Cache-line Decay: A Mechanism to Reduce Cache Leakage Power

Stefanos Kaxiras,* Zhigang Hu,+ Girija Narlikar,* Rae McLellan*

* Bell Laboratories, Lucent Technologies

{kaxiras, girija, rae}@research.bell-labs.com

+ Princeton University

hzg@ee.princeton.edu

Abstract—*Reducing the supply voltage to reduce dynamic power consumption in CMOS devices, inadvertently will lead to an exponential increase in leakage power dissipation. In this work we explore an architectural idea to reduce leakage power in data caches. Previous work has shown that cache frames are "dead" for a significant fraction of time [14]. We are exploiting this observation to turn off cache lines that are not likely to be accessed anymore. Our method is simple: if a cache-line is not accessed within a fixed interval (called* decay interval*) we turn off its supply voltage using a gated V_{dd} technique introduced previously [1]. We study the effect of cache-line decay on both power consumption and performance. We find that it is possible with cache-line decay to build larger caches that dissipate less leakage power than smaller caches while yielding equal or better performance (fewer misses). In addition, because our method can dynamically trade performance for leakage power it can be adjusted according to the requirements of the application and/or the environment.*

1 Introduction

Striving for low-power, high-performance CMOS devices drives supply voltage (V_{dd}) to ever lower levels [6]. To maintain performance, a reduction in V_{dd} necessitates a reduction in threshold voltage (V_{th}), which in turn increases leakage power dissipation exponentially [4,9,10]. Since chip transistor counts continue to increase, and every transistor that is powered on leaks irrespective of its switching activity, leakage power is expected to become a significant factor in the total power dissipation of a chip [4]. Given the current trends [8,10], the leakage power dissipated by a chip could equal its dynamic power within three processor generations.

Although the leakage power of SRAM cells can be lower than the leakage power of high-speed logic gates [14], on-chip caches can still contribute a significant percentage of a chip's leakage power for two reasons. First, because a large fraction of the transistors are in the cache memory. Second, memory fabric cells are composed of low fan-in gates, namely cross coupled inverters with only a single leaking transistor to a power rail. In contrast, significant parts of the logic circuits typically consist of higher fan-in gates with more transistors connected in series to a power rail (*stacked transistors* [9]). Given that the leakage power dissipation is becoming significant, circuit-level or micro-architectural solutions for on-chip caches are necessary to deal with the whole problem.

One solution for reducing leakage power is to switch off power to unused devices. Powell, Yang, Falsafi, Roy, and Vijaykumar recently proposed a micro-architectural technique called *DRI cache* and a circuit-level technique called *gated V_{dd}* to switch-off V_{dd} (or V_{ss}) to large blocks of the instruction cache [1,2]. Motivated by their approach, we extend it by applying a similar idea to data caches but instead of large portions of the cache, we propose switching off individual cache lines as required.

Our proposed scheme, called *Cache Decay*, consists of invalidating and turning off power to cache lines that have not been accessed for a certain interval, called the *Decay Interval*. When a powered down cache line is accessed, a cache miss is incurred while the line is switched back on and data are fetched from memory. Other cache-line aging techniques have been used in other contexts, for example for Dynamic Self Invalidation [11,13], and for managing group associative caches [12]. In contrast to previous work, we propose very simple, low-overhead implementations since our main goal is to reduce power consumption.

We studied the cache access patterns for a set of SPEC95 benchmarks; they display a high degree of temporal locality (see Section 2), and indicate that turning off cache blocks that remain inaccessible for an appropriate period of time will not significantly increase miss rates. We study the effect of varying the decay interval on a variety of benchmarks. For very small decay intervals (thousands of cycles), the application can suffer a larger number of cache misses; for very large intervals (hundreds of thousands of cycles), very few cache blocks may decay in time to be powered down. However, we find that for a wide range of decay intervals, the cache decay technique is successful in switching off large portions of the on-chip data cache without significantly affecting miss rates.

Contributions of this paper are as follows:

- We propose cache decay as a mechanism to turn off unused lines in the cache.
- We describe in detail a digital implementation of the cache decay mechanism and discuss an analog implementation.
- We study the effects cache decay on power and performance using execution-driven simulation and SPEC95 programs. Our results demonstrate the effectiveness of the cache decay scheme. In particular, we show that an L1 data cache without cache decay can be replaced by an L1 cache of twice the size with the same performance but up to 56% less leakage power.

***Organization of this paper*—**In Section 2 we discuss cache decay and its implementations. Section 3 presents details of out experimental methodology and Section 4 results of our experiments. We conclude in Section 5.

2 Cache Decay

Recent work by Powell et al. [1] showed that powering down sections of the instruction cache and resizing it results in significant leakage-power savings. Motivated by this approach we examined switching-off parts of the data cache but at a much finer granularity (cache-line granularity) and without resizing. We rely on the fact that many cache frames are underutilized and therefore can be turned off without impact on performance. Evidence of this comes from two papers:

- Wood, Hill, and Kessler showed that the miss rate of unknown references (cold misses) in a trace-driven simulation with unknown initial conditions is much higher than the steady-state miss rate (e.g., 0.40 vs. 0.02) [14]. The high cold-miss rate is simply the ratio of time a cache frame is dead (i.e., the time between last hit and replacement).
- In a paper examining cache efficiency, Burger, Goodman, and Kagi showed that most of the data in a cache will not be used in the future (either will be overwritten or will not be accessed at all) [3].

Although discovering dead data in a cache is not a trivial matter, we hypothesized that a simple technique could actually capture some of the benefit. Our approach attempts to switch off *least recently used* cache lines assuming that these will be unlikely to be accessed in the future. To substantiate our hypothesis we profiled the execution of SPEC95 programs. Figure 1 shows the distributions of access intervals—intervals between consecutive accesses to the same cache line—for three programs.[1] The horizontal axis of the graphs is the access interval (in hundreds of cycles) and the vertical axis is the percentage of the accesses corresponding to an access interval (i.e., distance from the previous access). The last point in the horizontal axis represents the tail of the distribution which is quite small in gcc and vortex but sizable in compress.

