD}, this case is nondifferential sensing. At the beginning of sensing, transistor N2 conducts to pass the *–out* signal to turn on transistor N3, and then the *out* signal discharges to low through N4 and N3. Thus, the *out* voltage is lower than the *–out* voltage. After the inverter pair amplifies this differential to a full rail-to-rail signal, the voltages of nodes *out* and *–out* are 0 and V_{DD}, respectively. As Figure 6a, obtained from HSpice postlayout simulation, shows, the DZS_D sense amplifier senses 0 in nondifferential sensing.

In differential sensing, the value stored in the accessed cell is a 1, which results in the *–rbit* bitline discharging. When the *SE* signal is asserted, the initial state is that the *–out* voltage is lower than the *out* voltage. Because we use the pulsed-wordline technique,[4] the *–out* voltage is about 1.5 V. Thus, pull-down transistor N3 is still turned on lightly by the *–out* signal but will soon be turned off because the inverter pair amplifies the initial voltage differential, pulling down *–out* to 0 to turn off N3. Figure 6b shows that the DZS_D sense amplifier senses a 1 in differential sensing.

The most important issue in cell design is access time. Unlike the single-ended bitline design,[8,9] in which the modified register cell consists of an asymmetric inverter pair to speed up read access, our sense amplifier is modified to facilitate read access, while the SRAM cell retains the architecture with a symmetric inverter pair.

Power reduction

Figure 7 shows column power distribution during a read for conventional and DZS_D caches. In this simulation, one cache column

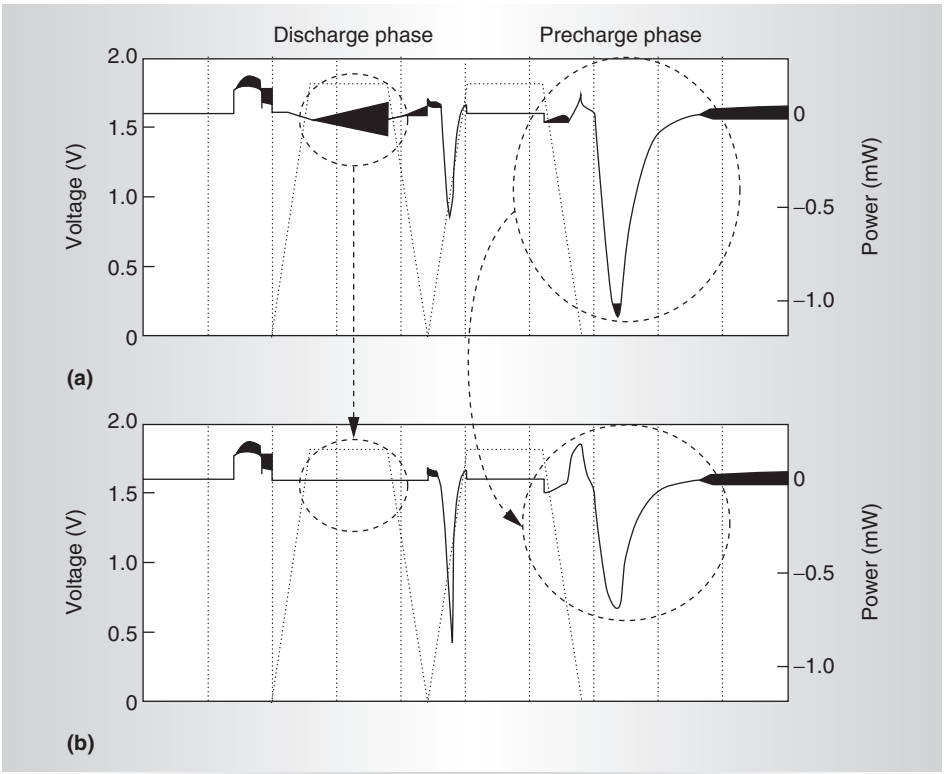

Figure 7. Column power distribution for a conventional cache during a read (a) and a DZS_D cache during a read of 0 (b).

consists of 64 bit cells. Compared with the conventional cache, in reading a 0, the DZS_D cache effectively reduces the power dissipated in bitline discharge and precharge. The DZS_D cell prevents the bitlines from discharging in locked state (that is, in reading a 0), thus completely eliminating the power dissipated in bitline swing during wordline signal assertion. After sensing, the bitlines should be precharged to high. During the precharge phase, the DZS_D cache easily achieves considerable power reduction because the DZS_D cell incurs no bitline discharge in reading a 0.

