

Performance and Energy Trade-offs of Bitline Isolation in Nanoscale CMOS Caches

Se-Hyun Yang and Babak Falsafi

Computer Architecture Lab at Carnegie Mellon (CALCM)

Carnegie Mellon University

{sehyun, babak}@ece.cmu.edu

<http://www.ece.cmu.edu/CALCM>

Abstract

High-performance cache architectures always pull up the bitlines in all cache subarrays to hide the bitline charging latency prior to a cache access. Unfortunately, such architectures lead to significant bitline discharge in unaccessed subarrays in nanoscale CMOS caches and waste power. Recent proposals advocate *bitline isolation* to reduce bitline discharge in unaccessed subarrays by turning off precharge devices located between the supply voltage and bitlines in these subarrays. Many of these proposals tacitly assume that on-demand precharging of isolated bitlines can be overlapped with address decoding and hidden from the cache access latency. Moreover, they assume that the energy overhead of switching precharge devices is insignificant.

In this paper, we carefully investigate the performance and energy impact of bitline isolation and show that these tacit assumptions are not true in the following senses: (1) Precharging isolated bitlines cannot be overlapped with address decoding and lies on the cache access critical path in both current and future CMOS technologies. Therefore, early precharging is necessary for isolated bitlines to avoid prohibitively impacting cache access latency. (2) The energy overhead of switching the precharge devices in current (i.e., 130nm) CMOS technologies is prohibitively high and nearly offsets the bitline isolation's energy saving. Fortunately, the switching overhead decreases with decreasing feature sizes because the precharging devices and bitlines tend to shrink for smaller feature sizes.

1 INTRODUCTION

Modern first-level caches account for an increasingly significant fraction of switching energy in high-performance wide-issue out-of-order processors. These caches typically pull up the bitlines in all subarrays statically or on every clock cycle to minimize cache access latency [5]. Unfortunately, current trends towards scaling down the transistor threshold voltage are resulting in a significant discharge through the statically pulled-up bitlines

even in cache subarrays that are not accessed. The large amount of bitline discharge is exacerbated by the trend towards using multiported caches (e.g., data caches in superscalar and data/instruction caches in SMT), because the number of bitlines and the discharge is proportional to the number of ports.

Bitline isolation is a technique that can be used to reduce the amount of energy discharged through the bitlines [7][11][12]. Instead of pulling up the bitlines statically or on every clock cycle, bitline isolation turns off the unaccessed subarrays' precharge devices and reduces energy discharged through the bitlines. The Alpha 21164's on-chip L2 cache exploits the idea of bitline isolation with on-demand subarray precharging [3]. Partial address decoding identifies the accessed subarrays and then the bitlines in the unaccessed subarrays are isolated. The isolated bitlines are precharged on demand when they are accessed and such delayed precharging incurs an extra cycle of delay to the cache access latency, which may be acceptable for L2 caches. On the other hand, resizable cache [12] utilizes bitline isolation periodically (regardless to the demand of the moment) where bitline isolation for each subarray is enabled/disabled once in a million instructions based on the utilization of the cache. Due to infrequent switching of precharge devices, the performance and energy overhead can be hidden and amortized into the overall performance and energy dissipation. However, such infrequency might end up with slow adaptation to the changes in cache size requirement and lost opportunity.

In this paper, we investigate bitline isolation with on-demand bitline precharging applied to performance-critical L1 caches. A number of recent proposals investigate bitline isolation with on-demand precharging but they tacitly assume that this technique can be performed without performance or power implications [7][11]. In this paper, we closely investigate and quantify the performance and power trade-offs of bitline isolation with on-demand bitline precharging. The primary contributions of this paper are:

