Adaptive Mode-Control: A Low-Leakage, Power-Efficient Cache Design

Huiyang Zhou, Mark C. Toburen, Eric Rotenberg, Thomas M. Conte

Department of Electrical & Computer Engineering

North Carolina State University

Adaptive Mode-Control: A Low-Leakage, Power-Efficient Cache Design

Abstract

With the advent of deep sub-micron circuit technology, the ratio of static-to-dynamic power in on-chip memories has become an increasingly important issue. At the circuit level, designers propose low-leakage SRAM operation modes (i.e., sleep mode or standby mode) and at architecture level, there are increasing interests in how to efficiently integrate such features into the design. In this paper, we propose an approach to low leakage on-chip cache design called Adaptive Mode-Control (AMC) that adaptively controls the operation mode of individual cache lines in the data store. By maintaining the tag store information, our method has the ability to track the miss rate with respect to the ideal miss rate (the miss rate with conventional cache) so as to control the overall performance. Our simulations show an average of 74% I-cache lines and 50% D-cache lines can be put into sleep-mode with an average impact on IPC of 1.6%.

1. Introduction

Power dissipation is becoming an increasingly important constraint on high performance processor design. Of particular concern is the rapid increase in the ratio of static-to-dynamic power in on-chip cache structures due to the scaling properties of deep sub-micron technologies. Borkar [2] estimates that with each new processor generation, leakage current and leakage power increase by a factor of 7.5 and 5.0, respectively. On-chip caches consume a significant percentage of total die area, and therefore power, especially in high performance embedded processors. For example, 60% of the StrongARM embedded processor's die area is dedicated to on-chip cache structures [1]. This makes the memory subsystem a dominant source of power dissipation. Since the majority of those devices are turned off each clock cycle, an increasingly large portion of power is the static power generated primarily through the sub-threshold leakage. With this inevitable increase in static-to-dynamic power ratios, and with ever increasing on-chip cache size, minimizing the effects of sub-threshold leakage at the architectural level is increasingly important.

Recently, two approaches have been proposed to aim at this goal. An architectural mechanism called the DRI cache and a circuit technique called *Gated-Vdd* [4, 5] are proposed to turn off (i.e., gate the SRAM cells from either or both of the supply nodes, similar to the SRAM sleep mode) large portions of the L1 instruction cache (I-cache). The basic idea is to compare the current miss rate with a statically determined threshold (called the miss bound) to either increase or decrease the effective cache size by turning on/off part of the I-cache. To achieve the ideal result, the static threshold needs to be pre-tuned since the static cache miss bound has a different impact on performance for different applications. Also, DRI limits the granularity of the amount of the cache lines that can be turned off because the size of the cache resizing is based on the current effective cache size.

Kaxiras, et al. [6], proposes a mechanism called *Cache-line Decay* that turns off individual cache lines of the L1 data cache (D-cache) if they have not been accessed for a predetermined, static time interval. This approach exploits the finer granularity by examining individual cache lines but requires a static turn-off interval to be set on an individual application basis to obtain the optimal results. This is illustrated in Figure 1, where the static time interval (same for I-cache and D-cache) is obtained for each benchmark with a performance degradation goal of 4% of the IPC. As can be seen from the figure, these thresholds vary widely across benchmarks.

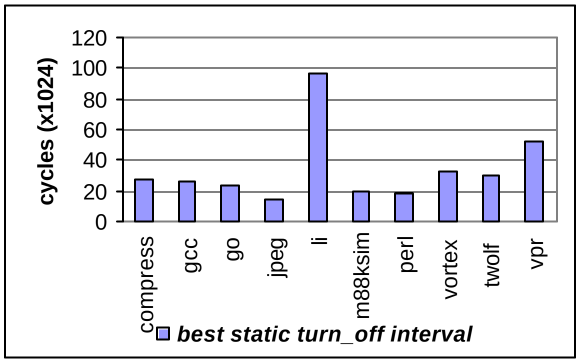

Figure 1. The static determined turn-off interval for each benchmark

In this paper, we propose a new approach that has the ability to *dynamically adjust* to the changing cache requirements of each individual program throughout its run-time. And the technique is equally applicable to both instruction and data caches. This approach, which we call *Adaptive Mode-Control* (AMC), similar to the cache line decay, controls the sleep mode of individual cache lines according to a *turn-off interval*. But the new important factor is that this interval is adjusted dynamically according to changes in the instruction and data working sets of a program as it proceeds through its execution.

AMC works by selectively powering down unused cache lines in the data store, thus saving the static power. Unlike other approaches, the tag store information is always maintained so that the impact of powering down data store cache lines on performance can be continually monitored. Two performance indicators are monitored throughout execution. One is the *ideal miss rate* resulting from tag mismatches. The other is the *sleep miss rate*, which is a miss due to the cache line in data store has been in sleep mode although the tag information matches. AMC monitors the relationship between these two miss rates during the course of execution and determines how the turn-off interval should be modified. Our results show that for SPECint95 and selected SPECint2000 benchmarks, we can turn off an average of 74% of I-cache lines and 50% of D-cache lines during program execution at an average performance cost of only 1.6% in IPC.

The remainder of this paper is organized as follows. Section 2 describes the architecture of AMC. Circuit level design issues are discussed in Section 3. Section 4 discusses two potential mechanisms for control of the turn-off interval. The simulation methodology and results are presented in Section 5. Section 6 discusses the related work, and Section 7 concludes the paper and discusses future work.

2. Adaptive Mode-Control Cache Design

There are three fundamental observations that drive the concept of low-leakage caches and the AMC approach to low-leakage cache design. First, a very small number of devices in the cache data array are active each clock cycle. Second, the majority of devices used to implement a cache structure are in the data array. Accordingly, the majority of the devices in the cache dissipate only static power each cycle, leading to increasingly high static-to-dynamic power ratios in large caches with deep sub-micron technologies. Third, the size of instruction and data working sets varies within and across applications. Based on these three observations, the ideal cache should be able to turn off its unused blocks by placing them into sleep mode (i.e., gating the power supply to the SRAM cells), to minimize the static power while maintaining its performance. The AMC architecture and adaptive mode control algorithm proposed below sets out to achieve this goal for both I- and D-caches.

2.1. AMC Architecture

In order to make efficient use of a sleep-mode SRAM design, we need the capability to monitor cache line accesses on a per line basis and to determine the time at which it is most beneficial to place a line into sleep-mode. In the AMC architecture, shown in Figure 2, we add a bank of counters, one per tag entry, to the tag store of the cache. We call these *Line Idle Counters* (LICs), as these counters monitor how long the cache line has not been accessed. An LIC is reset upon an access to the corresponding data line and is incremented if the data line has not been accessed for a given amount of time, which is called the *LIC update interval*. The *Mode-Control Logic* (MCL) compares the LIC value to the *turn-off interval* stored in the *Global Control Register* (GCR). If an LIC is greater than or equal to the turn-off interval, the MCL will place the corresponding data line into sleep-mode. Otherwise, active-mode will be maintained. The turn-off interval stored in GCR can be set via a compiler generated move instruction or by dynamic performance monitoring hardware. Both schemes are discussed in Section 4.