Since most consecutive accesses to the same cache-line are spaced closely in time—temporal locality—a cache line that has not been accessed for some time either will not be accessed again or it is one of the few cache lines that will be accessed very far into the future. Therefore, we propose to maintain power to cache lines as long as they are accessed within some predefined time interval (*decay interval*). We have identified digital and analog implementations to detect the passage of a decay interval from the last access to each cache line. We present these implementations in sections 2.1 and 2.2.

Regardless of the implementation, cache-line decay will increase the miss rate of the cache: a few lines will be powered-off before they are accessed. However, as we will show in Section 4 the miss rate of a decay cache is still less than a smaller cache whose size matches the average powered size of the decay cache. Another way to view the decay cache is from a leakage power efficiency perspective: the average powered size of a decay cache is smaller than a cache of equal miss rate.

2.1 Digital implementation

One way to represent recency of a cache line's access is via a digital counter. This counter is cleared on each access to the cache line and incremented periodically at fixed time intervals. Once the counter reaches its maximum count it saturates and switches off power (or ground) to the corresponding cache line.

A casual interpretation of the graphs in Figure 1 suggests decay intervals of tens or hundreds of thousands of cycles. Because the number of cycles needed for a reasonable decay interval makes it impractical for the counters to count cycles (too many bits would be required) it is necessary for the counters to "tick" at a

[1] Other SPEC95 programs produce similar results.

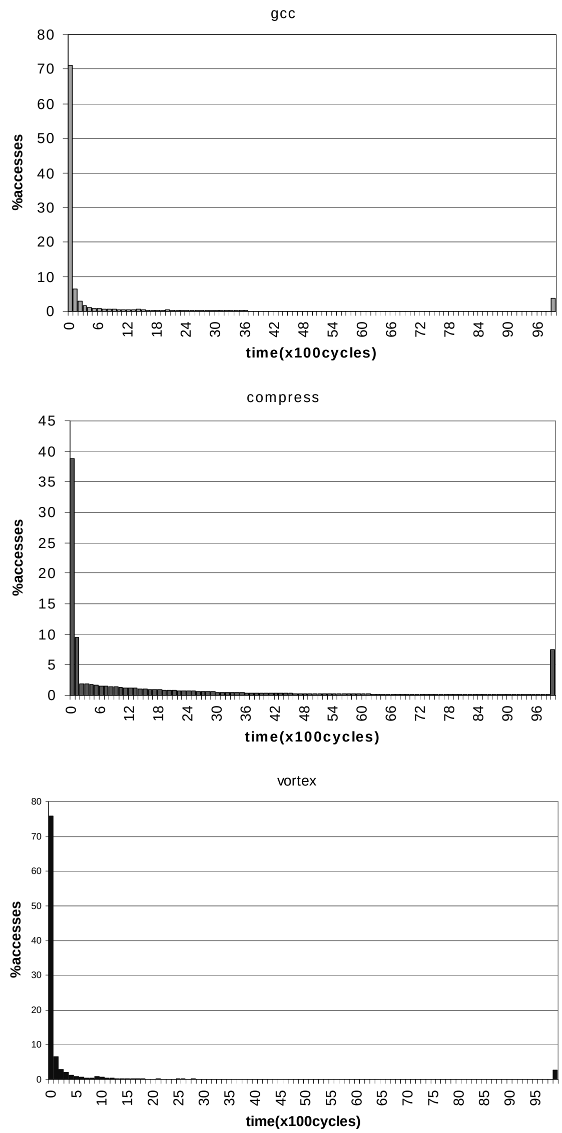

FIGURE 1. Access intervals for gcc, compress and vortex

much coarser level, for example every few thousand cycles. A *global cycle counter* can be set up to provide the ticks for smaller *cache-line counters* (as shown in Figure 2). Simulations show that a two-bit counter per cache line provides sufficient resolution.

Global counter—To save power, the global counter can be implemented as a binary ripple counter. An additional latch holds a maximum count value which is compared to the counter. When the counter reaches the maximum value, it is reset and a 1-clock-cycle *T* signal is generated. This circuit can be implemented with 40*N* + 20 transistors, where *N* is the number of bits required. The maximum-count latch is non-switching and doesn't contribute to dynamic power dissipation. On average, only two bits of the counter and comparator—less than 80 transistors—will change state and dissipate dynamic power each clock cycle.

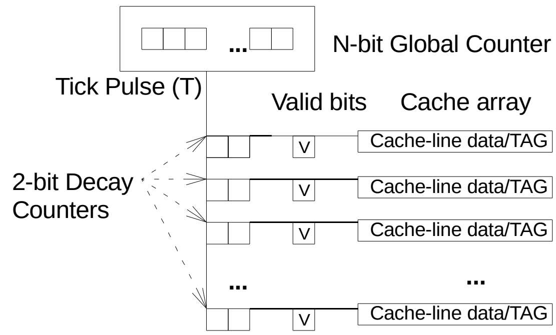

FIGURE 2. High-level view of the digital implementation with 2-bit, Gray-code, saturating counters

Cache-line counters—To minimize state transitions in these counters—and thus minimize dynamic power consumption—we use Gray coding so only one bit changes state at any time. Furthermore, to simplify the counters and minimize transistor count we chose to implement them asynchronously. Each cache line contains circuitry to implement the state machine depicted in Figure 3.

