Exploring the Limits of Leakage Power Reduction in Caches

Yan Meng, Timothy Sherwood and Ryan Kastner
University of California, Santa Barbara

If current technology scaling trends hold, leakage power dissipation will soon become the dominant source of power consumption. Caches, due to the fact that they account for the largest fraction of on-chip transistors in most modern processors, are a primary candidate for attacking the leakage problem. While there has been a flurry of research in this area over the Last several years, a major question remains unanswered. What is the total potential of existing architectural and circuit techniques to address this important design concern? In this paper, we explore the limits in which existing circuit and architecture technologies may address this growing problem. We first formally propose a parameterized model that can determine the optimal leakage savings based on the perfect knowledge of the address trace. By carefully applying the sleep and drowsy modes, we find that the total leakage power from the L1 instruction cache, data cache, and a unified L2 cache may be reduced to mere 3.6%, 0.9%, and 2.3%, respectively, of the unoptimized case. We further study how such a model can be extended to obtain the optimal leakage power savings for different cache configurations.

Categories and Subject Descriptors: B.3.2 [**Memory Structures**]: Design Styles—*Cache memories*; C.0 [**Computer Systems Organizations - General**]: Modeling of Computer Architecture

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Limits, cache intervals, leakage power

1. INTRODUCTION

Power dissipation has become a major concern to those designing processors for high performance desktops, servers, and battery-operated portable devices. Higher energy dissipation requires more expensive packaging and cooling technology, which in turn increases cost and decreases system reliability. There are fundamentally two ways in which power can be dissipated: either dynamically (due to the switching activity of repeated capacitance charge and discharge on the output of the millions of gates), or statically (mainly due to sub-threshold and gate leakage [Kim et al. 2003; Rabaey et al. 2002]). Dynamic power consumption is proportional to the square of the supply voltage, which reduces as process technology scales. While the scaling down of transistor geometries enables the reduction of the dynamic

Authors' addresses: Y. Meng, R. Kastner, Department of Electrical and Computer Engineering, University of California, Santa Barbara, Santa Barbara, CA 93106-9560; emails: yanmeng@engr.ucsb.edu; kastner@ece.ucsb.edu; T. Sherwood, Department of Computer Science, University of California, Santa Barbara, Santa Barbara, CA 93106-9560; email: sherwood@cs.ucsb.edu

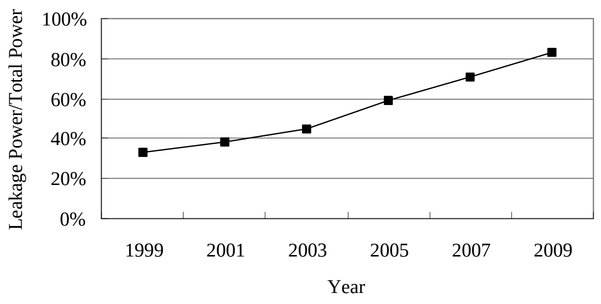


Fig. 1. Projected leakage power consumption as a fraction of the total power consumption according to the International Technology Roadmap for Semiconductors [ITRS].

power, it worsens the leakage problem greatly. If current technology scaling trends hold [ITRS], leakage will soon become the dominant source of power consumption, and as such new techniques are needed to battle this growing problem.

The problem of leakage becomes more significant as threshold voltage, channel length, and gate oxide thickness are reduced. Furthermore, the sub-threshold leakage, as a major component, stems from the need for a trade-off between dynamic power and performance. Scaling down the transistor supply voltage reduces the dynamic power dissipation. Yet, to maintain high switching speed under reduced voltages, the threshold voltage must also be scaled. As the threshold voltage drops, it is easier for current to leak through the transistor resulting in significant leakage power dissipation. The increases in device speed and chip density exacerbate the leakage problem. New technologies targeted at reducing dynamic power and increasing performance, such as low threshold voltage [Liu and Svensson 1993] and gate oxide scaling [Lee et al. 2004], further increase the relative importance of leakage power [ITRS] (Figure 1).

Cache memories have long been used to reduce the ever-growing gap between processors and memory. Modern processors typically provide two levels of on-chip caches (e.g. separate L1 instruction and data caches and a unified L2 cache). In these processors, a large and growing fraction of the total on-chip area, and an even larger fraction of the total number of transistors, is consumed by caches. Because they account for such a significant portion of the total chip real estate, caches provide a healthy-sized target for designers to try circuit and architectural optimizations with the goal of reducing leakage power. The central idea behind most of these techniques is to exploit some form of temporal locality. By putting infrequently or unused cache lines into low leakage mode, much of the power will be reduced. By keeping frequently accessed cache lines active, total performance will not be reduced significantly.

Though there are several circuit techniques and management schemes concerning how and *when* to turn on or off individual cache lines, little work has been done to explore the *limit* of how well such techniques can work. What is the best we could hope to do with a given low power technology? The primary goal of this paper is to explore these limits under different architectural and design assumptions in the hope of guiding research effort on leakage power in much the way that Belady's OPT algorithm [Belady 1966] helps (and continues to help) in the study of replacement policies.

There has been much work on leakage power reduction already, and any proposed methods for calculating the limits of their effectiveness must be both general enough to capture a variety of techniques, yet specific enough to provide useful bounds. Our methods capture both state-preserving and state-destroying techniques, and additionally we show how to optimally combine two such techniques into a hybrid scheme. We show that, given perfect knowledge of the future address trace, there exists a break-even point between Drowsy and Gated-V_{dd}. If the same cache line is accessed twice in an interval of time less than or equal to this break-even point then Drowsy mode should be used. If the same cache line is not used again within an amount of time greater than the break-even point then more power can be saved by turning off the cache line using Gated-V_{dd}. If these timings are known, then an optimal policy can be achieved.