From this postlayout simulation, we summarize that power consumption per conventional cache column is 5.714×10^{-2} mW and per DZS_D column is 3.792×10^{-2}mW and 6.053×10^{-2} mW for reading a 0 and a 1, respectively. In the conventional cache, the power dissipated for reading a 1 and a 0 is the same, but in the DZS_D cache, the power dissipated for reading a 0 is much less than for reading a 1. Because the DZS scheme is beneficial only for reading a 0, the additional transistors in the DZS_D cache incur a little more power consumption than the conventional cache in reading a 1.

Cache area and performance

Clearly, the DZS_D cache's area is larger than a conventional cache's area. The additional transistors devoted to the DZS_D cell and sense amplifier introduce most of the area overhead. Compared with the traditional sense amplifier, the number of transistors in the DZS_D sense amplifier increases from seven to ten. According to the cache area model presented by Shivakumar and Jouppi,[10] we can ignore the area overhead introduced by the DZS_D sense amplifier because the area sense amps contribute to total cache area is negligible. Compared with the conventional SRAM cell, the number of transistors in the DZS_D cell increases from eight to nine, and cell area increases from 32.30 μm^2 to 36.14 μm^2. Additional pass transistor N7 imposes about a 12 percent cell area overhead. With the CACTI 3.0 tool,[10] we determined that the data array's percentage of total cache area is about 70 percent for a 32-Kbyte, two- or four-way cache. Thus, overall cache area overhead is roughly 12% × 70% = 8.4%.

We define partial read delay as the elapsed time from asserting *RWL* to the correct sensing output, which comprises the bitline discharge and sensing phases. In the DZS_D cache, because no bitline discharge is incurred in reading a 0, we consider only the case of reading a 1, in which the *–rbit* bitline discharges to low. With the increased area, bitline length increases, resulting in an increase of discharge time. In the DZS_D sense amplifier, augmented transistors N2, N3, and N4 cause one pass transistor delay. Clearly, the DZS_D cache's partial read delay is larger than that of the conventional cache. For example, if one cache column consists of 128 bit

cells, the partial read delays in an Hspice post-layout simulation are 1.1374 ns and 1.1458 ns for the conventional and DZS_D caches, respectively. The difference is minute. The partial read delay includes only discharge and sensing time. Besides partial read delay, in determining the read cycle we must consider set decode time and output driver delay. Because they usually contribute significantly to total read delay,[10] we can ignore the tiny increase in partial read delay. Thus, we conclude that the DZS_D cache causes no performance degradation in a read.

DZS_S cache: an alternate implementation

To reduce the DZS_D cache's area penalty, we developed another implementation of the DZS scheme. Because this implementation is a single-bitline design, we call it the DZS_S cache.

DZS_S SRAM cell

The DZS scheme's key concept is that there is no bitline discharge in reading a 0. By further investigating the DZS_D cell (Figure 4), we see that during a read, the *rbit* bitline always retains a high voltage as in the precharge phase, whatever the reading value is. Thus, if we remove the *rbit* bitline from the SRAM cell, the cell consists of one single-bitline read port and one differential-bitline write port. Figure 8 shows this DZS_S cell. In the DZS_S cell, we use only the *–rbit* bitline to distinguish reading a 0 from reading a 1. In reading a 0, because node B remains high, the *–rbit* bitline maintains a high voltage as in the precharge phase. This implies that reading a 0 incurred no bitline discharge. In reading a 1, because driver transistor N2 of inverter B is turned on, the *–rbit* bitline with an initial high state is discharged to low through transistors N5 and N2 as a typical read 1 operation. Unlike the DZS_D implementation, which is used only in the read-only port, the DZS_S can be applied to a cache design with a single port for both read and write. However, we must use the technique developed by Ukita[11] to perform the write operation in this single-bitline design.