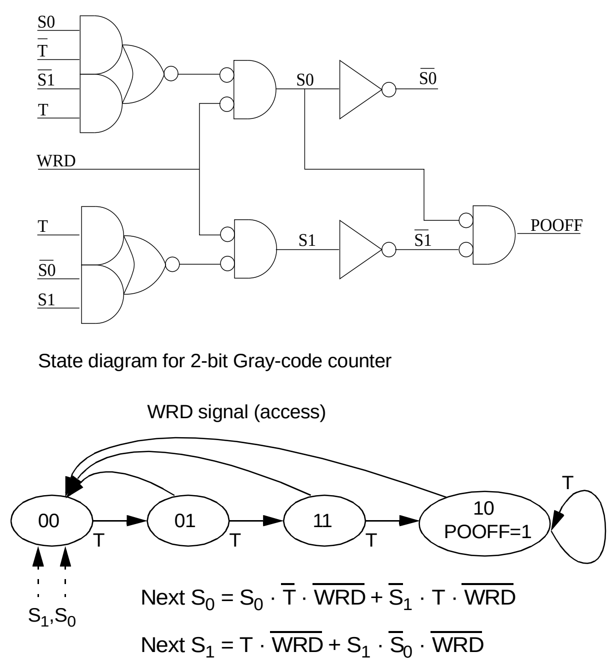

FIGURE 3. Two-bit (*S0,S1*), saturating, Gray-code counter with two inputs (*WRD* and *T*)

There are two inputs to the counter circuit:

1. A global time signal, *T*, is a periodic pulse to indicate the passage of time and it is supplied by the global cycle counter. *T* is a *well behaved* digital signal whose period may be adjusted externally to provide different decay intervals appropriate for different programs.
2. The second state machine input is the cache line access signal, *WRD*, which is decoded from the address and is the same signal used to select a particular row within the cache memory (e.g., the *WORD-LINE* signal).

State transitions occur asynchronously on changes of the two input signals, *T* and *WRD*. But since *T* and *WRD* are well behaved signals, there are no metastability problems. The only output is the cache-line switch state, *POOFF (POwer OFF)*.

Implementation details—Switching-off power to a cache line has important implications for the rest of the cache circuitry. In particular, the first access to a powered-off cache line should:

1. result in a miss (since data and tag might be corrupted without power)
2. reset the counter and restore power to the cache line (i.e., restart the decay mechanism as per Figure 3)
3. be delayed an appropriate amount of time until the cache-line circuits stabilize after power is restored.

To satisfy these requirements we use the *Valid* bit of the cache line as part of the decay mechanism (Figure 4). First, the valid bit is always powered. Second, we add a reset capability to the valid bit so it can be reset to 0 (invalid) by the decay mechanism. The *POOFF* signal clears the valid bit. Thus the first access to a power-off cache line always results in a miss *regardless of the contents of the tag*. Since satisfying this miss from the lower memory hierarchy is the only way to restore the valid bit, a newly powered cache line will have enough time to stabilize. In addition, no other access (to this cache line) can read the possibly corrupted data in the interim.

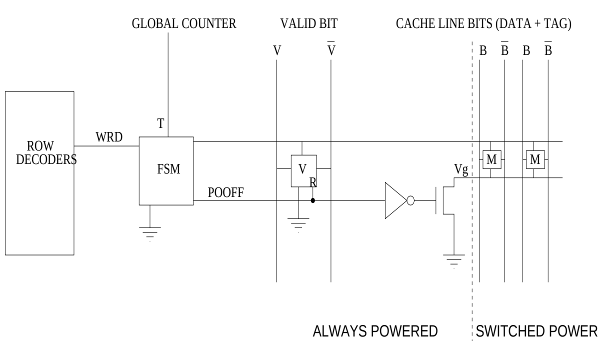

FIGURE 4. Cache-line power control

2.2 Analog implementation

An alternative way to represent the recency of a cache line's access is via charge stored on a capacitor (Figure 5). Each time the cache line is accessed, the capacitor is grounded. In the common case of a frequently accessed cache-line the capacitor will be discharged. Over time, the capacitor is charged through a resistor connected to V_{dd}. Once the charge reaches a sufficiently high level, a voltage comparator detects it, asserts the *POOFF* signal and switches off power (or ground) to the corresponding cache line (data bits and tag bits).

This method suffers from two problems. First, the *RC* time constant cannot be changed. It is determined by the fabricated size of the capacitor and resistor and cannot be adapted to different program's temporal access patterns. Second, it is inherently a noise sensitive analog circuit and can change state asynchronously with the remainder of the digital circuitry. Some method of synchronously sampling the voltage comparator must be employed to avoid metastability. As in the digital implementation the valid bit is set to 0 (invalid) when a cache line is powered-down.

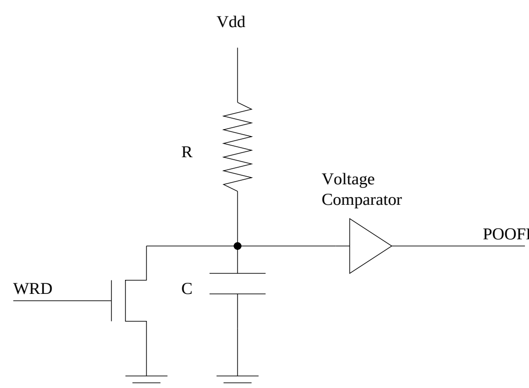

FIGURE 5. Analog implementation. Switch-off cache line on capacitor charge.

3 Methodology

In this work we use execution-driven simulation to study the run-time behavior of decay caches (e.g., miss rate and ratio of powered-off cache-lines). We then use the simulation results to model leakage power consumption. The results of the power models allow us to make comparisons among cache configurations with and without decay mechanisms. In Section 3.1 we describe in more detail our experimental setup and in Section 3.2 we discuss power consumption models.