Clearly perfect knowledge of the future trace is not always known, but it serves several purposes. First it provides an important bound. No management method will be able to beat our power reduction scheme under the given circuit assumptions. Second, it demonstrates that there is still a great deal of potential for policy decisions (when to turn a cache line on or off) to significantly reduce leakage power. Finally, while perfect knowledge of future references cannot be known, it can often times be approximated by architecture techniques such as address prediction or *prefetching* [Meng et al. 2005].

In particular, we make the following contributions:

— We relate the potential savings that can be obtained from Drowsy and Gated-V_{dd} techniques, under various assumptions for both the L1 instruction and data caches, and the unified L2 cache.

— We show that with oracle knowledge of future accesses, a simple optimal power management scheme can be derived from a small set of circuit parameters.

— In addition to showing the optimal leakage savings on a set of implementation parameters, we develop a parameterized model to determine the optimal leakage savings while the implementation technologies and architectures change over time.

— We show that while both Drowsy and Gated-V_{dd} schemes are useful on their own, when combined, we can push the upper bounds of the leakage power savings to 96.4%, 99.1%, and 97.7% for the instruction cache, the data cache and the unified L2 cache, respectively, with the 70nm implementation technology.

— We also show that the model can be applicable to explore the limits of leakage power savings for different implementation technologies and cache configurations.

— In addition to the limits study for the L1 instruction and data caches, We study the limits for L2 caches when both sleep and drowsy modes are employed. Taking a large amount of area in modern processors, L2 caches exhibit themselves as another interesting target to battle the leakage problem. Without careful attention to power, L2 caches may overtake the chip's power budget.

— We conduct the leakage study on different cache configurations to validate our methods.

— Instead of examining the leakage reduction on only five benchmarks [Meng et al. 2005], in this study we investigate our methods across all the SPEC2000 benchmark applications.

— We also empirically study the interval distribution to show how much percentage the short dead intervals contributes to the total leakage reduction.

The rest of the paper is organized as follows. We review related work and motivate our limit study in Section 2. In Section 3, we propose our method for combining the Gated-V_{dd} method and the drowsy method. We also explore the limit of leakage power saving that we can potentially achieve using our hybrid scheme. A model which parameterizes all the individual assumptions is also proposed. Section 4 describes our simulation setup and the benchmarks in our study, and shows the results of our empirical study in exploring the upper bounds. We also study the generality of the parameterized model on different cache configurations. We offer concluding remarks in Section 5.

2. CIRCUITS AND ARCHITECTURES OF REDUCED CACHE LEAKAGE POWER

In order to derive a useful limit for leakage power reduction in caches, we must first begin with a discussion of those related technologies so that our model will be grounded in reality. In this section we review several circuit techniques, and develop the general ideas of our approach.

Leakage power comes from transistors that are simply left on, and the easiest way to think about reducing the amount of the consumed leakage power is to "turn off" those transistors that are not needed. While this is the easiest to think about, it is by no means the easiest to implement. One such approach, Gated-V_{dd} [Powell et al. 2001], attempts to solve this problem by reducing leakage through the use of a high threshold sleep transistor (between pull-down NMOS and virtual V_{ss}) to break the connection and thus increases the L1 cache line access time. This transistor is in the read critical path which may impact performance, however we only consider the potential for energy savings in this paper. This leakage reduction technique is often called *sleep mode*, and this is the naming convention that we use here. While efficient in saving leakage, *sleep mode does not preserve the state of the data*. When a cache line is needed again after it has been put to sleep, it must be re-fetched from lower levels of the memory hierarchy. This re-fetch is essentially an extra cache miss, and this process can take many cycles depending on the memory hierarchy, architectural assumptions, etc.

A different way of saving leakage power in the caches is to make use of *multiple supply voltages*. When the cache line is left fully on, it will dissipate too much leakage power. If V_{dd} is fully gated, it will use very little power, but the data is lost. A compromise is to use a *lower* supply voltage when data is not needed for a while. This will reduce the leakage power without losing the data. The trade-off is that, while data will be preserved at this low supply voltage, it cannot be accessed while in this state. Thus there is a small wakeup time associated with changing from the lower voltage up to V_{dd} (hence the name "drowsy"). If this can be implemented without adding a high-Vth transistor in the read critical path as the initial proposal, drowsy mode has the potential to achieve a smaller L1 cache access time than sleep mode. Drowsy mode does not fully turn off the memory, and thus does not reduce the leakage power as much as Gated-V_{dd}. For a piece of data that is not going to be accessed for a very long time, sleep mode will be better because it reduces more leakage power. For a piece of data that is accessed in a moderate amount of time, drowsy mode will be better because there is not a large re-fetch penalty. This sets up one of the fundamental questions answered in our paper — how long is *long enough* for each mode?