2.2. LIC Update Interval

In the AMC architecture, each LIC is updated at a fixed interval, called the LIC update interval. Smaller intervals capture the cache line access history at finer time granularity. As discussed in [6], finer granularity means that there are more chances for the cache line to be turned off since the LICs are compared with the GCR more often. Coarser granularity, on the other hand, results in reduced area and energy consumption costs in the LICs. A preliminary study in which we examined the effect of varying the LIC update interval showed that an interval of 2048 cycles provides a good power-performance trade-off within the range studied, and we use this interval length throughout the remainder of this work. Details of this analysis will be given in Section 5.2.5.

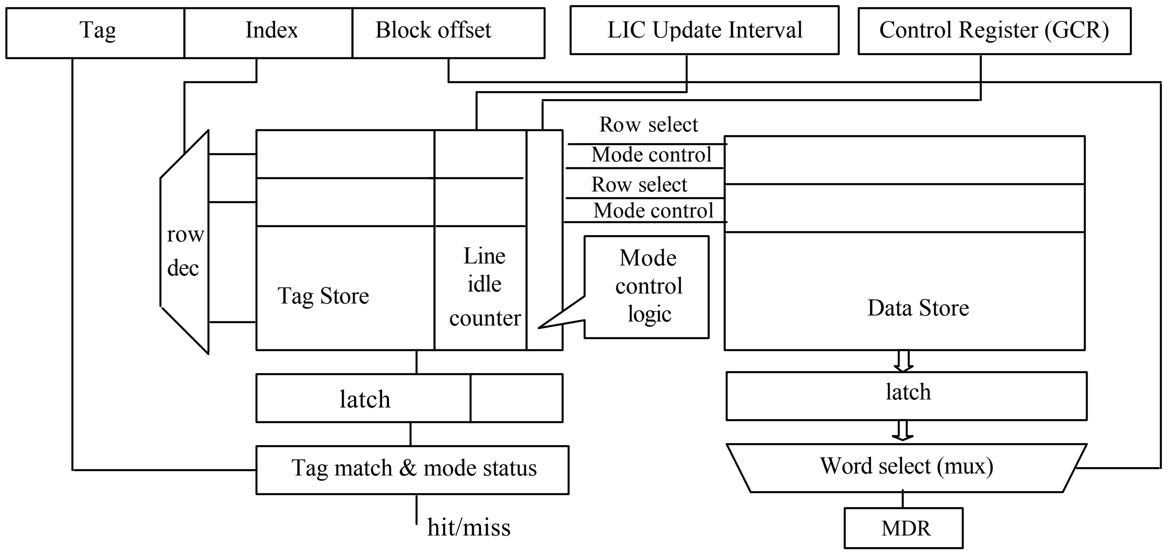

Figure 2. The AMC cache architecture (direct mapped)

2.3. Performance Impact

In cache design, the principle of temporal locality is used to determine the replacement policy. Temporal locality states that the most recently accessed blocks are the most likely to be accessed again in the near future. We make use of temporal locality by assuming that if we place a cache line into sleep-mode after it remain un-accessed for some time then it is highly probable that the next access to that line will be a conflict or capacity miss. If this assumption holds true, then there is no performance impact. However, if the assumption does not hold, we incur the penalty of sleep misses.

The number of sleep misses is primarily determined by the turn-off interval. Shorter turn-off intervals increase the aggressiveness of AMC, resulting in potentially high sleep miss rates. At the same time, shorter turn-off intervals also allow the MCL to put blocks into sleep mode more often, resulting in increased power savings. In Section 4, we propose two schemes for dynamically adjusting the interval as the cache requirements of the executing application change over time.

It is possible that there are applications whose working sets are large enough to warrant shutting off the AMC mechanism. This is because the turn-off interval in such applications is likely to be so small that it results in severe performance penalties. In such cases, we can turn off AMC functionality by zeroing out the GCR, telling the MCL to cease all AMC operations. This allows the cache to operate in active-mode at all times and achieve maximum performance, albeit without power savings.

3. Circuit Design Considerations

Recently researchers in the VLSI community have proposed several techniques for reducing the leakage power consumed in memory cells [3, 4, 8, 9, 10]. The implementation used in this work, which isolates the power and ground nodes from the storage cell, is shown in Figure 3. Two additional nodes, *virtual vdd* (vvdd) and *virtual gnd* (vgnd), are introduced, and the voltage at these two nodes is controlled by transistors Q1 and Q2, which are high-Vt or long-channel devices. When the circuit is in active-mode, both Q1 and Q2 are on and the circuit operates as usual. When in sleep-mode, Q1 and Q2 are turned off and the leakage current through the SRAM cell is reduced dramatically due to the transistor stacking effect [7]. Several variants on this implementation and the implications on power and performance are discussed in [4].

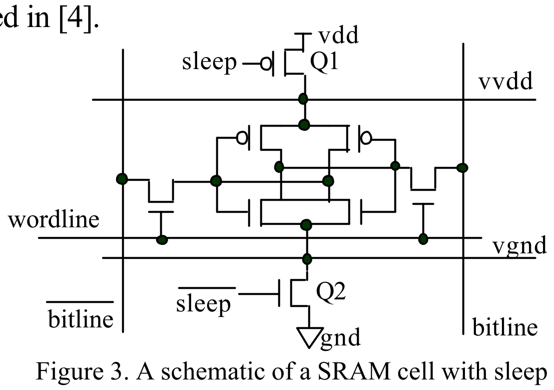

Figure 3. A schematic of a SRAM cell with sleep mode

An important issue that must be considered when dealing with sleep-mode memory is the stability of the stored data after a cell has been in sleep-mode for an extended period. When a cell is placed into sleep-mode, the vvdd and vgnd nodes become dynamic, potentially leading to metastability in the cell if it is isolated from the supply nodes long enough. In some architectural structures (e.g. I-caches) this is acceptable since the stored data is not critical to correct program execution or the data stored is not part of the architectural state of the processor. As long as the information can be regenerated, we can allow sleep-mode storage cells to remain turned off for arbitrarily long periods. However, there are some data values that must be guaranteed to be valid if requested (e.g. cache coherence information, locally modified shared data). A possible remedy is to refresh those cells that are in sleep-mode every N clock cycles to re-enforce the charge on the cell's internal nodes and ensure the validity of the data. The parity bit then can be used as an indicator of the validity of the data. Architectural features could then be added to support a new form of speculative execution so as not to stall the processor pipeline when sleep misses occur in I-cache. For sleep misses in the D-cache, such speculation is a natural extension to value prediction of load instructions. This would result in an interesting hybrid DRAM/SRAM design in which there is great potential for reducing leakage power dissipation.