3.1 Experimental Setup

To evaluate the effectiveness of cache-line decay we use seven SPEC95 benchmarks. We present detailed results for three programs exhibiting medium (gcc), high (compress), and low (vortex) miss rates. We simulated the execution of these benchmarks for 500 million instructions using SPEC95 reference inputs on the SimpleScalar simulator using the SimpleScalar 2.0 tool set [5]. We use the detailed, out-of-order superscalar processor (with non-blocking caches) simulator to run the benchmarks since we must accurately account for *time* differences in cache accesses. Simulator parameters are shown in Table 1. In our studies we concentrate on L1 caches. We chose to examine small cache sizes from 8Kbytes to 32Kbytes because SPEC95 programs do not stress larger caches (we want to be conservative since our methods would work much better in larger, less utilized caches). In small caches virtually all cache lines were accessed during the execution of the programs.

Parameter	Value
Physical Registers	64-INT, 64-FP
Fetch width	4 instructions per cycle
Decode width	4 instructions per cycle
Issue width	4 instructions per cycle
Commit width	4 instructions per cycle
Functional Units	4 IntALU, 1-IntMult/Div, 2 FP ALU, 1-FPMult/Div, 2 MemPorts
Branch Predictor	Combined, Bimodal 4K table, 2-Level 1K table, 10bit history, 4K chooser
BTB	1024-entry, 2-way
Return Address Stack	32-entry
Mispredict penalty	7 cycles
L1 Dcache Size	8K, 16K and 32K, 2-way, 16B blocks
L1 Icache Size	64K, 2-way, 32B blocks
L2 (Omitted in some experiments)	Unified, 256K, 8-way LRU, 32B blocks,12-cycle latency
Memory	100 cycles
TLB Size	128-entry, 30-cycle miss penalty

TABLE 1: Simulator configuration parameters

The simulator was modified to switch off cache lines (both tag and data) according to the cache decay schemes described in Section 2. Every access to the cache block restarts the decay mechanism for that line. We use the simulator to measure various statistics such as the number of cache misses, the fraction of the cache that is powered up, the number of times the decay counters are incremented and change state, etc.

3.2 Power computation

The additional dynamic power dissipated due to the decay circuitry is proportional to the product of its load capacitance and the switching activity of its transistors. For the implementation described in Section 2, less than 110 of its transistors switch every cycle, and the entire decay circuitry involves a very small number of transistors: a few hundred for the global counter plus under 30 transistors per cache line. Thus, the dynamic power dissipation of the decay circuitry is negligible compared to the dynamic power dissipated in the remainder of the chip, which presumably contains millions of transistors. We therefore compute in detail only the leakage power of the cache with and without decay.

We assume a fixed threshold voltage in our experiments. The leakage power for the cache is therefore assumed to be proportional to the total number of cache lines that are powered-on in the cache. We compare the leakage power of the original cache with that of the additional decay circuitry by assuming it to be proportional to the total number of transistors in both those subblocks. We compute the total number of transistors in the original cache (we include both data bits and tag bits) and the additional decay circuitry from the transistor counts in Section 2.

4 Results

We start our evaluation with the behavior of the decay scheme using full counters per cache-line (Section 4.1). This is an idealized scheme—clearly impractical—but shows the behavior of various cache configurations with precise control of the decay interval. Subsequently, Section 4.2 presents detailed results for a realistic digital implementation with a global counter and 2-bit counters per cache-line. In the same section we show power and performance benefits of the decay caches over standard caches half their size. Section 4.3 shows an alternative view of cache decay comparing equal size decay and standard caches for seven SPEC95 programs.

4.1 Sensitivity to decay interval, cache size and block size

Results in this section were obtained with precise control over the decay interval: a cache line is switched-off when a specific number of cycles has passed since it was last accessed. We achieve this by simulating full counters (as many bits as needed) per cache line. Varying the decay interval, we measure miss rate and active ratio for a given cache configuration. We define active ratio as the *average* part of the cache that is switched on per cycle during the execution of a program. Figure 6 shows the graphs for three programs using 16K, 16-byte-block caches. An infinite interval ("*inf.*" on the *x* axis) represents the standard cache where nothing is turned off. In this case we have the minimum miss rate and the maximum active ratio. For all three programs the active ratio is 100% meaning that the whole cache is accessed. Decreasing the decay interval to 512K and

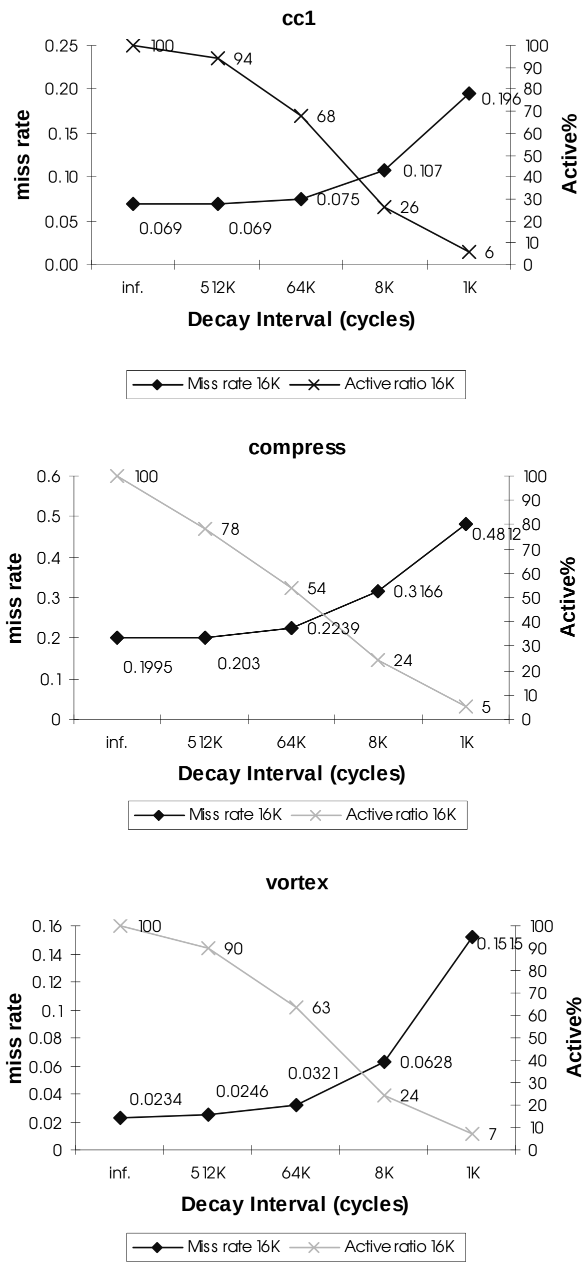

FIGURE 6. Miss-ratio/Active ratio graphs for gcc, compress and vortex varying the decay interval.