While our paper attempts to address a previously unanswered question, there is a great deal of prior work aimed at reducing leakage power in caches. Azizi et al. [Azizi et al. 2003] introduced asymmetric dual-V_t SRAM cell caches(ACCs). ACCs exploit the fact that in ordinary programs most of the bits in caches are zeros for both the data and instruction streams, and provide significant leakage reduction in the zero state. DRI-cache [Powell et al. 2001] uses the Gated-V_{dd} technique to dynamically adjust the size of the active portion of the cache by turning off a bank of cache lines based on the miss rates. DRG-cache [Agarwal et al. 2002] employs Gated-V_{dd} to reduce leakage power by turning off the gated-Ground transistor, while data is restored when the gated-Ground transistor is turned on. DTSRAM [H.Kim and Roy 2002] uses body biasing to separately control the V_t of each cache line. To minimize the energy and delay overhead, a cache line is switched to high V_t when it is not likely to be used anymore. Kaxiras et al. [Hu et al. 2002; Kaxiras et al. 2001] proposed the cache line decay scheme to turn off the cache lines in the dead periods of their cache generations using the Gated-V_{dd} technique. Instead of placing both the tag and the data into the sleep mode, AMC [Zhou et al. 2003] keeps the tag alive and tracks the miss rate with respect to the ideal miss rate. This helps to dynamically adjust the turn-off interval and control the overall performance. Velusamy et al. [Velusamy et al. 2002] used formal feedback-control theory to adaptively adjust the cache decay interval and cache lines are turned off accordingly. Another approach to reducing leakage power is called drowsy cache [Flautner et al. 2002; Kim et al. 2004; 2002], which decreases the supply voltage of idle cache lines. Specifically, all cache lines are periodically placed into drowsy mode. [Kim and Mudge 2004] studied techniques for data retention with lower supply voltage. [Hu et al. 2003] employed drowsy cache to exploit program hot-spots and code sequentiality for instruction cache leakage management. Parikh et al. [Li et al. 2004] compared Gated-V_{dd} and drowsy cache at different L2 latencies with HotLeakage and showed Gated-V_{dd} is superior for a set of faster L2 latencies. Heo [Heo et al. 2002] reduced bitline leakage by leaving bitlines open whose cache banks are not accessed. Hanson [Hanson et al. 2001] found that for L1 caches, MTCMOS, which is a state-preserving technique that operates multiple threshold voltages, outperforms Gated-V_{dd}. In [Li et al. 2003], the authors presented several architectural techniques that exploit the data duplication across the different levels of cache hierarchy. They found that the best strategy in terms of energy and energy-delay product is to place the L2 subblock into a state-preserving mode as soon as its contents are moved to L1 and to reactive it only when it is accessed. Bai et al. [Bai et al. 2005] investigated the impact of T_{ox} and V_{th} on power performance tradeoffs for on-chip caches. In contrast, [Sankaranarayanan and Skadron 2004; Zhang et al.

2002] studied software approaches. [Sankaranarayanan and Skadron 2004] decided the decay interval through profiling and showed that the optimal decay intervals can be estimated with a reasonable degree of accuracy using profiling. [Zhang et al. 2002] studied using compiler to insert power mode instructions that control the voltage for the cache lines to control leakage energy.

All of the above approaches strive to develop a scheme for predicting when a section of the cache should be put into a low power mode. They use some heuristics based on either static analysis or run-time behavior to determine what mode each line should be in. One major open question is: what is the best that these approaches could hope to do? Clearly some of the cache lines will have to be left in a high V_{dd} mode so they can be accessed, but how many and for how long? Are these approaches the ultimate in policy leakage power reduction, or is there still room for improvement?

3. CALCULATING THE LIMITS OF LEAKAGE POWER REDUCTION TECHNIQUES

Now that we have reviewed the circuit and architecture techniques employed to reduce leakage power, we describe how to calculate the savings that could be achieved by an optimal approach.

3.1 Cache Intervals

Our analysis of the leakage power saving limit relies on the idea of breaking up the life time of each cache line into a series of *intervals*. An interval is the time that a cache line rests between two accesses. If an interval is very long then it would be beneficial to put that cache line in sleep mode for the duration of that interval. If an interval is very short, it should be simply left in a high-V_{dd} mode. If an interval is somewhere in the middle, perhaps drowsy mode would be the best.

To illustrate the above situations, let's take a two-level loop example (Figure 2) extracted from a human resource management application. It counts the total number of people employed during a year. In the example, the interval (I_{add}) of the two consecutive accesses to the same instruction *add* depends on the size of the inner loop. When the range of the inner loop variable j is large, the interval I_{add} is long, which indicates the cache line of *add* instruction should be put into sleep mode to save leakage power. And when the range is very small, the interval I_{add} is small, which means this cache line should be left in the high-V_{dd} mode for fast accesses. While the range is in the middle, the drowsy mode should be applied to save leakage power without much performance cost. The idea behind our optimal scheme is to determine what the best policy would be for each interval in the program, and then to apply the appropriate leakage technique to that interval.

In an optimal approach, each interval can be thought of as atomic in the eyes of the optimal policy. With oracle knowledge of the future address traces known (as would be for an optimal approach), there should be no reason to perform any new power saving techniques in the middle of an interval. Instead, the same technique should have been applied for the entire duration of the interval as less power would be consumed with the same penalty (for either wakeup or re-fetch).

One thing to note is the notion of live intervals and dead intervals. A live interval starts when a new memory is brought into the cache frame, and ends after the last access. Between the last access to a line of memory and the time it is evicted from

```
......
int i, j, sum, total;
int low(int);
int high(int);
......
for (total = 0, i = 0; i < 12; i ++)
{
   for (sum = 0, j = low(i); j < high(i); j ++)
      sum += a[j];
   sum *= i;
   add: total += sum;
}
......
```

Fig. 2. The access interval example. The interval length of the consecutive accesses to the *add* instructions depends on the range of the inner loop $|high(i) - low(i)|$.

the cache, it is regarded as dead. Besides turning off cache lines in dead periods as the cache decay scheme does [Kaxiras et al. 2001], our method also explores the live period of a cache generation, which demonstrates great potential for leakage reduction. In fact we found that dead periods did not contribute a large amount of leakage savings in the optimal case, because *any* long interval would be turned off whether live or dead. The only additional savings that are achieved from considering dead intervals are from short dead intervals, of which there are very few. Figure 3 is used to demonstrate such a point. It is drawn based on our experimental setup (see Section 4.1). The x-axis shows the interval length and it is log_2-scaled. The y-axis shows the cumulative percentage of the live intervals over the sum of all live intervals and dead intervals over the sum of all dead intervals of the unified L2 cache for *crafty* and *vortex*. As it can be seen, the curves of the dead-interval-crafty and dead-interval-vortex arise when the interval lengths are large (greater than 2^{20} cycles), and the short dead intervals only contribute an insignificant amount (less than 1%) for the sum of all the dead intervals, which indicates that the short dead intervals contribute little to the leakage power reduction. Thus, for the rest of this paper we ignore the effect of live and dead intervals, and instead concentrate on only the durations of the intervals.