- Performance Impact:** Precharging for isolated bitlines cannot be overlapped with cache's other sub-operations such as address decoding and lies on the cache access critical path both for current and future nanoscale CMOS technologies. On cache access, isolated bitlines must be pulled up and our simulation results show that the delay that guarantees to fully charge isolated bitlines is longer than that of address decoding. Therefore, early precharging is necessary for isolated bitlines to avoid delaying cache accesses. Early precharging may require accurate and timely prediction of accessed subarrays.
- Energy Impact:** The energy overhead of switching the precharge devices to enable or disable bitline isolation is so significant in current CMOS technologies that it nearly offsets the energy saving achieved by isolating bitlines. Fortunately, the switching overhead decreases as feature sizes decrease, as the precharge devices and bitlines shrink with decreasing feature sizes. Therefore energy saving of bitline isolation increases in the future generation circuit technologies. In 70nm CMOS technology, bitline isolation with perfect subarray prediction (for early precharging) reduces 74% of the bitline discharge and 31% of energy in 32K 2-way i-caches with no performance impact.

The rest of the paper is organized as follows. In Section 2, we review the cache access procedure and the major sources of power dissipation in a cache access. Section 3 describes the bitline isolation technique and its implications for power and performance. In Section 4, we present and analyze experimental results. In Section 5, we present conclusions.

2 BITLINE PRECHARGING IN CACHES

To optimize for cache access speed, cache designers divide the array of blocks into multiple subarrays of SRAM cell rows, each containing one or more cache blocks [10]. Before a cache access begins, all the subarrays' bitlines are precharged, and the cache access address is supplied to the address decode logic. Upon completion of the decode, the decode logic activates the wordline corresponding to the selected row. SRAM cells on the row read their values out onto the precharged bitlines (in the case of a read) or write the values through the bitlines into the cells (in the case of a write) by discharging the bitlines.

After reading or writing the bitline values, the voltage on the two bitlines must be equalized by precharging to the processor's supply voltage. Figure 1 depicts a typical 6-T SRAM cell with precharge devices. To achieve a high clock speed and low cache access latency, high-performance caches overlap the bitline precharging time with other operations, such as address decode or output drive, and hide the entire bitline precharging latency of each cache access. As such,

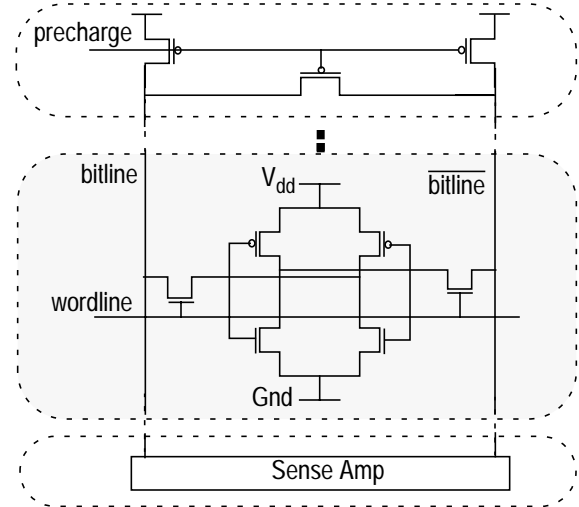


FIGURE 1: 6-T SRAM cell and precharge logic.

high-performance caches precharge *all* the subarrays every cycle irrespective of the accessed subarray.

Bitline precharging is achieved by either clocking the precharge devices every cycle (clocked precharging) or statically turning them on all the time (static pullup) [5]. In this paper, the base cache design assumes a static pullup for precharging the bitlines. The static pullup scheme has the advantage that it does not require a heavily loaded precharge clock signal. In addition, clocked precharging requires precise timing on the precharge clock, which is often extremely difficult to engineer. Therefore, recent processors such as the Alpha 21264 [6] advocate the static pullup scheme for their performance-critical L1 caches.