DZS_S sense amplifier

By extending the DZS_D sense amplifier described earlier, we easily adapted it for the DZS_S cache. Figure 9 shows this DZS_S sense amplifier. The only difference between the DZS_D and DZS_S sense amps is the *rbit* input. Because the *rbit* bitline is absent from the DZS_S cache, the DZS_S sense amp uses the *HI* input instead of the *rbit* input. *HI* is synchronized by a precharge signal (the same as the *–rbit* bitline's precharge signal) and always precharged to high. Like the DZS_D sense amp, the DZS_S sense amp has differential and nondifferential sensing operations, both identical to the DZS_D sensing operations.

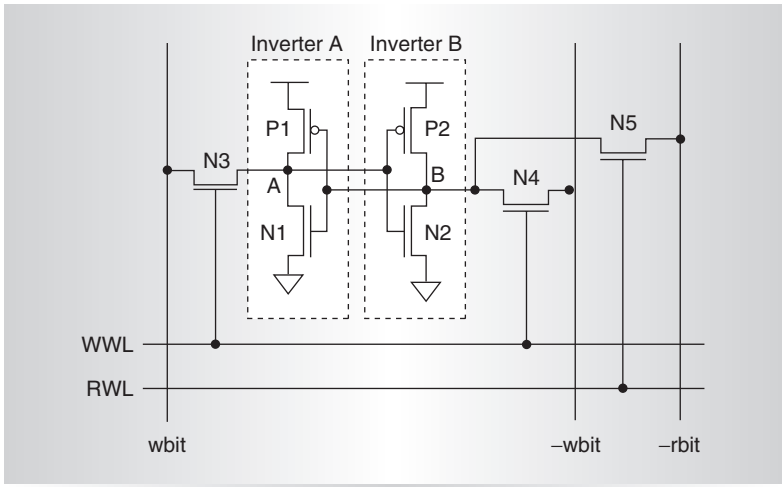

Figure 8. DZS_S cell, consisting of one single-bitline read port and one differential-bitline write port.

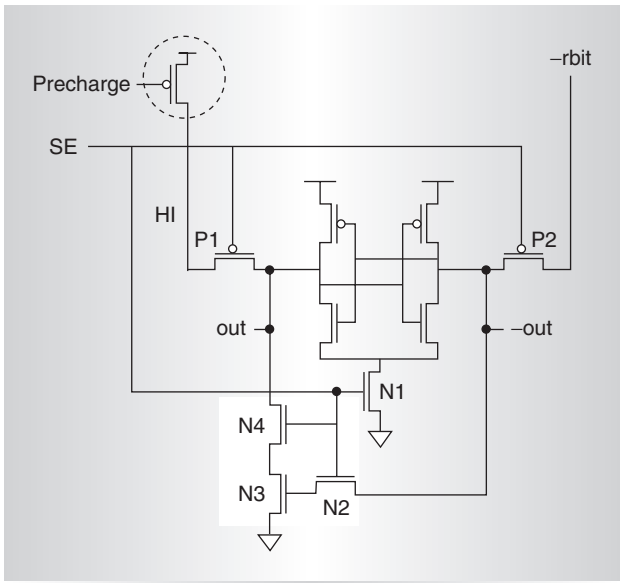

Figure 9. DZS_S sense amplifier.

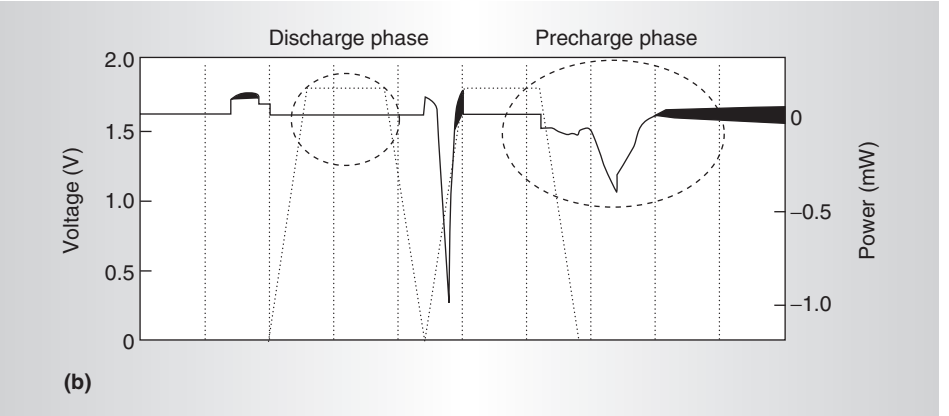

Figure 10. Column power distribution for DZS_S cache in reading a 0.