3.2 The Optimal Approach

Our optimal approach works as follows. Given an interval distribution of cache accesses, which can be obtained based on a memory configuration, our optimal approach first classifies each cache access interval into one of the following three types: sleep-mode optimal, drowsy-mode optimal and active-optimal, and applies the appropriate mode on each interval to obtain the optimal leakage power saving. If the size of an interval is very small (i.e. there are multiple consecutive accesses within a short period of time), then it is best to leave the cache line in a fully active (non-power saving) mode. If the size of an interval is long, then the best policy is to completely turn off that cache line (sleep) and then re-fetch it when it is needed

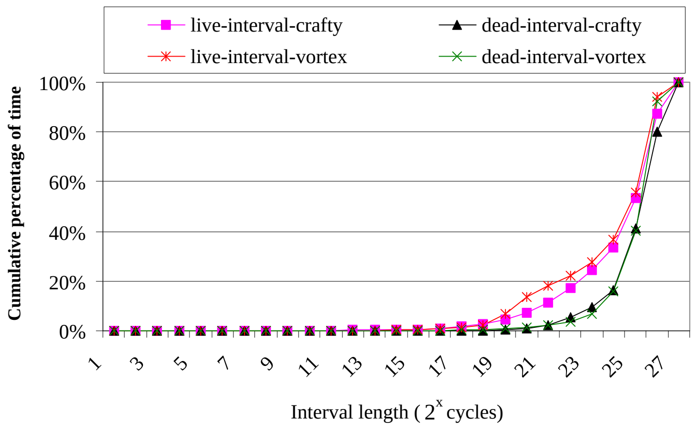


Fig. 3. Cumulative distribution of live intervals and dead intervals of the L2 cache for *crafty* and *vortex*. The total amount of short dead intervals only contribute little to the leakage power reduction, while the long intervals play a major role.

again. The final case is if the interval size is neither very long nor very short. In this case it is best to put the cache line into a drowsy state, which consumes a small amount of power, has a small wakeup cost and has the advantage of retaining the data values.

The key to dividing intervals into these categories is knowing the precise length of an interval that should be put into sleep mode, drowsy mode or left active. The interval length where the power saving mode changes is an *inflection point*. There are two inflection points: one between sleep and drowsy modes and the other between drowsy and active modes.

One thing to note is that our optimal approach will have *no-effect* on the performance of the machine. Because we assume perfect access pattern knowledge, an optimal approach can *re-fetch any needed data just before it is needed* and avoid any performance impact. By exploiting this fact we can separate out the power problems from the performance problems. Even though a just-in-time re-fetch or perfect prefetching will not affect the performance of the machine, it does have a power cost which we do consider in this paper. Figure 4 is used to illustrate this point. In the sleep mode, due to turning off the cache line to save leakage power, the data is not preserved. If the data is accessed again, it needs to be refetched, and this refetching process may usually take several cycles. Without just in time refetch (Figure 4(b)), the other parts of the whole system will have to stall for that amount of cycles, waiting for the data to be ready. The stall will lead to significant energy consumption as the big circle indicates. Similar things happen to the drowsy mode. But the drowsy mode preserves the data and only takes a couple of cycles [Kim et al. 2004] to wake up the cache line. So, without just-in-time refetch (Figure 4(d)), the amount of energy the drowsy mode consumes is less than that of the sleep mode during the system stalling, which is indicated by a small cycle.

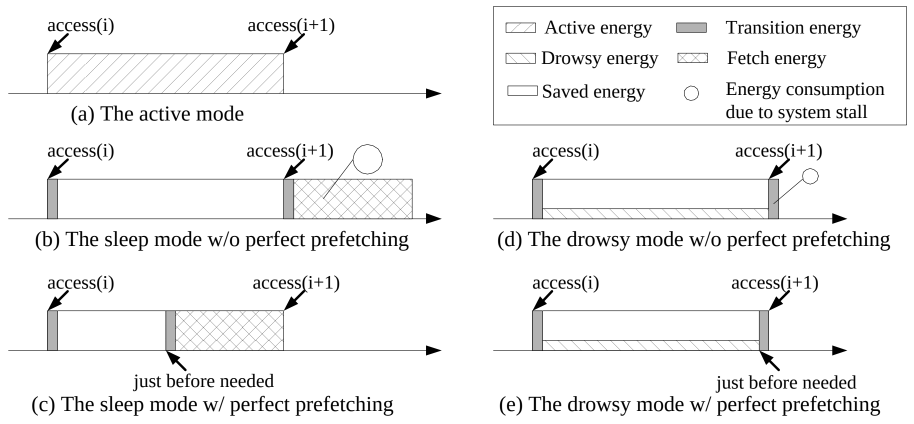
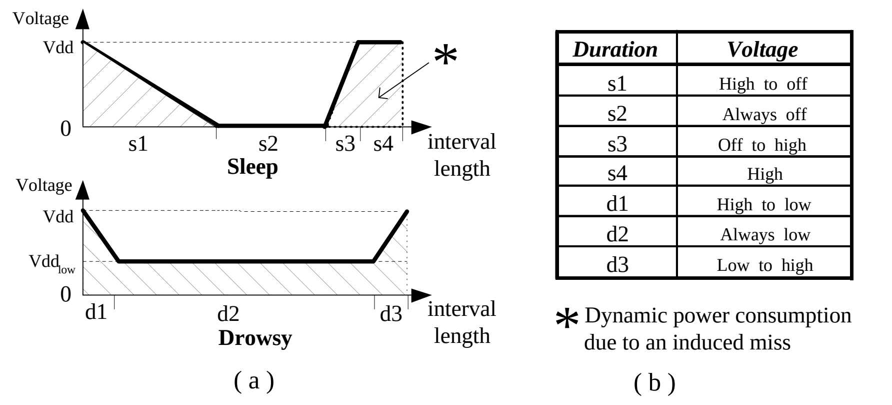