Because the bitlines are charged on every cycle, bitline activity is the key source of energy dissipation in high-performance caches [8]. There are two access scenarios for each subarray. First, as we described earlier, if a subarray is accessed, the precharged bitlines of the subarray discharge and sense amps read the voltage differential between the two bitlines. Sense amps in high-performance caches are designed to reliably sense a small voltage differential to optimize cell read time, as such bitlines are discharged 0.1 to 0.2V during a read access. Second, even without an active cache access, the bitlines are discharged by the large bitline leakage in current and future generation circuit technologies. In either case, the energy associated with pulling up the discharged bitlines is significant.

3 BITLINE ISOLATION

As we have seen in Section 2, maintaining charged bitlines for all subarrays requires a significant amount of energy in high-performance nanoscale caches. A conceptually simple solution is to not precharge all subarrays but instead to precharge only the accessed subarrays. The bitlines of the other subarrays are isolated from the processor's supply voltage

by turning off precharge devices. The precharge devices turned back on, only when there is an access to the subarray. We call this technique bitline isolation.

Bitline isolation with on-demand bitline precharging is first used in the L2 cache of the Alpha 21164 [3] as an extension of clock gating. To reduce the capacitive load associated with clock distribution in L2, the Alpha 21164 predecodes the access address and enables only the relevant subarrays. Predecoding the address allows the conditional clock generation circuitry to force the majority of its subarrays into a state where their bitlines and sense amps are frozen to conserve power. The bitlines in these subarrays are isolated from the supply voltage. Precharging for the relevant subarrays is delayed until they are identified. Such delayed precharging incurs an extra cycle delay for a cache access, which may be acceptable for L2 caches. Moreover, conditional clocking with 21164’s circuit process saves energy a lot larger than the overhead in switching precharge devices. On the other hand, the resizable cache technique [12] is recently proposed to utilize periodic bitline isolation regardless to the demand at the moment, where bitline isolation for each subarray is enabled or disabled once in a million instruction execution. Due to infrequent switching of precharge devices, the performance and energy overhead of bitline isolation can be amortized into the overall performance and energy dissipation. However, such infrequency might end up with slow adaptation to the changes in cache size requirement and lost opportunity.

Recently, a number of proposals advocate the bitline isolation technique with on-demand precharging from the perspective of bitline discharge in performance-critical L1 caches [7][11]. However, these proposals lack detailed analysis of the timing and energy aspects of bitline isolation. Moreover, one of them suggests without detailed analysis that on-demand precharging of isolated bitlines without prediction can be used with no performance impact. In the following sections, we investigate the performance and power implications of bitline isolation with on-demand precharging of isolated bitlines in the context of high-performance nanoscale cache design.

3.1 Performance Impact of Bitline Isolation

In this section, we study the performance implications when applying bitline isolation and precharging isolated bitlines on demand with partial address decoding. When there is no cache access, all bitlines are isolated from the processor’s supply voltage and the isolated bitlines approach steady state by slowly discharging the stored energy. At steady state, isolated bitlines may be fully discharged (or floating at the steady state) and there is little energy discharged from the bitlines. On a cache access, a part of the address is decoded to identify relevant subarrays. Isolated bitlines in the subarrays must be pulled back up before a cell read

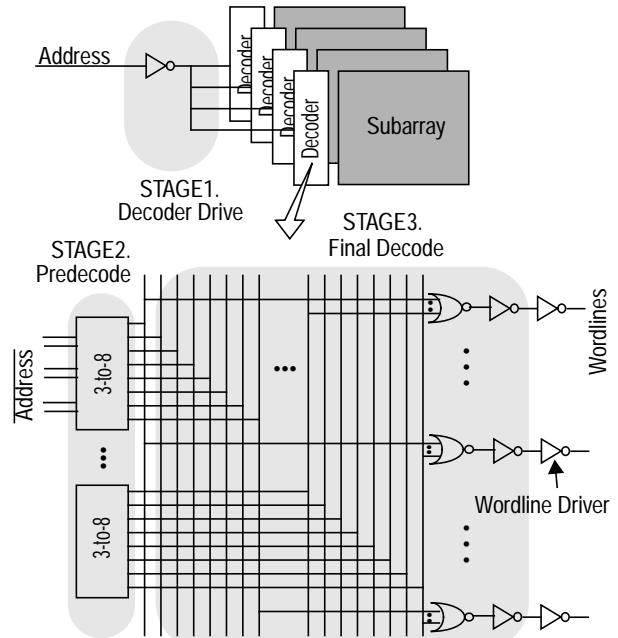


FIGURE 2: Cache decoder architecture.

begins. The delay for this process can be hidden if it completely overlaps with the delay of full address decoding for wordline assertion.