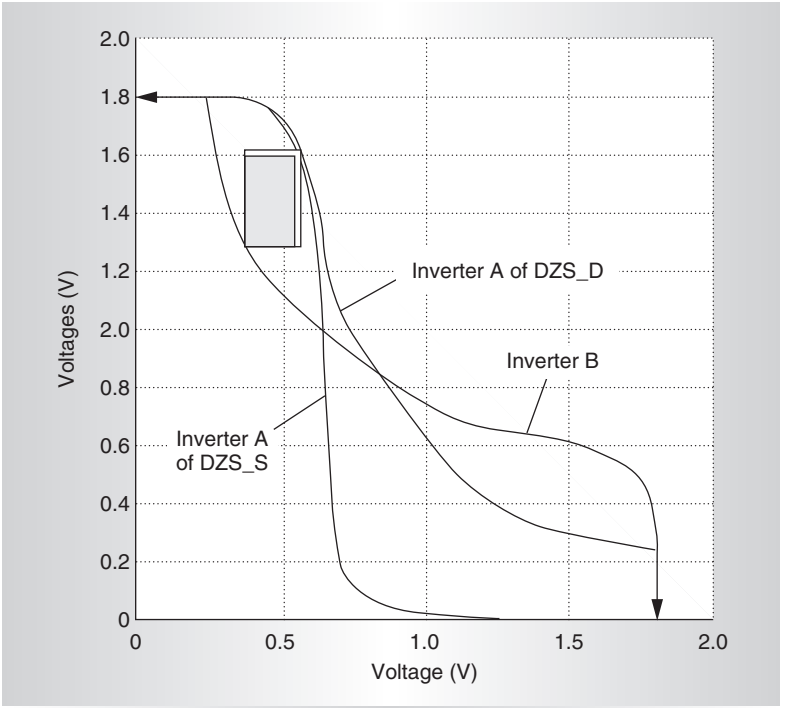

Figure 11. Graphical representation of the SNM for DZS_D and DZS_S caches.

Power reduction

Figure 10 shows the column power distribution for the DZS_S cache in reading a 0. As in the DZS_D cache, because there is no bitline discharge in reading a 0, the power dissipated in bitline swing during wordline signal assertion can be eliminated completely. In the precharge phase, the absence of the *rbit* bitline results in the DZS_S cache consuming less power than the DZS_D cache. In the post-layout simulation, the power consumption per DZS_S column is 3.372×10^{-2} mW and 6.051×10^{-2} mW in reading a 0 and a 1, respectively. Because read 1 operations of the DZS_S and DZS_D caches are identical, power consumption per cache column in reading a 1 is almost the same in the two caches.

Cache area and stability

As mentioned earlier, we don't consider the area overhead introduced by the sense amplifier modification because the area sense amps contribute to total cache area is negligible. Compared to the conventional SRAM cell, transistors decrease from eight to seven in the DZS_S cell, and the cell area decreases from 32.30 μm^2 to 28.77 μm^2. The absences of the *rbit* bitline and one access transistor reduce cell area about 10 percent, and overall cache area reduction is roughly 10% × 70% = 7%.

An important consideration in SRAM cell design is stability—the ability to hold a stable cell state. In general, the static noise margin (SNM) is an important parameter determining cell stability. A SRAM cell's SNM is the maximum value of noise that the cross-coupled inverters can tolerate before altering state. We obtain a basic understanding of the SNM by drawing and mirroring the inverter characteristics and then finding the maximum possible square between them.

Because the DZS_S cache is a single-bitline design, the absence of one bitline and one access transistor leads to a slight degradation in stability compared with the DZS_D cache. Obtained from the HSpice simulation, Figure 11 shows a representation of the SNM for the DZS_D and DZS_S caches. From the size of the square

Table 1. Benchmark summary.