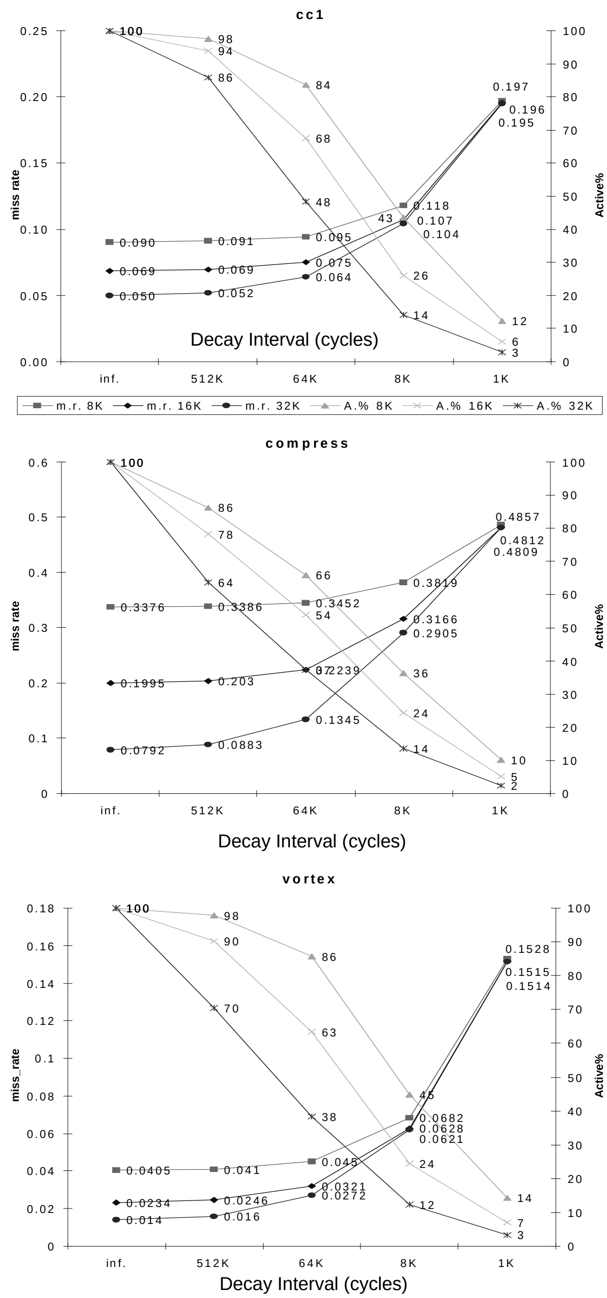

FIGURE 7. Miss rate and Active ratio as a function of decay interval for GCC, Compress and Vortex. 32, 16, and 8 KByte caches, 2-way, 16-byte blocks

64K cycles, increases miss rate slightly but decreases the active ratio significantly. However, further decreasing the decay interval to 8K and 1K cycles starts to show dramatic increases in miss rate. This result also agrees intuitively with data presented in Section 2: as we begin to switch-off valuable cache lines (frequently and heavily accessed), miss rate becomes increasingly worse.

Figure 7 extends the graphs of Figure 6 by adding curves for smaller (8KBytes) and larger (32KBytes) caches. Similar miss rate and active ratio curves resulted for smaller and larger caches albeit shifted with respect to the axes. As expected, smaller caches have higher miss rates. However, the miss rate of all caches converges to the same value as we decrease the decay interval. Smaller caches also have higher active ratios for a given decay interval. Active ratios converge toward 100% as we increase the decay interval. Figure 7 provides two rules for decay intervals:

1. Smaller caches need a smaller decay interval to achieve the same active ratio.
2. Smaller caches need a smaller decay interval to yield the same relative *increase* in miss rate.

We also examined the relation of the decay interval to line size. Figure 8 shows how the active ratio and the miss rate change as a function of line size (16, 32 and 64 Bytes). Whereas miss rate can either increase or decrease depending on the spatial characteristics of the application, active ratio decreases with larger line size. For a given decay interval larger cache lines are less likely to be turned-off.

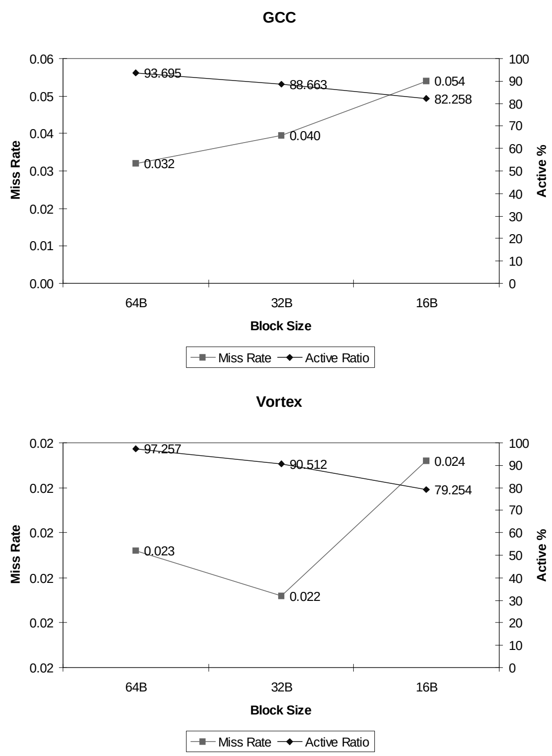

FIGURE 8. Miss rate and active ratio as a function of block size for GCC and Vortex with 16KBytes, 2-way caches.