Figure 2 show the cache decoder architecture for full address decoding. We assume that our decoding logic is similar to that of the CACTI simulator’s model without loss of generality [10]. One decoder is associated with each data array and one driver drives all the data array decoders. The decoder in Figure 2 contains three major sources of delay, each of which corresponds to one of three decoding stages. In the first stage, the cache access address is fed into the decoders in the data arrays. The second stage divides the address into a number of three bit blocks and generates 1-of-8 codes via 3-to-8 block. These 1-of-8 codes are combined using NOR gates in the third stage. Each NOR gate takes an input from each decoder block to identify the accessed row.

Partial address decoding for subarray identification still needs the first stage (decoder driver) of the full address decode. We can assume that depending on the number of subarrays in a cache, one or more of the 3-to-8 decoder blocks are utilized to identify relevant subarrays. Therefore, partial address decoding shares the first and second stages with full address decoding. In the best case relevant subarrays can be identified right before the third stage begins.

To hide the latency of partial address decoding and precharging for isolated bitlines, we need to overlap the third stage of full address decoding with precharging for the isolated bitlines. Bitline precharging on an active cache access in conventional caches is overlapped with other sub-operations such as address decode or output drive, because a cell

read lowers bitline voltages only a small amount (0.1 to 0.2V), thus the latency of pulling up these bitlines is low. However, if a bitline is isolated and not recharged every cycle, the bitline may fully discharge. In this case, the latency for pulling up the isolated bitlines may exceed the full address decoding latency.

The bitline precharging delay depends on several factors including the size of precharge devices and subarray size. First, we can reduce precharging delay by increasing the size of the precharge devices. However, with the static pull-up scheme, the precharge devices are always on, fighting against the bitline discharge for the bitlines that are moving low for a cell read. We cannot increase the size of precharge devices indefinitely because larger devices imply smaller and slower discharges on a cell read, which may result in increased cache access latency. Moreover, larger precharge devices require larger switching energy and this overhead may offset the energy savings achieved by bitline isolation.

The precharging delay decreases with the subarray size, because the bitline length and capacitive load of the bitlines is reduced. As process technology evolves, subarray sizes and bitline lengths tend to shrink. Because SRAM cells in the unaccessed rows exhibit larger leakage with a smaller feature size, the effective voltage differential in the accessed cells decreases, requiring fewer cells to be attached to the bitlines to reduce leakage. However, reducing subarray size increases the number of subarrays in a cache, in turn, increasing cache area and routing delay. Moreover, a larger number of subarrays require more bits from a cache access address to identify the relevant subarrays by partial address decoding, increasing the delay required for the partial address decoding.

In Section 4, we will see that even with aggressive feature sizes and subarray sizes, the precharging delay for fully discharged bitlines is always longer than the delay for address decode, therefore the delay for isolated bitline pullup is exposed on the cache access critical path.

3.2 Impact on Energy Dissipation

Bitline isolation cuts off the paths from supply voltage to ground through bitlines by turning off the precharge devices (Figure 1), and reduces the energy dissipated through the bitlines when there is no access. After turning off the precharge devices, the energy stored in the isolated bitlines is discharged and the energy discharge rate decreases with time. After a while, the isolated bitlines reach steady state where there is no more energy discharge through the bitlines. The time required for the bitlines to reach steady state is on the order of 10 to 100ns depending on the circuit technology. When there is an access on these bitlines, they must be pulled back up, which requires extra energy.