access(i)
access(i+1)
(a) The active mode
Active energy
Transition energy
Drowsy energy
Fetch energy
Saved energy
Energy consumption due to system stall
(b) The sleep mode w/o perfect prefetching
(d) The drowsy mode w/o perfect prefetching
just before needed
(c) The sleep mode w/ perfect prefetching
(e) The drowsy mode w/ perfect prefetching

Fig. 4. Using perfect prefetching to avoid performance degradation. Assuming perfect access pattern knowledge, an optimal approach uses perfect prefetching to refetch data just before it is needed and avoids stalling the whole system to reduce energy consumption.



Duration	*Voltage*
s1	High to off
s2	Always off
s3	Off to high
s4	High
d1	High to low
d2	Always low
d3	Low to high

Fig. 5. Time-voltage diagrams of sleep-mode and drowsy-mode. In Sleep-Mode the cache line is essentially turned completely off and the power consumed drops to nearly zero. While beneficial over a long period of time, there is a more significant overhead due to re-fetch. Drowsy-Mode has a smaller overhead, but the cache line still consumes a measurable amount of power because the voltage has not been completely turned off.

By contrast, with just-in-time refetch (Figure 4(c) and (e)), The data will be ready when it is needed, avoiding stalling the rest parts of the whole system to wait for data to be ready, which consequently saves power. It is worth noting that our scheme calculates the optimal power savings for a given replacement policy, it does not change the replacement policy to further save power.

For the convenience of illustrating how our approach works in general, we will use (Figure 5(a) and (b)) to show how the inflection points are calculated. Figure 5(a) shows that the sleep mode and the drowsy mode require time to reduce the voltage from high V_{dd} to off (s_1) and from high to $V_{dd_{low}}$ (d_1), respectively. Also, there is a similar time overhead in coming out of the mode (s_3 or d_3). For the sleep mode, since the latency D of fetching data from L2 cache is longer than s_3, there is another overhead ($s_4 = D - s_3$) before the next access. We divided the life time of an interval into several durations to illustrate these overheads. Figure 5(b) shows the length of each duration s_1, s_2, s_3, s_4, d_1, d_2, d_3 in an access interval of both a sleep mode and a drowsy mode. The total length of the cache access interval using the sleep technique is $s = s_1 + s_2 + s_3 + s_4$, and that of using the drowsy mode is $d = d_1 + d_2 + d_3$.

For the sleep mode, the data has been lost due to an induced miss [Kaxiras et al. 2001] and must be re-fetched from the memory hierarchy. as such, there is a significant amount of power consumed by the dynamic activity required to fetch the data from the L2 cache, marked with "*" in Figure 5(b). This dynamic power cost can be obtained from analytical models, such as the interconnect model based on logical effort [Amrutur and Horowitz 2001] or the CACTI [Shivakumar and Jouppi 2001] model, which has been used in this paper.

The sleep-drowsy inflection point is derived as the access interval length when the sleep and the drowsy modes consume the same amount of energy. If the interval is of a length less than the inflection point then drowsy mode would be optimal. If it is greater than the inflection point then sleep mode would be optimal. We denote the leakage power consumption of each cache line as P_L, which can be obtained from the HotLeakage tool [Zhang et al. 2003], and the cost of dynamic power due to an induced miss for the sleep mode as C_D. The energy of a sleep mode interval can be calculated as Equation 1:

$$E_S = \sum_{i=1}^{4} P_L(s_i) * s_i + C_D. \tag{1}$$

Similarly the energy consumption using the drowsy model can be calculated as Equation 2:

$$E_D = \sum_{i=1}^{3} P_L(d_i) * d_i. \tag{2}$$

When the two modes consume the same amount of energy, we reach Equation 3:

$$E_S = E_D. \tag{3}$$

Applying the data in Figure 5(b) into Equation 3, we can calculate the sleep-drowsy inflection point.

The other inflection point is between drowsy and active modes. The drowsy-active inflection point is calculated as the sum of the durations d_1 and d_3, within which the voltage changes either from V_{dd} to $V_{dd_{low}}$ or from $V_{dd_{low}}$ to V_{dd}.

Note that the sleep-drowsy inflection point is the point at which sleep mode has the potential to save power of drowsy mode. Sleep mode does not provide benefit at small interval lengths because of the larger penalty associated with coming out of sleep mode (the power of re-fetch) as opposed to drowsy mode. The only way to save power on small interval lengths is to know exactly when the cache line will be accessed again so that it can be brought out of sleep mode before the data is needed. This is how an optimal leakage management scheme would take advantage of it's perfect knowledge.