Category	Benchmark	Description	Instruction count (billions)	Data count (billions)
CINT2000	164.gzip	Compression	81.64	26.63
	176.gcc	Programming language compiler	78.42	31.38
	253.perlbm	PERL programming language	43.73	20.10
	255.vortex	Object-oriented database	168.62	88.47
CFP2000	177.mesa	3D graphics library	492.14	246.40
	179.art	Image recognition, neural networks	181.40	78.28
	183.equake	Seismic wave propagation simulation	597.51	235.11
	188.ammp	Computational chemistry	1,924.89	542.42
MediaBench	adpcm	Speech compression/decompression program	enc. 0.60 dec. 0.50	enc. 0.08 dec. 0.08
	gsm	European GSM 06.10 provisional full-rate speech transcoding standard.	enc. 1.86 dec. 0.62	enc. 0.42 dec. 0.08
	jpeg	Standardized image compression/decompression program	enc. 0.10 dec. 0.03	enc. 0.03 dec. 0.01

shown in the figure, the DZS_S cache's SNM is smaller than the DZS_D cache's SNM. They are 365 mV and 393 mV, respectively. Although the DZS_S cache reduces cache power and area effectively, the penalty is a 7.1 percent degradation in cell stability.

Experimental results

For our baseline cache, we assume an on-chip architecture with split instruction and data caches, which are a 32-Kbyte, two-way instruction cache (IC) and a 32-Kbyte, four-way data cache (DC). To avoid an explosion in the number of simulations, the block size for both caches is fixed at 32 bytes. Instead of using a conventional implementation, we implemented the baseline caches with the way prediction scheme,[12] an efficient low-power technique for saving cache power. This scheme reduces cache power consumption by accessing only one predicted cache way rather than all the cache ways.

Benchmarks

We used SimpleScalar to evaluate the 0 and 1 distribution of the data read from cache for both the SPEC2000 and the MediaBench benchmark suites. SPEC2000 is a suite of general-purpose programs, specifically developed to assist in commercial evaluation and marketing of desktop computing systems. MediaBench applications focus on multimedia and communications systems. To get a good mix of CPU- and memory-intensive loads, we used four CINT2000 (integer) benchmarks, four CFP2000 (floating-point) benchmarks, and three MediaBench integer benchmarks (adpcm, jpeg, and gsm). Each MediaBench benchmark has two separate programs: encoding and decoding. Table 1 provides a brief description of the benchmarks and lists the number of instructions and data simulated for each workload.

Read-zero rate

The read-zero rate is the ratio of the number of 0 bits to the total number of bits read from cache. Its higher read-zero rate means that the proposed DZS scheme is more efficient in reducing cache power consumption during a read. Figure 12 shows the read-zero rates of each benchmark for both the IC and the DC. Because our scheme concentrates only on read accesses, we don't consider write accesses in this simulation.

Figure 12 shows the prevalence of 0 bits in the cache read data in general-purpose and multimedia programs. Because the result difference between integer and floating-point benchmarks is hardly noticeable, we don't present these two benchmarks individually. For all benchmarks depicted in this figure, the IC read-zero rates are more than 40 percent, but the DC read-zero rates of the MediaBench benchmarks are far lower than those of SPEC2000 benchmarks. Except for jpeg, the DC read-zero rates are around or less than 60 percent for the MediaBench benchmarks.

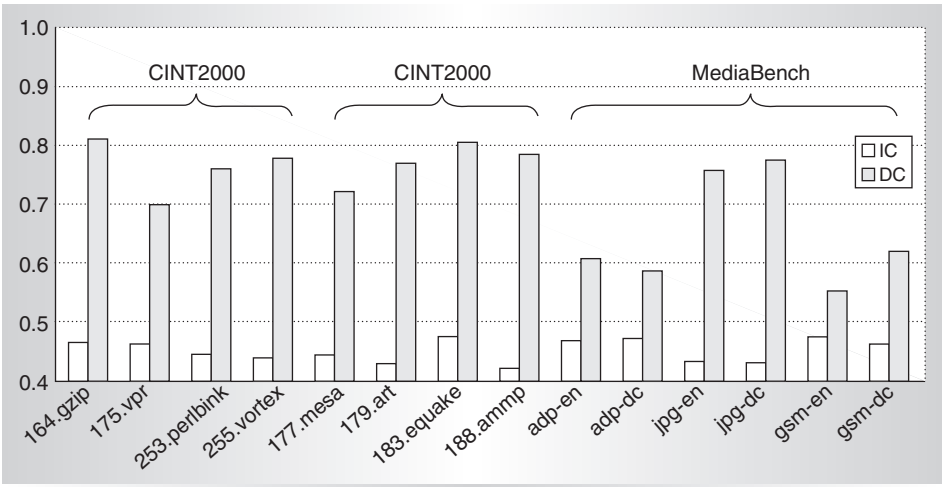

Figure 12. IC and DC read-zero rates of benchmarks.