4.2 Digital cache-line decay in data caches

As we described in Section 2, a realistic digital implementation employs two-bit counters per cache line and a global counter to generate the tick (T) signal at regular intervals. The global counter's period, T_{period}, can be set externally. The decay interval in this case is not exact but rather varies from two to three times T_{period} with an average value of 2.5 T_{period}. Comparing results of the variable decay interval with those presented in the previous section (precise decay interval) we found very little difference. The resolution of the two-bit counter is enough to approximate a precise decay interval equal to 2.5 T_{period}.

Using two-bit counters we compare decay caches to standard caches by: first, keeping the miss rates equal, and second, maintaining the *effective* size of the decay cache equal to the size of the standard cache.

4.2.1 Equal miss rate comparisons

Cache-line decay is a trade-off between dynamic and static power. By switching off cache lines we save leakage power but, on the other hand, we incur more misses which consume power. Dynamic power is also dissipated by the power-management circuits but this is negligible. Quantifying power consumption for a cache miss requires precise knowledge of implementation details: bus power consumption, timing, buffers, etc. We believe that results specific to an implementation cannot be generalized.

We remove power consumption due to cache misses by comparing standard caches to decay caches of double size but of equal miss rate. We control miss rate in the decay caches by choosing an appropriate decay interval per application.

We use curve fitting on the data presented in Figure 7 to approximate miss-rate curves. In this way we estimate a decay interval that will give us a desired miss-rate.[2] Figure 9 shows the results of this approach. The first graph compares a 32KByte decay cache to a 16KByte standard cache and the second graph compares a 16KByte decay cache to a 8KByte standard cache. We list the estimated decay intervals for every case in Table 2. The bar pairs show miss rates for the decay and standard caches, while the solid line shows the effective size of the decay cache (active ratio multiplied by actual size). The effective size of the 32KByte decay cache is 27%, 50%, and 10% *smaller* than the 16KByte standard cache for gcc, vortex, and compress respectively (for the 16KByte decay cache, 26%, 59%, and 5% *smaller* than the 8KByte standard cache).

By keeping the miss rates constant we reduce leakage power but on the other hand we have added additional circuitry that dissipates both dynamic and leakage power. The increase in dynamic power is negligible compared to the dynamic power dissipation of the entire chip; in particular, on average 80 transistors in the additional decay circuitry switch every clock cycle (com-

---

[2] Our methods introduce small errors so the miss rates of the decay caches are not strictly equal or smaller to those of the standard caches for this paper. However, miss-rate differences are so small that for our purposes we consider them insignificant. More conservative decay-interval estimates can reduce miss-rate to the desired value.

	32KByte Decay cache	16KByte Decay cache
gcc	32Kcycles, T_{period} = 14.4K	17Kcycles, T_{period} = 6.8K
compress	24Kcycles, T_{period} = 9.6K	6Kcycles, T_{period} = 2.4K
vortex	100Kcycles, T_{period} = 40K	29Kcycles, T_{period} = 11.6K

TABLE 2: Decay intervals for equal-miss-rate comparisons

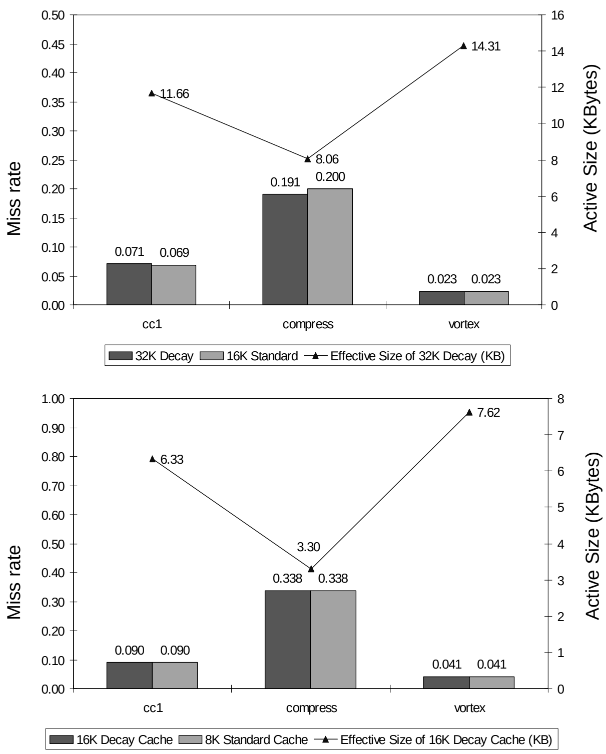

FIGURE 9. Equal miss-rate comparisons: a 32KByte (16KByte) decay cache has smaller *effective* size than a 16KByte (8KByte) standard cache.

pared to the millions switching in the processor core). We therefore focus on computing leakage power. Figure 10 shows the relative change in the leakage power of the cache itself when the cache decay mechanism is used. Although the additional circuitry increases leakage to a small extent, the total leakage power of the cache is reduced significantly because large portions of the data cache get turned off. Since the cache is one of the main contributors to the total leakage power of the chip (Section 1), cache decay results in large savings when leakage power becomes significant.