When an interval between two accesses to the same cache line is longer than the sleep-drowsy inflection point, using sleep mode has the potential to save more leakage power. When an interval is less than the sleep-drowsy inflection point but still greater than the active-drowsy inflection point, the drowsy mode saves more leakage. When its interval length is less than the active-drowsy inflection point, the cache line is always active and cannot have its leakage power reduced without causing a delay in delivering the data.

```
Input: A set of intervals $\mathbf{I}$
Output: Total leakage power saving
optimal_leakage(I)
    total_saving := 0
    $i := 0$
    while ($I_i \in \mathbf{I}$) do
        if ($|I_i| > b$) then
            total_saving := total_saving + sleep_saving($|I_i|$)
        else if ($|I_i| > a$) then
            total_saving := total_saving + drowsy_saving($|I_i|$)
        else
            no leakage power saving can be obtained
        $i := i + 1$
    end do
    return(total_saving)
```

Fig. 6. Algorithm to compute the optimal leakage power saving given an interval distribution. Intervals are classified into one of the three categories based on the drowsy-active inflection point a and the sleep-drowsy inflection point b: $(0, a]$, $(a, b]$, and $(b, +\infty)$. The *Sleep mode* is applied on intervals within the range of $(b, +\infty)$; the *Drowsy mode* is applied on intervals within the range of $(a, b]$; and the cache lines are left on for intervals within the range of $(0, a]$.

Figure 6 details our optimal leakage power saving approach. By classifying cache intervals into the three types and applying to them the appropriate leakage saving mode, the maximal leakage power saving can be obtained as the accumulation of the leakage saving over all access intervals, which provides us an upper bound for optimal leakage power savings. It can be proved that based on the perfect knowledge of the lengths of all intervals, the optimal leakage power saving can be achieved by applying the proper operating mode on each interval.

3.3 Theorem of the Optimal Policy for Leakage Power Saving

In this section, after defining the relevant terms in our study, we provide the theorem of the optimal policy for leakage power saving.

Definition 3.3.1. We define $\mathbf{I}$=$\{I_i\}$ as a set of intervals, and the length of interval I_i as $|I_i|$ ($|I_i| \in (0, +\infty)$).

Definition 3.3.2. For each interval $I_i \in \mathbf{I}$, we define three possible operating modes $T_j \in \mathbf{T}, where \mathbf{T} = \{T_1 = active, T_2 = drowsy, T_3 = sleep\}$, and the leakage energy saving of the interval I_i working in the mode of T_j is defined as $E(I_i, T_j)$.

Definition 3.3.3. We define two inflection points, the active-drowsy inflection point a and the sleep-drowsy inflection point b. The active-drowsy inflection point a is defined as the sum of the durations within which the supply voltage changes either from high to low or from low to high, i.e. $a = d_1 + d_3$. The sleep-drowsy inflection point b is defined as the access interval length when the sleep and the drowsy modes consume the same amount of energy, i.e. $b = s_1 + s_2 + s_3 + s_4 = d_1 + d_2 + d_3$, where $s_i \geq 0$, $d_j \geq 0$ $(i = 1, 2, 3, 4;\ j = 1, 2, 3)$.

LEMMA 3.3.4. *The active-drowsy inflection point a is less than the sleep-drowsy inflection point b.*

PROOF. Because loads have physical capacities, the discharging process takes less amount of time for the voltage dropping from high to low than from high to off, i.e. $d_1 < s_1$. Similarly, the charging process takes less time for increasing the voltage from low to high than from off to high, i.e. $d_3 < s_3$. Since there is no overlapping time between s_i and s_k $(i \neq k,\ i, k = 1, 2, 3, 4)$ and $s_i \geq 0$ $(i = 1, 2, 3, 4)$, we can conclude that the sum of the durations $a = d_1 + d_3 < s_1 + s_3$, and the sleep-drowsy inflection point $b = s_1 + s_2 + s_3 + s_4$ is greater than $s_1 + s_3$. So, a is less than b. Our study based on the 70nm technology process also justifies that a (6 cycles) is less than b (1057 cycles) from the experimental perspective. □

THEOREM 3.3.5. *Under the context of the independent model, where access intervals of a cache block are independent from each other, we assume that for each interval $I_i \in \mathbf{I}$, one and only one of the three operating modes $T_j \in \mathbf{T}$ can be applied for reducing leakage energy consumption based on the following policy:*

(*1*) *When the interval length $|I_i| \in (0, a]$, the active operating mode or non-power saving mode is applied.*
(*2*) *When the interval length $|I_i| \in (a, b]$, the drowsy mode is applied.*
(*3*) *When the interval length $|I_i| \in (b, +\infty)$, the sleep mode is applied.*

Then the maximal leakage saving can be obtained as the combination of the power saving over all intervals $I_i \in \mathbf{I}$, which gives an upper bound for optimal leakage power saving.

PROOF. We prove the theorem by contradiction. We divide the whole range of the interval length $(0, +\infty)$ into three independent portions based on the active-drowsy inflection point a and the sleep-drowsy inflection point b, i.e. $(0, a] \cup (a, b] \cup (b, +\infty)$ (see Lemma 1 that $a < b$). Suppose the energy saving M based on the above assumptions is not maximal, then there must be another energy saving M' that is greater than M, which indicates that there is at least one interval I_i whose operating mode T'_j is different from T_j.

Figure 7 shows the function of interval vs. energy consumption. In the figure, we can have the following derivations:

(1) The function is continuous and monotonically increasing.
(2) The slopes P_1, P_2 and P_3 indicate the power consumptions within the interval ranges of $(0, a]$, $(a, b]$ and $(b, +\infty)$ respectively.

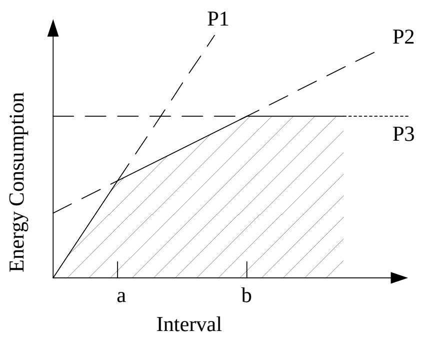


Fig. 7. Energy consumption for each of the three operating modes and the lower envelope $E(I_i, T_j)$ function for minimal energy consumption.