4. Adaptive Mode-Control Schemes

A cache line's operational mode is determined by the MCL, which compares the corresponding LIC with the turn-off interval stored in the GCR. If the LIC is greater than or equal to the turn-off interval, the line will be placed into sleep-mode. If the turn-off interval is accurately tuned to application behavior, significant power reduction can be achieved at very little cost in performance.

We have designed the GCR so that it can be modified via dynamic hardware or compiler generated move instructions. In this paper, we study in detail the dynamic hardware approach and briefly discuss the potential and merits of the software approach.

4.1. Adaptive Mode-Control via Software

Since we have designed the GCR to be software visible, it is possible for the compiler to insert instructions to update the GCR based on static analysis of application behavior. Static determination of

I-cache behavioral characteristics can be obtained accurately for many fundamental programming constructs, especially loop structures, assuming the compiler is knowledgeable of the I-cache architecture. However, static analysis and characterization of D-cache accesses is much more difficult due to the dynamic nature of load/store address generation. Our expectation, however, is that adding static analysis support will likely increase AMC effectiveness, at least for the I-cache, since we do not have to pay the penalty of a GCR *warm-up period* or endure potential periods of *hysteresis* with the dynamic approach. A detailed treatment of a software approach is beyond the scope of this paper and is part of our continuing research.

4.2. Adaptive Mode-Control via Dynamic Hardware

In this approach, the miss rate is monitored dynamically to detect distinct phases of varying cache performance. Since the tag store is kept in active mode at all times, the ideal and real cache miss rates can be monitored via counters. The ideal miss rate is the miss rate resulting from tag mismatches, and the real miss rate is the ideal miss rate plus the sleep miss rate. Based on the difference between these two rates and the performance factor, the GCR will be updated accordingly. This scheme is shown in Figure 4. There are two additional inputs to the GCR update logic aside from the ideal and real miss counters. The first is the *end-of-sense-interval* (ESI) signal. The sense interval is a preset, fixed number of cycles that defines how often the GCR is updated. We will show in Section 5.2.5 that this interval actually has little impact on overall performance. The second is the *performance factor* (PF), which defines the degree to which sleep misses can be tolerated in exchange for power savings. The PF is expressed as a fraction of the ideal miss rate.

The GCR update algorithm is shown in Figure 5. In this algorithm, we update the GCR based on the ratio of the sleep miss rate over the ideal miss rate. When the sleep-to-ideal ratio is hovering around the PF fraction, the algorithm ensures we are not too sensitive in adjusting the GCR. Due to the dynamic nature of the algorithm, the sleep miss rate should eventually settle to the desired performance factor fraction. It should also be noted that the GCR needs to be greater than 1. If the GCR is less than or equal to one, then each cache block will be placed into sleep mode when MCL compares LICs with GCR, leading to unacceptable levels of sleep misses and performance degradation.

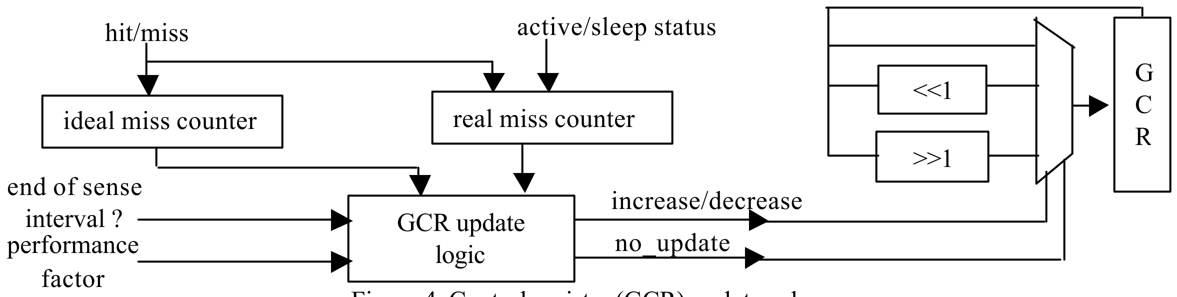

Figure 4. Control register (GCR) update scheme

```
if (sleep miss rate / ideal miss rate < 0.5 * PF)
{decrease GCR by shifting right one bit}
else if (sleep miss rate / ideal miss rate > 1.5 * PF)
{increase GCR by shifting left one bit}
else { no change }
```

Figure 5. The GCR update algorithm

5. Methodology & Results

In this section, we first describe our simulation environment including the microarchitectural configuration of our simulator and the benchmarks used in the simulation. Second, we present architectural simulation results. Then, we discuss our energy analysis of the AMC scheme at both the cache and chip level and present *Energy*Delay* results to demonstrate the effectiveness of AMC.

5.1. Simulation Environment

We developed a cache simulator that fully models the AMC architecture and integrated it into the out-of-order timing simulator in the Simplescalar toolset [16] to perform execution-based simulation. Table 1 shows the processor configuration. The entire SPECint95 benchmarks and two SPECint2000 benchmarks are used. Table 2 lists the input sets and all benchmarks were run to completion.

5.2. Performance Results

In this section we study the use of AMC on I- and D-caches separately and together. We then discuss how the GCR update algorithm adapts to program behavior and demonstrate the impact of different parameters on the AMC approach.

5.2.1 AMC I-cache Only

In an AMC I-cache, our goal is to exploit the temporal locality of instruction sequences to reduce static power. In our simulations, we studied a 64KB, 2-way set associative cache with a PF (performance factor) of ½ and a sense interval of 1 million cycles. The LIC update interval is 2048 cycles and GCR is saturated at 128 (i.e., the longest turn off interval, in cycles, is 2048*128 = 256K). The simulation results in Table 3 show the ideal IPC, real IPC, ideal miss rate, sleep miss rate, and the cache line turn-off ratio. Means (harmonic mean for IPC) are given in the last row.

Table 1. Processor configuration

Instruction cache	Fetch bandwidth: 2-way interleaved to fetch full cache block
	Size/assoc/replacement = 64kB/4-way, 2-way, 1-way(directed mapping)/LRU
	Line size = 16 instructions
	Miss penalty = 12 cycles
Data Cache	Size/assoc/replacement = 64kB/4-way/LRU
	Line size = 64 bytes
	Miss penalty = 14 cycles
Super-scalar Core	Re-order buffer: 64 entries
	Dispatch/issue/retire bandwidth: 4-way
	4 fully-symmetric function units
	Issue mem bandwidth: 4
Execution latencies	Address generation: 1cycle
	Memory access: 2 cycle (hit in cache)
	Integer ALU ops = 1cycle
	Complex ops = MIPS R10000 latencies

Table 2. Benchmarks

Benchmark	Input dataset	Instr. Count
Compress	Compress95.ref	24 million
Gcc	-O3 genrecog.i –o genrecog.s	117 million
Go	9 9	133 million
Jpeg	Vigo.ppm	166 million
Li	Test.lsp (queens 7)	202 million
M88ksim	-c < ctl.in (dcrand.big)	120 million
Perl	scrabble.pl < scrabble.in (dictionary)	108 million
Vortex	vortex.in (persons.250, bendian.*)	101 million
Twolf	Test	214 million
Vpr	net.in arch.in place.out dum.out *	1557 million

From the results in Table 3, we can see that the effect of sleep misses on IPC is very small. This is primarily due to the decoupled nature of the superscalar processor pipeline. Since instructions are queued prior to issue, the processor can tolerate a slight increase in I-cache miss rate without significantly impacting IPC. We can divide the benchmarks into two classes based on their respective

I-cache miss rates. The first class, comprised of *gcc*, *go* and *vortex*, has higher ideal miss rates (greater than 1%) and sleep-to-ideal miss rate ratios close to what is defined by the performance factor. The second class, including all remaining benchmarks, has small ideal cache miss rates and large sleep-to-ideal ratios since the GCR is saturate most of the time in these cases.