Feature Size (nm)	180	130	100	70
Supply Voltage (V)	2.0	1.7	1.3	1.0

Table 1: Experimented Circuit Parameters.

The major energy overhead involved in bitline isolation is the energy to switch the precharge devices. Because these precharge devices are typically an order of magnitude larger than cell transistors, precharge devices require significant energy to turn on and off. Moreover, switching the precharge devices creates an overshoot in the voltage level of the bitlines before they start discharging. This overshoot results in a larger energy discharge than would occur with the precharge devices on.

In the next section, we show that the energy overhead is quite large in the current circuit technology and it may offset the energy saving of bitline isolation. However, the overhead is decreasing as circuit technology evolves and the overhead beyond 70nm technology is insignificant.

4 RESULTS

In this section, we present empirical results on the performance and energy trade-offs for the bitline isolation technique and compare them to a conventional cache design. The performance and energy impact of bitline isolation depend highly on the circuit technology. Therefore, we investigate bitline isolation with different circuit technologies and with different supply voltages corresponding to each circuit technology (Table 1). In this evaluation, we assume a 32-byte block size for 32K 2-way set-associative L1 caches. To avoid the issues related to instruction scheduling complicated by the data cache’s variable hit latency, we investigate instruction caches only.

To perform circuit simulations, we modified CACTI 3.0 [10] to produce the delays and energy dissipations of all the subcomponents in a cache access. We run SPICE simulations with Berkeley Predictive Technology Models[1] to estimate the delays and energy dissipations of the isolated bitlines. For architectural simulations, we test 12 applications including *ammp*, *applu*, *apsi*, *compress*, *gcc*, *jpeg*, *m88ksim*, *su2cor*, *swim*, *tomcatv*, *vortex* and *vpr* from the SPEC2000 and SPEC95 benchmark suites using SimpleScalar 3.0. Table 2 shows the base system configuration.

4.1 Performance Evaluation

Table 3 shows the delays for the full address decoding and precharging for isolated bitlines for different subarray sizes and feature sizes. For the full address decoding, we indicate the delays for all the three stages of the process (Section 3.1).

As discussed in Section 3.1, we assume partial address decoding that initiates precharging for the isolated bitlines

Instruction issue/decode	8 per cycle
Reorder buffer entries	128
LSQ size	64
Branch predictor	combination
Base L1 i- and d- cache	32K, 2-way; 2 cycle lat
L2 unified cache	512K, 4-way; 12 cycle lat
Memory access latency	50 + 4 cycles per 8 bytes
Writeback buffer entries	8

Table 2: Base system configuration parameters.

shares the first two stages of full address decoding. Therefore, by comparing the delays for the final decoding stage and the bitline precharging, this table shows that isolated bitline precharging with partial address decoding cannot be fully overlapped with full address decoding. The maximum time required to precharge fully discharged bitlines is always more than twice the final decoding stage. Therefore, on-demand precharging for the isolated bitlines delay bitline precharging and are not applicable to high-performance cache design. The average performance impact of this delayed precharging technique is 9% for the tested applications.

As discussed earlier, with a smaller subarray size, the first stage of address decoding, essentially the process of routing the access address, takes longer and accounts for the largest portion of the full address decoding. Regardless of the actual implementation of the address decode, the first stage must be shared by the full and partial address decoding.

4.2 Energy Evaluation

In this section, we look at the average power profiles based on the interval between two accesses on a cache subarray. For static pullup of the conventional caches, the same average power is consumed regardless of the subarray access

Sub-array size	Feature size (nm)	Address Decode (ns)			Discharged Bitline Precharge (ns)
		Decode Drive	Pre Decode	Final Decode	
1KB	180	0.51	0.28	0.15	0.39
	130	0.46	0.27	0.16	0.31
32-rows	100	0.36	0.21	0.13	0.24
	70	0.25	0.15	0.09	0.16
4KB	180	0.16	0.20	0.18	0.50
	130	0.11	0.15	0.13	0.36
128-rows	100	0.088	0.11	0.10	0.28
	70	0.062	0.077	0.07	0.19

Table 3: Performance Impact.