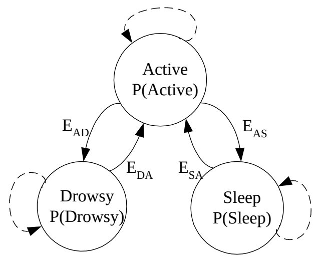


Fig. 8. The optimal leakage power saving model. The circles indicates states and edges represent transitions between states.

(3) For intervals in the range of $(0, a]$, the minimal energy consumption can be achieved through the active mode T_1. For intervals in the range of $(a, b]$, the minimal energy consumption can be achieved through the drowsy mode T_2. For intervals in the range of $(b, +\infty)$, the minimal energy consumption can be achieved through the sleep mode T_3.

For a set of independent intervals, if at least one interval I_i was applied with T'_j, not the corresponding mode T_j, then $E(I_i, T'_j)$ is greater than $E(I_i, T_j)$ (above the shadow area in Figure 7), giving the contradiction. Therefore, the maximal leakage power saving can be obtained by the proposed policy. $\square$

3.4 The Generalized Model for Optimal Leakage Power Savings

After illustrating our optimal leakage power saving approach, we evolve our approach to a complete model that can capture the optimal leakage savings as configurations and technologies change.

As Figure 8 depicts, the model has three states, which are indicated by the circles, representing the three operating modes: Active, Drowsy, and Sleep. It also models the transitions between states, demonstrated by the edges (self edges means that the state remains the same in the next cycle). Each state is associated with its static power consumption (P), and the weights (E_{AD}, E_{AD}, E_{AD}, E_{AD}) on the edges are the transition energy consumptions. For example, E_{AD} is the energy consumption when transiting from the state Active to the state Drowsy.

In the model, all the individual assumptions namely the durations, energy costs of transitions between modes, the leakage power consumption of each mode, and the intervals, are parameterized and used as inputs to the model. And the outputs of the model are the optimal leakage saving percentages of using the optimal sleep, optimal drowsy, and the optimal combining methods.

The model has been designed to explore the optimal leakage savings with parameterized architectural and design considerations, i.e. if the architectural configuration changes(for example, cache configurations), the model can adapt to the input change, and if a new low power mode is employed, the model can be easily extended by adding a new state.

The model for optimal leakage power savings serves two major functions. First, instead of being an abstract model, it is coded in C language and is publicly available for cache leakage studies[1]. To use the tool, designers only need to feed the following inputs into the C program, such as the transition energies obtained from CACTI [Shivakumar and Jouppi 2001], the leakage power consumption from HotLeakage [Zhang et al. 2003], the interval distribution from SimpleScalar [Desikan et al. 2001], and the duration parameters $\{s_1, s_2, s_3, s_4\}$. The tool will then find out an optimal mode transition sequence that can achieve the maximal leakage power savings and output the optimal leakage saving percentages of using the optimal sleep, the optimal drowsy, and the optimal combining methods. Second and the most important is that this model was designed to explore the optimal leakage savings under different architectural and design assumptions with the hope of guiding research effort on leakage power study.

Concurrent to our work, similar approaches are being developed to guide policy decisions in the domain of Computer Aided Design [Liu and Chou 2004]. In [Liu and Chou 2004], the optimal energy mode transition sequence for generic devices is calculated under a fixed delay constraint. Because of the complex timing involved in a modern superscalar microprocessor, a simple timing model will not accurately reflect the impact changing cache parameters. In our study on cache leakage we have factored out any performance impact through perfect prefetching, which acts as a guiding bound for those developing leakage reduction schemes. We have built our model around the state of the art in leakage reduction techniques and show how the most common techniques considered today can be mapped onto this simple framework quickly and precisely. In addition, we have given a formal proof of optimality, which has not been offered in their work.

4. EMPIRICAL STUDY

In Section 3, we discussed the limits of leakage power reduction techniques and how they are calculated. In this section we show limit results gathered from actual benchmarks with parameters extracted from modern processors and prior work. Our objective is to evaluate the limits on some leakage power saving techniques as applied to both the L1 instruction and data caches, and the L2 cache. We show upper bounds on the possible savings using Sleep mode, Drowsy mode, or a

[1]http://express.ece.ucsb.edu/software/leakage.html

potential hybrid of the two. We also evaluate the generality of the parameterized model in deriving the limits of leakage power savings from the perspectives of different implementation technologies and different cache configurations.

4.1 Methodology

To test the amount of power that can be saved by using an improved leakage reduction technique, we employed detailed cycle-level simulation. The simulator we use is a version of SimpleScalar closely resembling Compaq Alpha 21264 [Kessler 1999]. The execution core is a 4-wide superscalar pipeline, and the memory hierarchy includes a 64KB, 2-way set associative L1 instruction cache with a single-cycle hit latency, a 64KB, 2-way set associative L1 data cache with a 3-cycle hit latency, and a unified 2MB direct-mapped L2 cache with a 7-cycle hit latency. The main memory system consists of 16 32MB DDR2 SDRAM chips [electronics 2002], for a total main memory capacity of 512MB, with its access time 40ns, and access power 300mW. Because leakage is exponentially dependent on temperature, we use 85°C in our experiments. LRU is employed as the replacement policy throughout the memory hierarchy. To calculate inflection points, we assumed a 500MHz processor clock.

In order to capture the most important program behaviors while at the same time reducing simulation time to reasonable levels, we used the simulation points that were described and verified in SimPoint [Sherwood et al. 2002]. The benchmark suite for this study consists of all the SPEC2000 benchmarks compiled for the Alpha AXP ISA. Because modern processors typically have two levels of on-chip caches (e.g. separate L1 instruction and data caches and a unified L2 cache), in the rest of our empirical study, we will first explore the limits of the leakage power savings for the L1 instruction and data caches. We will then conduct the limits study for the L2 caches.