Table 3. Simulation results – AMC I-cache only

Benchmarks	IPC		I-cache miss rate		Cache line turn-off ratio
	Ideal	Real	Ideal_miss	Sleep_miss	
Compress	1.56	1.56	4.1×10^{-5}	1.27×10^{-4}	97.0 %
Gcc	1.84	1.80	1.60×10^{-2}	7.68×10^{-3}	60.2 %
Go	1.64	1.61	2.00×10^{-2}	1.00×10^{-2}	58.9 %
Jpeg	1.97	1.96	2.61×10^{-4}	5.62×10^{-4}	83.9 %
Li	2.19	2.19	1.02×10^{-5}	1.53×10^{-4}	83.2 %
M88ksim	1.74	1.74	2.14×10^{-4}	6.98×10^{-4}	78.6 %
Perl	1.91	1.91	9.85×10^{-4}	7.15×10^{-4}	70.3 %
Vortex	2.35	2.32	1.12×10^{-2}	6.46×10^{-3}	51.2 %
Twolf	1.23	1.23	2.17×10^{-4}	8.11×10^{-4}	75.9%
Vpr	1.60	1.60	1.38×10^{-5}	1.13×10^{-4}	85.7%
Average	1.748	1.739	4.89×10^{-3}	2.83×10^{-3}	74.5 %

5.2.2 AMC D-cache Only

In an AMC D-cache, our goal is to exploit the temporal locality of load and store data to reduce static power. In our simulations we studied a 64KB, 4-way set associative cache with the same AMC parameters used in the I-cache study in Section 5.2.1. The results are shown in Table 4.

Comparing Table 4 with Table 3, it can be seen that D-cache turn-off ratios are smaller in comparison to the I-cache ratios. This is primarily due to fewer localities in the data streams of most benchmarks relative to those of instruction streams. The benchmarks *compress*, *gcc*, *go*, *jpeg* and *vortex* all have high ideal miss rates and sleep-to-ideal ratios close to what is defined by the performance factor. The remaining benchmarks have relatively large sleep-to-ideal ratios with the saturate GCR. Overall, the D-cache mean turn-off ratio is around 50%, which shows there is still a big opportunity for static power reduction.

5.2.3 AMC I- and D-cache

In a system with AMC I- and D-caches, we expect a combined effect in static power savings but are primarily concerned with the increased cost in performance and dynamic power due to increased accesses to a unified L2 cache. Here we use the same cache and AMC parameters as in the two previous studies. The results are given in Table 5.

Table 4. Simulation results – AMC D-cache only

Benchmarks	IPC		D-cache miss rate		Cache line turn-off ratio
	Ideal	Real	Ideal_miss	Sleep_miss	
Compress	1.56	1.51	2.98×10^{-2}	8.34×10^{-3}	41.3 %
Gcc	1.84	1.82	7.51×10^{-3}	4.22×10^{-3}	51.9 %
Go	1.64	1.63	4.66×10^{-3}	2.53×10^{-3}	47.0 %
Jpeg	1.97	1.95	4.81×10^{-3}	1.73×10^{-3}	52.8 %
Li	2.19	2.18	1.32×10^{-4}	7.12×10^{-4}	36.4 %
M88ksim	1.74	1.74	3.17×10^{-4}	4.02×10^{-4}	63.8 %
Perl	1.91	1.91	3.22×10^{-4}	4.59×10^{-4}	66.1 %
Vortex	2.35	2.29	1.21×10^{-2}	5.37×10^{-3}	68.3 %
Twolf	1.23	1.23	8.26×10^{-4}	1.03×10^{-3}	48.9%
Vpr	1.60	1.57	7.99×10^{-3}	5.32×10^{-3}	26.0 %
Average	1.748	1.729	6.85×10^{-3}	3.01×10^{-3}	50.3 %

Table 5. Simulation results – AMC I- and D-caches

Bench-marks	IPC		I-cache miss rate		I-cache turn-off ratio	D-cache miss rate		D-cache turn-off ratio
	Ideal	Real	Ideal_miss	Sleep_miss		Ideal_miss	Sleep_miss	
Compress	1.56	1.51	4.08×10^{-5}	1.26×10^{-4}	97.0 %	2.97×10^{-2}	8.34×10^{-3}	41.3 %
Gcc	1.84	1.78	1.60×10^{-2}	7.09×10^{-3}	59.6 %	7.46×10^{-3}	4.49×10^{-3}	53.6 %
Go	1.64	1.60	2.00×10^{-2}	1.00×10^{-2}	58.9 %	4.66×10^{-3}	2.36×10^{-3}	46.5 %
Jpeg	1.97	1.95	2.62×10^{-4}	5.48×10^{-4}	83.8 %	4.81×10^{-3}	1.79×10^{-3}	54.6 %
Li	2.19	2.18	1.02×10^{-5}	1.49×10^{-4}	82.9 %	1.29×10^{-4}	7.10×10^{-4}	36.4 %
M88ksim	1.74	1.74	2.28×10^{-4}	7.65×10^{-4}	78.5 %	3.15×10^{-4}	3.99×10^{-4}	63.9 %
Perl	1.91	1.91	1.00×10^{-3}	7.12×10^{-4}	70.3 %	3.21×10^{-4}	4.60×10^{-4}	66.1 %
Vortex	2.35	2.27	1.13×10^{-2}	5.70×10^{-3}	49.5 %	1.21×10^{-2}	4.94×10^{-3}	66.9 %
Twolf	1.23	1.23	2.17×10^{-4}	8.09×10^{-4}	75.8%	8.25×10^{-4}	1.02×10^{-3}	48.9%
Vpr	1.60	1.57	1.38×10^{-5}	1.15×10^{-4}	85.7 %	7.99×10^{-3}	5.33×10^{-3}	26.0 %
Average	1.748	1.721	4.91×10^{-3}	2.60×10^{-3}	74.2 %	6.83×10^{-3}	2.98×10^{-3}	50.4 %

Comparing the simulation results in Tables 3, 4, and 5, it can be seen that for benchmarks such as *compress*, *jpeg*, *li*, *m88ksim*, *perl*, *twolf*, and *vpr*, the performance is equivalent to the case when only one of I- or D-cache is implemented using AMC. In the remaining benchmarks, the performance is slightly worse than that shown in Tables 3 and 4. The static power savings, as given by the turn-off ratio, is close to the sum of the individual AMC I- and D-cache savings. Overall, on average, the degradation in IPC is 1.54% with I- and D-cache turn-off ratios of 74.2% and 50.4%, respectively.