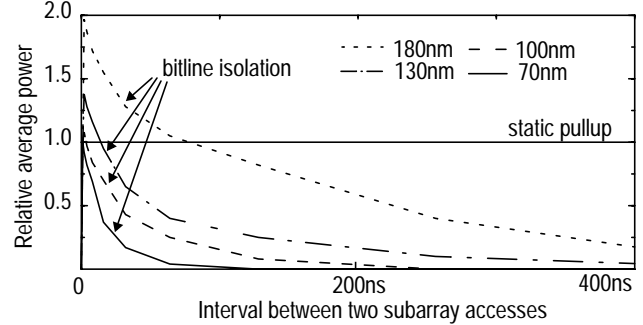


FIGURE 3: Average power profiles.

interval. However, the average power dissipation of the isolated bitlines depends on the access interval. If the bitlines are accessed soon after they are isolated, bitline isolation does not save much power, because the energy discharge from the isolated bitlines still remains large and it also incurs large overhead for switching the precharge devices.

We present the average power profiles for the caches with 1KB subarrays in Figure 3. The average power dissipation is normalized to the average power of static pullup for each process. This figure clearly shows that the overhead of switching the precharge devices widely varies across the circuit technologies. In 180nm technology, this overhead is up to 195% of the average power of static pullup. Moreover, the isolated bitlines reach steady state more than 500ns after bitlines are isolated. However, the energy overhead and falling time dramatically decrease for smaller feature sizes, mainly because the precharge device size decreases. In 70nm technology the overhead melts away so quickly that the amount of the overhead is negligible. Therefore, the bitline isolation in 70nm technology is more efficient than in the current generation technologies.

4.3 Energy Savings with Bitline Isolation

As we have seen in Section 4.1, the bitline isolation technique increases cache access latency, and the on-demand precharging for the isolated bitlines delays precharging and slows down the program executions by 9%.

To reduce the performance impact, we apply two mechanisms to select subarrays before the relevant subarrays are identified. First, as a limit study, we assume that we have a perfect and timely subarray prediction mechanism for early precharging. The bitlines are isolated after every cache access and the perfect prediction pulls up the bitlines on time without incurring an extra cycle of delay for a cache access. Perfect subarray prediction incurs no performance impact but creates a large number of switchings of the precharge devices. Therefore, for the current circuit technologies such as 180nm and 130nm, we expect a large energy overhead as discussed in Section 4.2. However, for 70nm technology or beyond, we expect apparent energy savings because the overhead is almost negligible.

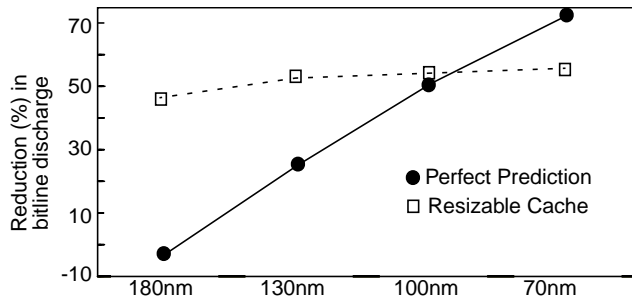


FIGURE 4: Average reduction in bitline discharge.

Second, we apply the resizable cache technique [2][12] as an implementable example for the subarray selection mechanism. In resizable caches, the enabled subarrays are statically pulled up while disabled subarrays isolate their bitlines from the supply voltage. By switching the precharge devices only once in a million instructions, resizable caches virtually remove the energy overhead of switching the precharge devices. However, resizable caches' coarse granularity result in sub-optimal energy savings and a performance impact, which we limit to 2% in this study.