Table 2. Average read-zero rates for SPEC2000 and MediaBench benchmarks.

Read-zero rate	SPEC2000	MediaBench
IC	0.4476	0.4564
DC	0.7653	0.6494

Table 2 summarizes the average IC and DC read-zero rates for these two benchmark suites.

Average cache power consumption during a read

A cache consists of a tag array and a data array, which store the tag and the actual data, respectively. We simplify the power dissipated in the cache data array as $P_{data\_array} = P_{way} \times A$, in which P_{way} is power consumption per cache way, and A is the degree of associativity (that is, the cache way number). Power consumption per cache way is $P_{way} = P_{col} \times N_{col}$. P_{col} is power consumption per column, and N_{col} is the number of columns per cache way. Because we fix the block size at 32 bytes in this article, N_{col} is fixed at 256.

Table 3 shows power consumption per column for the conventional cache and the proposed DZS scheme including the two cache implementations DZS_D and DZS_S. In the conventional cache, because P_{col} in reading a 1 is the same as P_{col} in reading a 0, average power consumption per cache way (P_{Conv_way}) is independent of the read-zero rate and is given as

$$P_{Conv\_way} = P_{col} \times 256. \quad (1)$$

In contrast, in the DZS cache, P_{col} in reading a 1 ($P1_{col}$) is much larger than P_{col} in reading a 0 ($P0_{col}$). Therefore, average power consumption per cache way (P_{DZS_way}) varies with the read-zero rate (ZR) and is given as

$$P_{DZS\_way} = (P0_{col} \times 256 \times ZR) + (P1_{col} \times 256 \times (1 - ZR)), \quad (2)$$

from which a higher ZR implies a lower P_{DZS_way}.

By applying the results listed in Tables 2 and 3 to Equations (1), (2), we can obtain the average power dissipated in data-array per read access, as shown in Table 4. Conv+WP is the conventional cache implemented with the way prediction scheme.

The power dissipated in the data array is only a part of total cache power consumption. With the CACTI 3.0 estimation tool, we obtained the partial components considered in our DZS scheme: data bitlines and data sense amplifiers. These contribute roughly 44.2 percent and 53.6 percent to total cache power consumption for the baseline IC and DC, respectively. To obtain the entire cache's power reduction, we must multiply data array power reduction by 0.442 and 0.536, respectively. Table 5 summarizes cache power reduction (P_{cache}) for the DZS scheme. Compared with a conventional instruc-

Table 3. Power consumption per column for conventional and DZS caches.

	Conventional	DZS_D cache		DZS_S cache	
Pcol (mW)	cache	Read 1	Read 0	Read 1	Read 0
IC (32K-2W)	8.80E-02	9.34E-02	3.83E-02	9.20E-02	3.45E-02
DC (32K-4W)	7.47E-02	7.95E-02	3.83E-02	7.82E-02	3.43E-02

Table 4. Cache power dissipated in data array per read access.

	Base	DZS_D		DZS_S	
$P_{data\ array}$ (mW)	(Conv+WP)	SPEC2000	MediaBench	SPEC2000	MediaBench
IC (32-Kbyte, 2-way)	22.53	17.60	17.47	16.96	16.83
DC (32-Kbyte, 4-way)	19.12	12.28	13.50	11.41	12.72

tion cache (32-Kbyte, two-way) implemented with way prediction, the DZS_D cache reduces total cache power consumption during a read access roughly 10 to 19 percent, and the DZS_S cache reduces it roughly 11 to 21 percent.

Table 5. DZS scheme's read power reduction.