Note that the ideal miss rates of the I- and D-caches in Table 5 vary slightly from those in Tables 3 and 4. The reason lies within the context of execution-based simulation. When the I-cache sleep-to-ideal miss ratio is high, there are more opportunities for pipeline stalls. In such cases, the dynamic window size becomes smaller and there are fewer chances for cache misses that are result from miss-speculated paths. Therefore, the ideal miss rate may potentially increase when the sleep miss rate decreases.

5.2.4 Fixed GCR vs. Adaptively Updated GCR

As working set sizes vary within and across application programs, the hardware-based scheme of Section 4.2 provides a mechanism to dynamically adjust the GCR based on the current ratio of sleep-to-ideal misses. To illustrate the effectiveness of this scheme, we first demonstrate how the turn-off interval fluctuates within an application. We then compare the adaptive GCR update scheme with a scheme in which the turn-off interval is determined statically. For this study, we examine *gcc* since it has relatively high ideal miss rates in both I- and D-caches. The performance factor is again set at ½.

Figures 6a and 6b show the ideal and sleep miss rates and the turn-off interval, respectively, of *gcc* for both I- and D-caches plotted over execution time. Figure 6a demonstrates how severely the miss rate can fluctuate during program execution. However, by adaptively updating the GCR, we are capable of dynamically maintaining a balanced sleep-to-ideal miss ratio at the PF fraction and maximize the power-performance trade-off throughout the lifetime of the program. The balance in the sleep-to-ideal miss ratio is easier to maintain in the I-cache than in the D-cache. The reason being, as can be seen in Figure 6a, the ideal miss rate of the D-cache changes much more rapidly over time. Hence, it is more difficult to adjust the GCR to balance the sleep-to-ideal miss ratio and to maintain that balance. One final observation from Figure 6b is that the GCR tends to be relatively large for the D-cache, which is due to the smaller ideal cache miss rate comparing to the I-cache ideal miss rate.

Using a fixed GCR scheme may be either too conservative or too aggressive within as well as across applications, as indicated in Figure 1. To obtain optimal results using a fixed approach, the GCR must be chosen on a case-by-case basis and separate GCRs for I- and D-caches must be set within a single application. For example, we studied *gcc* and *go* using a fixed GCR range from 1 to 256. To match the performance of the adaptive scheme, we found the fixed GCRs of 12 and 24 for the I-cache and 24 and 48 for the D-cache for *gcc* and *go* respectively.

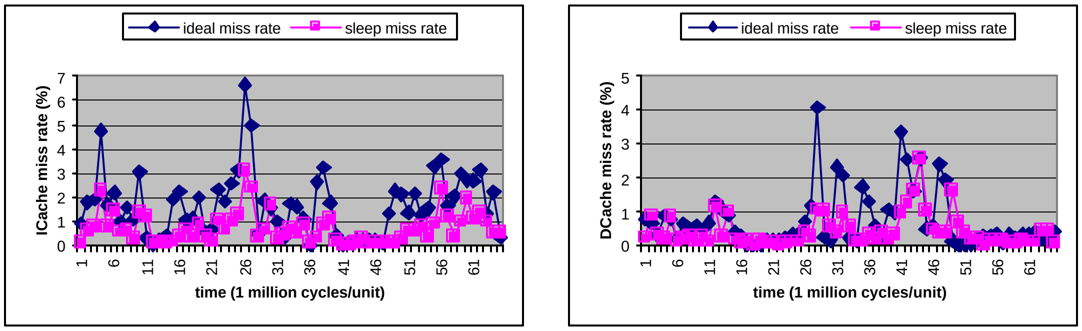

Figure 6a. AMC I- and D-cache miss rate over time

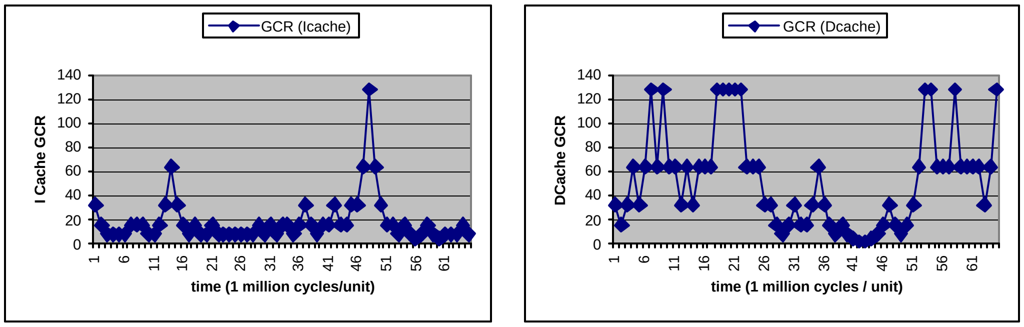

Figure 6b. AMC I- and D-cache GCR over time

5.2.5 Sensitivity of AMC to sense interval, performance factor, cache structure and LIC update interval

The AMC sense interval determines how often the GCR should be updated to account for changing program behavior. In our studies we use a fixed value of 1 million cycles for both I- and D-caches. Although a time-varying value capable of detecting distinct execution phases is ideal, our studies do not show a significant difference when the sense interval is varied from 250000 to 4000000 cycles. The variation in performance is less than 1% and the variation in power saving is less than 2% for I- and D-caches with *compress* being the exception. In *compress*, the performance variation is 1.3% (IPC from 1.49 to 1.51) and the variation in power savings in the D-cache is 9% (from 48.1% to 39.0%).

The performance factor determines the trade-off between performance degradation and static power savings by controlling the sleep-to-ideal ideal miss ratio. A smaller performance factor implies that we are more sensitive to an increase in sleep misses. In the extreme, by setting the performance factor to zero, we effectively turn AMC off. Table 6 shows results for each benchmark in which we varied the performance factor from 1/8 to 1. From these results, it can be seen that the PF provides tight control over IPC except in *li*, *m88ksim*, *twolf* and *vpr*. In these benchmarks, the sleep-to-ideal miss ratio is large enough that the GCR saturates in most cases causing the PF shows no significant effect on IPC.