Figure 4 shows the reduction in the amount of energy discharged through the bitlines, averaged over all 12 applications with both selection mechanisms across different circuit technologies. This figure first shows that perfect prediction's energy savings vary widely depending on the circuit technology. In the 70nm technology, bitline isolation with perfect subarray prediction reduces the energy discharged through bitlines by 74% (31% in cache energy), while perfect prediction in 180nm CMOS consumes 3% more than conventional static pullup. However, resizable caches show almost constant energy savings across the technologies. Resizable caches reduces 56% of bitline discharge and 23% of energy in the cache. Because the resizing interval is as long as one million instructions, the isolated bitlines stay at steady state most of the time regardless of the technology. Moreover, the number of switchings for the precharge devices is much smaller than the case of perfect prediction.

5 CONCLUSIONS

In this paper, we have investigated the performance and power trade-offs of bitline isolation with on-demand precharging for isolated bitlines, to reduce bitline discharge of nanoscale CMOS cache memories.

Our primary contributions are: (1) Precharging for isolated bitlines cannot be overlapped with address decoding and lies on the cache-access critical path both for current and future nanoscale CMOS technologies. Therefore, isolated bitlines require early precharging with accurate and timely prediction of accessed subarrays. (2) The energy overhead for enabling/disabling bitline isolation is significant in current circuit generations and it reduces the effectiveness of bitline isolation. However, the overhead decreases as circuit process evolves, resulting in larger energy savings in future generation circuit technologies. In 70nm CMOS, bitline isolation with perfect subarray prediction (for early precharging) reduces 74% of the bitline discharge and 31% of energy in 32K 2-way i-caches with no performance impact.

REFERENCES

- [1] <http://www-device.eecs.berkeley.edu/ptm/>.
- [2] D. H. Albonese. Selective cache ways: On-demand cache resource allocation. In *Proceedings of the 32nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 32)*, pages 248–259, Nov. 1999.
- [3] B. J. Benschneider, A. J. Black, and et. al. A 300-mhz 64-b quad-issue cmos risc microprocessor. In *IEEE Journal of Solid-State Circuits*, pages 1203–1214, Nov. 1995.
- [4] D. Brooks, V. Tiwari, and M. Martonosi. Watch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 83–94, June 2000.
- [5] A. Chandrakasan, W. J. Bowhill, and F. Fox. *Design of High-Performance Microprocessor Circuits*. IEEE Press, 2001.
- [6] B. Gieseke and et. al. A 600-mhz superscalar risc microprocessor with out-of-order execution. In *ISSCC Digest of Technical Papers*, pages 176–177, Feb. 1997.
- [7] S. Heo, K. Barr, M. Hampton, and K. Asanovic. Dynamic fine-grain leakage reduction using leakage-biased bitlines. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, May 2002.
- [8] M. B. Kamble and K. Ghose. Analytical energy dissipation models for low power caches. In *Proceedings of the 1997 International Symposium on Low Power Electronics and Design (ISLPED)*, pages 143–148, Aug. 1997.
- [9] S. Manne, A. Klauser, and D. Grunwald. Pipline gating: Speculation control for energy reduction. In *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pages 132–141, June 1998.
- [10] S. J. E. Wilton and N. P. Jouppi. An enhanced access and cycle time model for on-chip caches. Technical Report 93/5, Digital Equipment Corporation, Western Research Laboratory, July 1994.
- [11] S.-H. Yang and B. Falsafi. Gated precharging: Using temporal locality of subarrays to save deep-submicron cache energy. In *Proceedings of Workshop on Complexity-Effective Design held in conjunction with the 29th International Symposium on Computer Architecture (ISCA 29)*, May 2002.
- [12] S.-H. Yang, M. D. Powell, B. Falsafi, and T. N. Vijaykumar. Exploiting choice in resizable cache design to optimize deep-submicron processor energy-delay. In *Proceedings of the Eighth IEEE Symposium on High-Performance Computer Architecture*, pages 151–161, Feb. 2002.