4.2.2 Equal size comparisons

An alternative way to examine decay caches is to keep their effective size the same as a standard cache half the size, i.e., keep the active ratio less than or equal to 50%. Again we estimate the decay interval that will

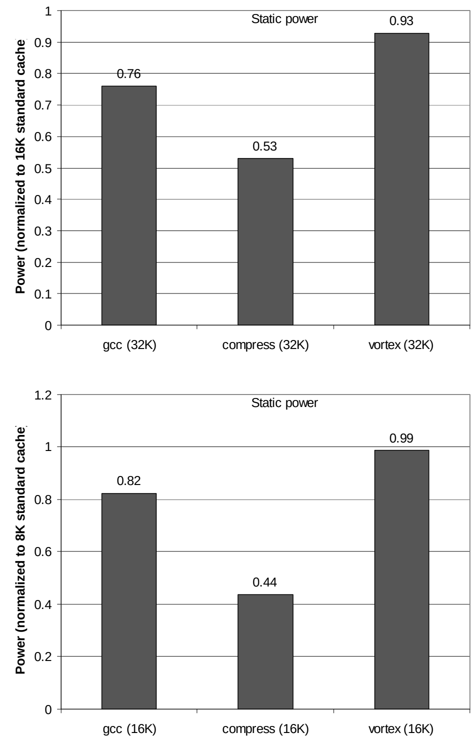

FIGURE 10. Static power dissipation of decay caches (16K and 8K) normalized to standard caches (16K and 8K) for gcc, compress and vortex

give us such an active ratio from the graphs in Figure 7. Figure 11 shows that equal effective size means decreases in miss rate (significant in the case of compress). This in turn translates into increased performance and decreased dynamic power attributable to misses.

4.3 Effects on performance

Besides the above relative comparisons, we show the effects of cache-line decay on miss rate, IPC, and active size by comparing decay caches to standard caches of equal size. Here, we do not use L2 caches—L1 misses are serviced directly from memory—for two reasons: i) we want to make evident the IPC impact of the decay mechanisms (L2 caches tend to reduce it to insignificant levels), and ii) we concentrate on small L1 caches to reflect the size of the benchmarks—cache-line decay would work well in relatively large cache hierarchies.

Figure 12 shows the miss rate, IPC, and effective size (percent of actual size) for 32KByte decay and standard caches. We chose a single decay interval of 128K cycles for all benchmarks (which may not be optimum for all benchmarks). This decay interval is larger than what Figure 7 would suggest for the miss rates and

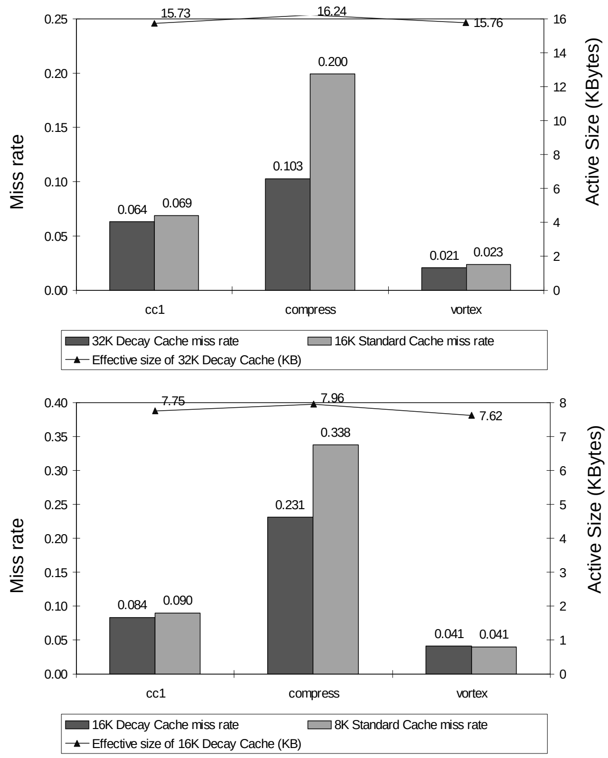

FIGURE 11. Equal area comparisons: a 32KByte (16KByte) decay cache of about 16KByte (8KByte) *effective* size has a smaller miss rate than a 16KByte (8KByte) standard cache.

active ratios of Figure 12. This is because the lack of L2 expands the time between cache accesses. The increased average memory latency necessitates an increase in the decay interval. Increases in miss rate range from 8% (m88Ksim) to 293% (go) with an average of 88%. However, the decrease in IPC is moderate and ranges from to 6% (m88Ksim and li) to 18% (go) with an average of 14%. The superscalar out-of-order core largely hides the increase in average memory latency. Decrease in leakage power for the cache memory array ranges from 57% (gcc) to 75% (go) with an average of 67%.

5 Conclusions

In this paper we propose cache decay, a mechanism to reduce leakage power dissipation in caches. We turn off power to cache lines that have not been accessed within a decay interval. By controlling power at a cache-line granularity we can achieve a significant reduction in leakage power while at the same time preserve much of the performance of the cache. A decay cache can have an effective size much smaller than a cache of equal miss-rate. Alternatively, a decay cache with the effective size of a small cache performs better. In addition, the full performance of the decay cache is available