Table 6. Simulation results with different performance factors (PFs)

Benchmarks	IPC				I-cache turn-off ratio			D-cache turn-off ratio		
	Ideal	PF=1/8	PF=1/4	PF=1	PF=1/8	PF=1/4	PF=1	PF=1/8	PF=1/4	PF=1
Compress	1.56	1.53	1.53	1.49	97.0%	97.0%	97.0%	35.0%	36.3%	46.8%
Gcc	1.84	1.83	1.81	1.74	45.2%	52.9%	66.0%	36.5%	44.3%	60.3%
Go	1.64	1.63	1.62	1.58	15.8%	31.3%	73.4%	40.1%	42.2%	53.3%
Jpeg	1.97	1.96	1.95	1.94	83.8%	83.8%	83.9%	48.0%	51.9%	62.3%
Li	2.19	2.18	2.18	2.18	82.9%	82.9%	83.0%	36.4%	36.4%	36.4%
M88ksim	1.74	1.74	1.74	1.74	78.1%	78.3%	78.7%	63.7%	63.8%	79.7%
Perl	1.91	1.91	1.91	1.91	70.2%	70.3%	70.3%	65.8%	66.1%	66.7%
Vortex	2.35	2.32	2.30	2.21	40.5%	43.6%	57.6%	45.7%	55.6%	76.8%
Twolf	1.23	1.23	1.23	1.23	75.7%	75.8%	76.0%	48.9%	48.9%	49.1%
Vpr	1.60	1.59	1.59	1.56	85.7%	85.7%	85.8%	18.3%	20.1%	31.0%
Average	1.748	1.738	1.733	1.707	67.5%	70.2%	77.2%	43.8%	46.8%	56.2%

To study the effects of cache size on AMC performance, we started off by keeping the same 64KB, 4-way D-cache as in Section 5.2.2 but varying the I-cache size from 16 to 64KB, maintaining the same associativity as in Section 5.2.1. The results show mean turn-off ratios of 49.5%, 64.0%, 74.2% for the 16, 32, and 64KB I-caches, respectively. The mean turn-off ratio for the D-cache showed a change of less than 1% of that given in Table 3. The degradation in IPC for these three cases is 3.0%, 2.3%, and 1.4% for the 16, 32, and 64K I-caches, respectively, while the change in ideal IPC from 32 to 16KB is 4.7% and 1.6% from 64 down to 32KB. Thus, larger caches seem to provide more opportunity for static power savings at lower performance costs.

To examine the effects of associativity, we studied 4-way, 2-way, and directed-mapped I-caches of the same size. Turn-off ratios of directed-mapped caches show increases of up to 30% in *vortex* and 6% in the average as compared to 4-way I-caches. The direct-mapped caches, however, results in a decrease in ideal IPC of up to 2.2% over 4-way caches. Compared to the performance degradation resulting from sleep misses (less than 2%), changes in cache associativity have more of an effect on IPC. In summary, less associativity generates greater static power savings but at the cost of increased performance degradation due to elevated ideal miss rates.

Finally, we looked at the effects of the LIC update interval on AMC performance. As discussed in Section 2.2, finer interval granularity generates more opportunity for cache lines to be put into sleep mode. However, coarser interval granularity results in smaller area costs and energy consumption of the LICs. Simulations showed variations of 0.5% and 2.0% in IPC and turn-off ratio, respectively, as the LIC granularity varies from 256 to 4096 cycles, and, as previously mentioned, an 2048 cycle interval provides a good power-performance trade-off in this range.

5.3. Energy Analysis

In this section, we evaluate the power efficiency of AMC caches using *Energy*Delay* (EDP) [21]. In our studies there are two such products of interest. One is the overall chip-level EDP and the other is the EDP of the cache(s). We examine the effect of AMC cache design on both EDPs in this section. As part of our overall energy analysis of AMC, we performed a detailed analysis of the 0.35μm Compaq 21264 I-cache, which is a 64kB, 2-way set associative cache with 64-byte lines [20].

5.3.1. Cache-Level EDP

At the cache level, we use the average access time of the cache as the delay term and the energy term is calculated as:

$$E_{AMC} = E_{Conv} - E_{\mathrm{Re}duction} \quad \textbf{(1)}$$

where E_{AMC} and E_{Conv} are the energy consumptions of the AMC and conventional caches, respectively, and $E_{Reduction}$ is the energy saved by the AMC cache over a conventional cache. It is calculated as:

$$E_{\mathrm{Re}duction} = E_{leakage\_da\_savings} - E_{LIC} - E_{L2\_cache} \quad \textbf{(2)}$$

$$E_{leakage\_da\_savings} = E_{leakage\_da\_conv} * P_{turned\_off} \quad \textbf{(3)}$$

In Equations 2 and 3, $E_{leakage\_da\_savings}$ is the leakage energy saved in the AMC data store over the conventional data store, E_{LIC} is the total energy dissipation of the LICs and MCL, E_{L2_cache} is the additional L2 dynamic energy consumed due to sleep misses, $E_{leakage\_da\_conv}$ is the data store leakage energy of the conventional cache, and P_{turned_off} is the average AMC turn-off ratio. Note in Equation 3 that we ignore the leakage energy consumed during sleep-mode. This is acceptable since the sleep-mode leakage is on the order of 1/1000 of the active-mode leakage [8].

In Equations 2 and 3, P_{turned_off} is the sole factor determined by our architectural simulations. All remaining terms are implementation dependent. In order to ascertain the energy savings potential of AMC, we first analyze the cost of the LIC, the GCR update hardware and the increased L2 accesses due to sleep misses. We then calculate $E_{leakage\_da\_savings}$ and E_{Conv} to derive the cache-level EDP.

a. Energy Cost Analysis of AMC Hardware and L2 Accesses

The energy of the LICs is determined by the size of the LICs. If we use seven bits to implement each LIC, then, from the 21264 process data, each LIC will consume 15.4fJ of leakage energy. Due to the length of the LIC update interval and the relatively small size of the LICs, the dynamic energy of the LICs is negligible. Similarly, due to the length of the sense interval and the diminutive size of the GCR update logic in comparison to the total cache size, the energy of the GCR and its update logic is also negligible. Thus, we have a total energy contribution of 15.8pJ from the AMC hardware.

The cost of additional L2 accesses due to sleep misses can be calculated as:

$$E_{L2\_cache} = E_{L2\_per\_access} * R_{sleep\_miss} \quad \textbf{(4)}$$

where $E_{L2\_per\_access}$ is the dynamic energy consumed per L2 access and R_{sleep_miss} is the sleep miss rate. It should be noted that the added energy cost of sleep misses can be discarded if we use a sleep-mode refresh mechanism in the data array as described in Section 3. Using the same technology parameters, $E_{L2\_per\_access}$ is calculated as 0.16nJ. Since the mean sleep miss rate is less than 1% for both I- and D-caches using the adaptive GCR update algorithm, as shown in Section 5.2, the added cost of the additional L2 accesses is almost negligible.

b. Static Energy Savings from Data Array and Conventional Cache Energy

Using Equation 3, the static energy savings is a fraction of the static energy consumed by the conventional data array, which is calculated to be 0.17nJ according to the 0.35μm process data and the physical structure of the 21264 I-cache.