	DZS_D		DZS_S	
P_{cache} (%)	SPEC2000	MediaBench	SPEC2000	MediaBench
IC (32-Kbyte, 2-way)	9.67	9.92	10.93	11.18
DC (32-Kbyte, 4-way)	19.18	15.75	21.61	17.95

Our experimental results show that while retaining the same stability and performance as a conventional cache, the DZS_D cache reduces the average cache read power up to 19 percent with an 8.4 percent area penalty. The DZS_S cache reduces cache area by 7 percent and reduces average cache read power up to 21 percent without impairing cache performance, but with a stability degradation of 7.1 percent. With the advent of 64-bit architectures, 0 bits will likely become more prevalent. This implies that the DZS scheme will be still more efficient in reducing cache read power.

References

1. J. F. Edmondson et al., "Internal Organization of the Alpha 21164, a 300-MHz 64-Bit Quad-Issue CMOS RISC Microprocessor," *Digital Technical J.*, vol. 7, no. 1, 1995, pp. 119-135.
2. J. Montanaro et al., "A 160 MHz, 32b 0.5W CMOS RISC Microprocessor," *Proc. 43rd IEEE Int'l Solid-State Circuits Conf.* (ISSCC 96), IEEE Press, 1996, pp. 214-215, 447.
3. C. Lee, M. Potkonjak and W.H. Mangione-Smith, "MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems," *Proc. 30th Int'l Symp. Microarchitecture* (Micro 30), IEEE Press, 1997, pp. 330-335.
4. B. Amrutur and M. Horowitz, "Techniques to Reduce Power in Fast Wide Memories (CMOS SRAMs)," *Proc. IEEE Symp. Low Power Electronics*, IEEE Press, 1994, pp. 92-93.
5. D.C. Burger and T.M. Austin, "The SimpleScalar Tool Set, Version 2.0," *Computer Architecture News*, vol. 25, no. 3, June 1997, pp. 13-25.
6. J.L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd ed., Morgan Kaufmann, 1995.
7. N. Azizi et al., *Asymmetric-Cell Caches: Exploiting Bit Value Biases to Reduce Leakage Power in Deep-Submicron, High-Performance Caches*, tech. report TR-01-01-02, ECE Computer Group, Univ. of Toronto, 2002.
8. J.H. Tseng and K. Asanovic, "Energy-Efficient Register Access," *Proc. 13th Symp. Integrated Circuits and Systems Design*, IEEE Press, 2000, pp. 377-382.
9. B.K. Park, Y.S. Chang, and C.M. Kyung, "Confirming Inverted Data Store for Low Power Memory," *Proc. Int'l Symp. Low Power Electronics and Design* (ISLPED 99), IEEE Press, 1999, pp. 91-93.
10. P. Shivakumar and N.P. Jouppi, *CACTI 3.0: An Integrated Cache Timing, Power, and Area Model*, Compaq WRL research report, 2001/2.
11. M. Ukita et al., "A Single-Bit-Line Cross-Point Cell Activation (SCPA) Architecture for Ultra-Low-Power SRAM's," *IEEE J. Solid-State Circuits*, vol. 28, no. 11, Nov. 1993, pp. 1114-1118.
12. K. Inoue, T. Ishihara, and K. Murakami, "Way-Predicting Set-Associative Cache for High Performance and Low Energy Consumption," *Proc. Int'l Symp. Low Power Electronics and Design* (ISLPED 99), IEEE Press, 1999, pp. 273-275.

Yen-Jen Chang is an assistant professor in the Department of Computer Science at National Chung-Hsing University, Taiwan. His research interests include computer architecture, low-power VLSI design, and embedded-system and SoC design. Chang has an MS in computer science and information engineering from Chung-Yuan Christian University, and a PhD in computer science and information engineering from National Taiwan University. He is a member of the IEEE.

Feipei Lai is a professor in the Department of Computer Science Information Engineering and the Department of Electrical Engineering at National Taiwan University. He is also the director of the Computer and Information Network Center at National Taiwan University. His research interests include low-power computing, computer architectures, and VLSI SoC design. Lai has a BSEE from National Taiwan University and an MS and a PhD, both in computer science, from the University of Illinois at Urbana-Champaign. He is a senior member of the IEEE.

Direct questions and comments about this article to Yen-Jen Chang, Dept. of Computer Science, National Chung-Hsing University, No. 250, Kuo Kuang Rd., Taichung, 402 Taiwan; ychang@cs.nchu.edu.tw.