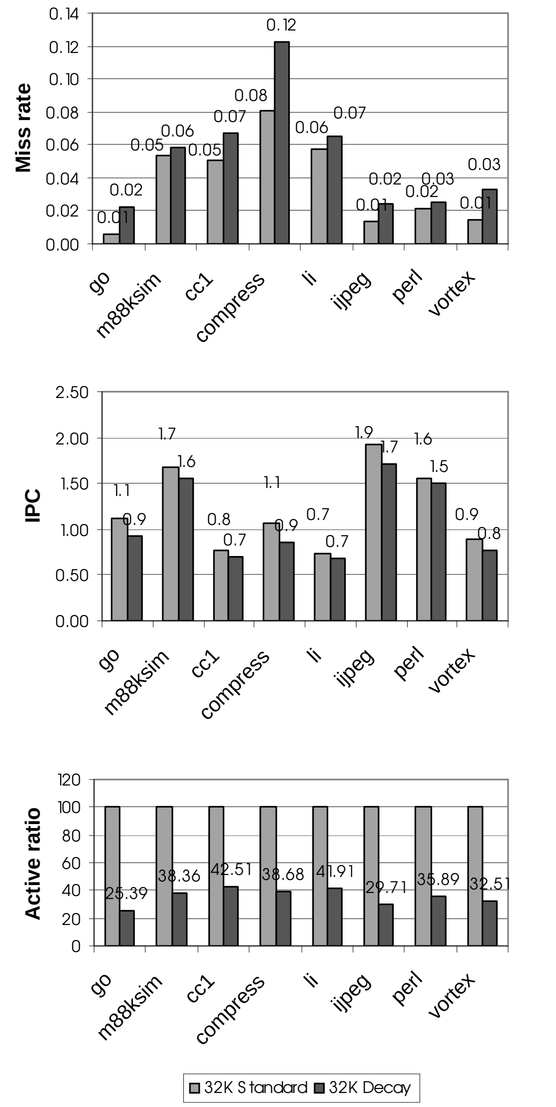

FIGURE 12. Miss rate, IPC and Active ratio for 32K standard and 32K decay cache. There is no L2 cache. 128K cycles decay interval for all programs.

to demanding applications when power consumption in not an issue. This flexibility of the decay cache is particularly useful in battery-powered computers.

Our results show that different applications have different optimal decay intervals for a given miss-rate/active ratio target. Our proposed digital implementation can be controlled at run-time by the operating system via the global cycle counter. The OS can set the period of the global counter (T_{period}) to produce the desired decay interval according to the demands of the executing application and the power-consumption requirements of the system. Profiling and/or run-time monitoring can be used to adjust decay intervals. In contrast, the analog implementation is not as flexible—the decay interval is fixed at design time by the *RC* delay. However, its simplicity may be appealing for selected embedded applications. We are working to refine implementation details of the analog design and quantify its behavior. Cache decay can also be applied to instruction caches. Limited experimentation showed moderate success but further work is needed to understand decay in instruction caches.

6 Acknowledgments

We would like to thank Margaret Martonosi (Princeton U.) for the discussions we had about this work. We would also like to thank Sean Dorward (Bell Labs), Mark Hill, Guri Sohi, and Jim Goodman (U. of. Wisconsin), and the anonymous referees of PACS ‘00 for their comments and suggestions.

7 References

[1] Mike D. Powell, Se-Hyun Yang, Babak Falsafi, Kaushik Roy, T. N. Vijaykumar “Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories,” *ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*.

[2] Se-Hyun Yang, Michael D. Powell, Babak Falsafi, Kaushik Roy, and T. N. Vijaykumar, “An Integrated Circuit/Architecture Approach to Reducing Leakage in Deep-Submicron High-Performance I-Caches,” To appear in *ACM/IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2001.

[3] Douglas C. Burger, James R. Goodman, and Alain Kagi “The Declining Effectiveness of Dynamic Caching for General-Purpose Microprocessors,” *University of Wisconsin-Madison Computer Sciences Dept. Tech. Report 1261*, January, 1995.

[4] Shekhar Borka, "Design Challenges of Technology Scaling", in IEEE Micro, Vol. 19, No. 4, July/August 1999.

[5] Doug Burger and Todd Austin, “The SimpleScalar Tool Set Version 2.0,” *University of Wisconsin-Madison Computer Sciences Dept. Tech. Report 1342*, June, 1997

[6] I.C Kizilyalli et al. “A WSi/WSiN/Poly:Si Gate CMOS Technology for 1.0V DSPs,” In *Proceedings of the First International Symposium on ULSI Process integration, The Electrochemical Society Proceedings Vol. 99-18*. pp 347-352.

[7] Tohru Ishihara and Hiroto Yasuura, “Experimental Analysis of Power Estimation Models of CMOS VLSI Circuits,” *IEICE Trans. Fundamentals, Vol.E80-A No.3, pp.480-486*, March 1997.

[8] Scott Thompson, Paul Packson, Mark Bohr, “MOS Scaling: Transistor Challenges for the 21st Century,” *Intel Technology Journal*, 3rd Quarter, 1998.

[9] Zhanping Chen, Mark Johnson, Liqiong Wei and Kaushik Roy, "Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modeling of Transistor Stacks," *Proceedings of 1998 international symposium on Low power electronics and design*. Pages 239-244.

[10] S. Bobba and I.N. Hajj, “Maximum Leakage Power Estimation for CMOS Circuits,” *Proceedings of the IEEE Alessandro Volta Memorial Workshop on Low-Power Design.*

[11] Alvin R. Lebeck and David A. Wood, “Dynamic Self-Invalidation: Reducing Coherence Overhead in Shared-Memory Multiprocessors,” *International Symposium on Computer Architecture (ISCA)*, June 1995

[12] Jih-Kwon Peir, Yongjoon Lee, Windsor W. Hsu, “Capturing Dynamic Memory Reference Behavior with Adaptive Cache Topology,” *Architectural Support for Programming Languages and Operating Systems (ASPLOS VIII)*, 1998.

[13] An-Chow Lai and Babak Falsafi, “Selective, Accurate, and Timely Self-Invalidation Using Last-Touch Prediction,” *International Symposium on Computer Architecture (ISCA)*, May 2000

[14] David A. Wood, Mark D. Hill, R. E. Kessler, “A Model for Estimating Trace-Sample Miss Ratios,” *ACM SIGMETRICS*, May 1991.

[15] J. Adams Butts and Gurindar Sohi, “A Static Power Model for Architects,” To Appear in *MICRO-33 - 33rd Annual International Symposium on Microarchitecture.*