To estimate E_{Conv}, we utilize the following relationship from our analysis of the 21264 I-cache: the static energy of data array is one fourth of the worst-case total data array energy. Recall that this analysis is based on 0.35μm technology and the ratio of static-to-dynamic energy will increase dramatically for smaller technologies. Based on the cache energy analysis in [19], the energy consumed in the data array of a conventional 64kB, 2-way set associative cache accounts for 40% of the total cache energy. From this we can estimate the ratio of E_{AMC} to E_{Conv} as:

$$\frac{E_{AMC}}{E_{Conv}} = 1 - \frac{E_{\mathrm{Re}duction}}{E_{Conv}} = 1 - \frac{E_{\mathrm{Re}duction}}{E_{leakage\_da\_conv}} \times \frac{E_{leakage\_da\_conv}}{E_{da}} \times \frac{E_{da}}{E_{Conv}} \le 1 - \left(P_{turn\_off} - 10\%\right) \times 10\% \qquad \textbf{(5)}$$

where E_{da} is the energy consumption of the data array, and the inequality is based on the fact that the sleep miss rate is less than 1%. Now we can define the normalized cache-level EDP, assuming the L1 miss penalty is six times the access time, as:

$$EDP_{cache} = \frac{E_{AMC}}{E_{Conv}} \times \frac{1 + R_{ideal\_miss} \times 6 + R_{sleep\_miss} \times 6}{1 + R_{ideal\_miss} \times 6} \qquad \textbf{(6)}$$

5.3.2 Chip-Level EDP

The delay term at chip level can be defined as CPI, and similar to the cache-level, the chip-level energy term can be computed as:

$$E_{up\_AMC} = E_{up\_conv} - E_{\mathrm{Re}duction} \qquad \textbf{(7)}$$

where E_{up_AMC} and E_{up_conv} are the energy terms for processors using AMC and conventional caches, respectively. Based on the analysis in [17], which shows that on-chip caches account for approximately 20-40% of total chip power, we assume our caches account for 30% of overall processor's total power. Then, similar to Equation 5, the ratio of E_{up_AMC} to E_{up_conv} is defined by:

$$\frac{E_{up\_AMC}}{E_{up\_conv}} \leq 1 - (P_{turn\_off} - 10\%) \times 10\% \times 30\% \quad \textbf{(8)}$$

And we can define the normalized EDP at the chip level as:

$$EDP_{up} = \frac{E_{up\_AMC}}{E_{up\_Conv}} \times \frac{IPC_{ideal}}{IPC_{AMC}} \quad \textbf{(9)}$$

where IPC_{ideal} and IPC_{AMC} are the IPCs using conventional and AMC caches, respectively.

5.3.3 Results

Using the simulation results in Section 5.2, we calculated the EDPs for those cases when AMC caches are used for the I-cache only, the D-cache only, and both I- and D-cache. The results are shown in Figures 7-9, respectively. Note that in our calculations, there are two important ratios affecting the EDP results. The first is the static-to-dynamic energy ratio in the data array, which we calculated as 1/3 according to the 0.35µm 21264 I-cache. This ratio will increase dramatically when technology scales down into deep submicron ranges and it will produce better EDP results at both cache and chip level. The other is the cache-to-chip energy ratio. The larger this ratio, the better the EDP results at chip level. This shows that AMC is more appropriate for those memory intensive embedded systems and we expect that as technology scales, AMC will become more applicable to general-purpose systems.

Figure 7 shows the EDP results for an AMC I-cache and a conventional D-cache. At the chip level, *gcc*, *go*, and *vortex* all have EDPs slightly greater than 1. However, only *go* has an EDP greater than 1 at the cache level. A detailed look at the *go* results will give us some insight into the reasons behind this. Although the I-cache turn-off ratio for *go* is 58.9%, we achieve an energy reduction of less than 4% over the conventional cache according to Equation 5. This indicates that our estimation of the ratio of static-to-dynamic power of 1/3 in the data array is extremely conservative. Thus, we expect significantly increased energy reductions in process technologies in the 0.18µm and lower range. Regardless, the EDPs shown in Figure 7 show some gains even with our conservative assumptions.

Figs. 7 and 8 show EDP results for the AMC D-cache with conventional I-cache and the joint use of AMC I- and D-caches, respectively. Comparing the results of Figure 8 to those in Figure 7, we see an increase in the mean cache EDP and a smaller increase in the mean chip-level EDP. This confirms our prior analysis of Tables 3 and 4 that there are more opportunities for energy savings in the I-cache than in the D-cache. Figure 9 shows the net effect of implementing both caches using AMC. In this case, there is a 5% gain at the I-cache level EDP and a 2.5% gain at the D-cache level EDP.

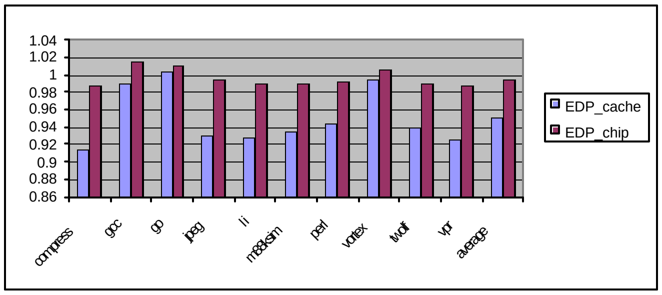

Figure 7. EDP –AMC I-cache only

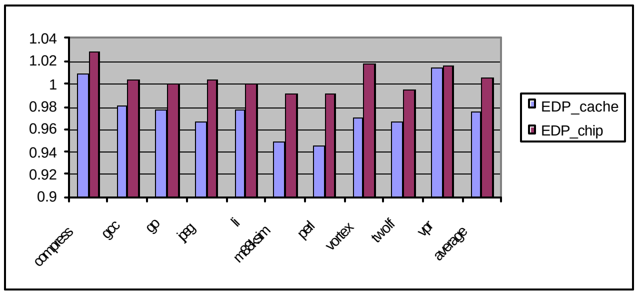

Figure 8. EDP – AMC D-cache only

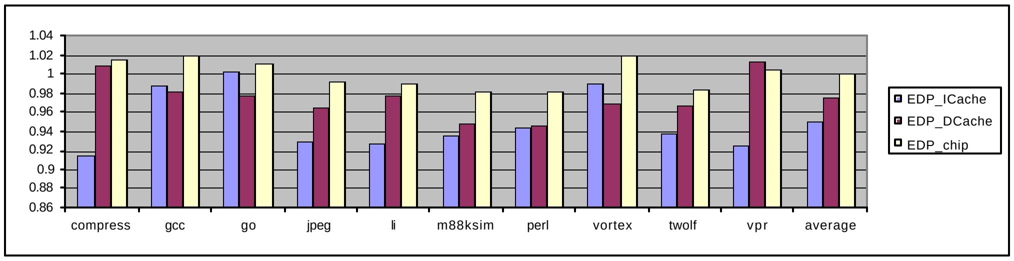

Figure 9. EDP – AMC I- and D-caches

6. Related work

Recently, as power has become a first-order design constraint, there has been a deluge of research in architectural power modeling and optimization of on-chip caches. Several techniques have been proposed to reduce the switching power of on-chip caches. With support from the compiler, selective cache ways [11] enables an appropriate number of ways based on the cache requirements of the current application. The unused ways are disabled by the cache controller through the Cache Way Select Register (CWSR). The L-Cache [12] and Filter cache [13] attempt to reduce L1 cache activity by placing a small L0 cache between the L1 and the processor. With the compiler taking the responsibility of code modification and allocation of instructions into the L-Cache, much smaller performance degradations result as compared to the Filter cache. Block buffering [14] is similar in concept, but, instead of an additional cache level, it places recently requested words into a block buffer inside the cache. With the use of two-phase clocking, the additional access latency can be minimized. Sub-banking in the data array [14] and multiple-divided modules (MDM) [15] also reduce the power consumption by accessing only part of the cache line. In addition to these techniques, several analytical energy models [14, 18, 19] have been proposed to estimate and evaluate cache power and power saving techniques.

The primary goal of the approaches discussed previously is to reduce dynamic power dissipation. The DRI I-cache [5], as mentioned in Section 1, is a mechanism for reducing static power consumption by dynamically resizing and turning-off unused sections by way of the Gated-Vdd technique [4]. As the I-cache size changes over time, an index re-mapping mechanism is necessary which incurs a resizing penalty. In order to obtain optimal power-performance trade-off results, the control parameters, such as miss bound and size bound, must be pre-tuned for different applications. Cache Line Decay [6] targets static power reductions through the use of the Gated-Vdd technique by turning off individual cache lines that have not been accessed for some predefined interval – the decay interval. Since the decay interval is statically fixed, it cannot be updated dynamically to accommodate changes in cache requirements within and across applications.

7. Conclusions

In this paper, we propose an architecture technique that dynamically adapts to the evolving needs of on-chip caches in order to conserve static power while maintaining performance. The main contributions of this study include: 1) Maintaining the tag store information enables the accurate monitor of the cache performance. 2) Proposing an adaptive turn-off interval update scheme to effectively balance the trade-off between power and performance. 3) The power/performance study over I-cache and D-cache demonstrating that AMC is applicable to both L1 caches. Our simulation results show that an average 74% of the I-cache lines and 50% of the D-cache lines can be turned off across all benchmarks at an IPC impact of less than 1.6%.

Our continuing investigation of AMC caches has two primary directions. The first involves the static analysis of cache requirements during compilation and using it to develop a compiler-directed adaptive update of the turn-off interval. The second direction is to utilize dynamic optimization techniques to obtain run-time cache usage profiles to dynamically morph the application to obtain better balance between power and performance. This is particularly useful for applications whose cache usage is hard to discern statically and for D-caches whose accesses depend on run-time load and store addresses.

8. References

[1] J. Montanaro, *et al*, "A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor." *Digital Technical Journal*, vol. 9, Digital Equipment Corporation, 1997.

[2] S. Borkar, "Design Challenges of Technology Scaling." *IEEE Micro*, 19(4): 23-29, July 1999.

[3] Martin Margala, "Low Power SRAM Circuit Design", *IEEE Int. Workshop on Memory Technology, Design, and Testing*, 115-122, 1999

[4] M. Powell, S-H. Yang, B. Falsafi, K. Roy and T. Vijaykumar, "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories." *Proc. of the International Symposium on Low Power Electronics and Design (ISLPED)*, 2000.

[5] S-H. Yang, M. Powell, B. Falsafi, K. Roy and T. Vijaykumar, "An Integrated Circuit/Architecture Approach to Reducing Leakage in Deep-Submicron High-Performance I-caches", *Proc. of the International Symposium on High Performance Computer Architecture (HPCA)*, Jan. 2001.

[6] S. Kaxiras, Z. Hu, G. Narlikar, and R Mclellan, "Cache-line decay: A Mechanism to Reduce Cache Leakage Power", *IEEE workshop on Power Aware Computer Systems*, 2000.

[7] Y.Ye, S. Borkar and V. De, "A New Technique For Standby Leakage Reduction in High Performance Circuits." *IEEE Symposium on VLSI Circuits*, pp. 40-41, 1998.

[8] K. Noii, et.al., "A Low Power SRAM using Auto-Backgate-Controlled MT—CMOS", *ISLPED*, 293-298, 1998

[9] S. Shigematsu, et. al., "A 1-v High Speed MTCMOS Circuit Scheme for Power-Down Application Circuits", *IEEE Journal of Solid-Srate Circuits*, vol. 32, 861-869, 1997

[10] T. Kuroda, et. al., "A 0.9v, 150MHz 10mW, 4mm^2, 2-D Discrete Cosine Transform Core Processor with Variable Threshold-Voltage Scheme", *IEEE Journal of Solid-Srate Circuits*, vol. 31, 1770-1779, 1996

[11] D. Albonesi, "Selective Cache Ways: On-Demand Cache Resource Allocation", *Proceedings of the 32nd Annual IEEE/ACM International Symposium on Microachitecture (MICRO 32)*, 248-259, Nov. 1999

[12] N. Bellas, I. N. Hajj, C. D. Plychronopoulos, and G. Stamoulis, "Architectural and Compiler Techniques for Energy Reduction in High-Performance Microprocessors", *IEEE Trans. On VLSI Systems*, 8(3): 317-326, 2000

[13] J. Kin, M. Gupta, and W. Mangione-Smith, "The filter cache: An energy efficient memory structure", *MICRO 30*, 184-193, 1997

[14] M. B. Kamble and K. Ghose, "Analytical Energy Dissipation Models for Low Power Caches", *ISLPED*, 143-148, Aug. 1997.

[15] U. Ko and P.T. Balsara, "Characterization and Design of A Low-Power, High-Performance Cache Architecture", *International Symposium on VLSI Technology, Systems, and Applications*, 235-238, 1995

[16] D. Burger and T. M. Austin, "The Simplescalar Tool Set Version 2.0", Technical Report, Computer Science Department, University of Wisconsin-Madison, 1997

[17] N. Bellas, I. Hajj, and C. Polychropoulos, "A detailed, transistor-level energy model for SRAM-based caches", *Proc. Int. Symp. Circuits and Sytems*, 1999

[18] N. Vijaykrishnan, M. Kandemir, M.J., Irwin, H.S. Kim and W. Ye, "Energy-Driven Integrated Hardware-Software Optimizations Using SimplePower", *Proceedings of 27th Int. Symp. On Computer Architecture (ISCA)*, 95-106, 2000

[19] G. Reinman and N. Jouppi, "An Integrated Cache Timing and Power Model", *CACTI 2.0 Technical Report, COMPAQ Western Research Lab*, 1999

[20] L. Gwennap, "Digital 21264 Sets New Standard", *Microprocessor Report*, vol. 10, no. 14, Oct. 1996.

[21] R. Gonzalez and M. Horowitz, "Energy Dissipation in General Purpose Microprocessors", *IEEE Journal of Solid-State Circuits*, 31(9): 1277-1284, 1996