4.2 Limits Study for the L1 Instruction and Data Caches

In this section, we explain how the optimal approach works from the experimental aspect in exploring limits of leakage power savings for the modern L1 caches by the ways of how we calculate the inflection points, and how we optimally combine the sleep and drowsy modes.

4.2.1 *Calculating Inflection Points.* Inflection points are the keys to choosing the best power saving mode of a given interval distribution. An intervals with its length greater than the sleep-drowsy inflection point is put into the sleep mode, and an interval with its length less than the active-drowsy inflection point is left in the active mode. For an interval with its length between the two inflection points, it is put into drowsy mode to save power and has little performance impact.

Technology	70nm	100nm	130nm	180nm
Active-Drowsy point	6	6	6	6
Drowsy-Sleep point	1057	5088	10328	103084

Table I. Active-drowsy and drowsy-sleep inflection points depicted in cycles for different technologies.

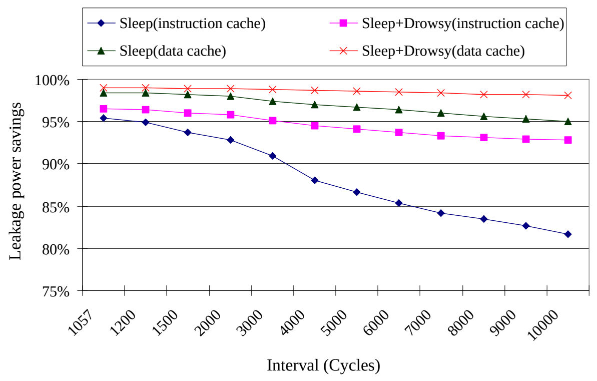
To calculate inflection points with respect to different technologies, we used the durations s_1=30, s_3=d_1=d_3=3 and s_4=4 cycles [Li et al. 2004] (s_2 and d_2 are dependent on an interval length). When we applied the parameters into Equation 1, 2 and 3, we obtained the inflection points for the L1 instruction and data caches shown in Table I. The table shows that the value of the sleep-drowsy point decreases while the technology scales down from 180nm to 70nm (These are the only currently available technologies provided by the Hotleakage [Zhang et al. 2003] tool. If in the future Hotleakage is extended to incorporate more technologies, our approach can still be applied to obtain their inflection points). This is due to the fact that the leakage power consumption per cache line increases while the dynamic energy consumption caused by an induced miss decreases with technology scaling down (see Equation 3).

Since 70nm is the most advanced technology that will be reached in a few years according to ITRS [ITRS], we employed it and its corresponding sleep-drowsy inflection point (1057 cycles) in the rest of our study on the L1 caches.

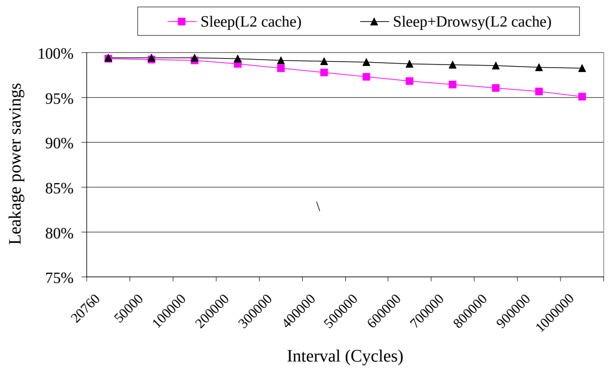
4.2.2 *Combining Sleep and Drowsy Modes.* With the inflection points calculated, the first question to be answered is how well a hybrid of sleep and drowsy modes can perform versus sleep mode. If, in the optimal case for sleep mode, we can perfectly predict the distances between access to cache lines then we can potentially make use of sleep mode even if the cache line is accessed every 1057 cycles. In this case, there will be little benefit from using drowsy mode for those cache lines that are accessed more frequently than every 1057 cycles. However, if the threshold was different, if the inflection point between drowsy and sleep modes changed dramatically, there would be a point at which using both drowsy mode (for occasionally accessed line) and sleep mode (for rarely accessed lines) would become beneficial. The purpose of Figure 9 is to demonstrate this point. In our experiments, when a sleep mode is applied, the dynamic power consumption due to an induced miss was removed from the total leakage power savings.

The results in Figure 9(a) are derived based on the average leakage power savings for both instruction and data caches across all the given benchmarks. Through this figure, we examine the potential effectiveness of a pure sleep mode versus a hybrid sleep/drowsy method where we change the minimum interval length that can be put into sleep mode from 1057 to 10000. These results indicate that a hybrid method (Sleep+Drowsy) can work consistently better than the sleep or the drowsy method alone, especially if one is very conservative about which lines are put to sleep. However, as the minimum sleep length approaches the sleep-drowsy inflection point (decreases), the usefulness of applying the drowsy method in addition to the sleep mode decreases. Under such conditions, the sleep mode removes most of the leakage power and thus there is not much more for drowsy to save. While clearly an implementable scheme will not have the luxury of perfect future knowledge, for those that we do have knowledge for, sleep mode should be applied very aggressively.

Moreover, the figure depicts that the gap between the hybrid method and the sleep mode for the data cache is much smaller than that for the instruction cache. The reason is that the same cache block in the data cache tends to be less frequently accessed than in the instruction cache, and the interval-lengths between consecutive accesses are much longer. Hence, the sleep mode plays a much more important role



(a) L1 Instruction and Data Caches



(b) Unified L2 Cache

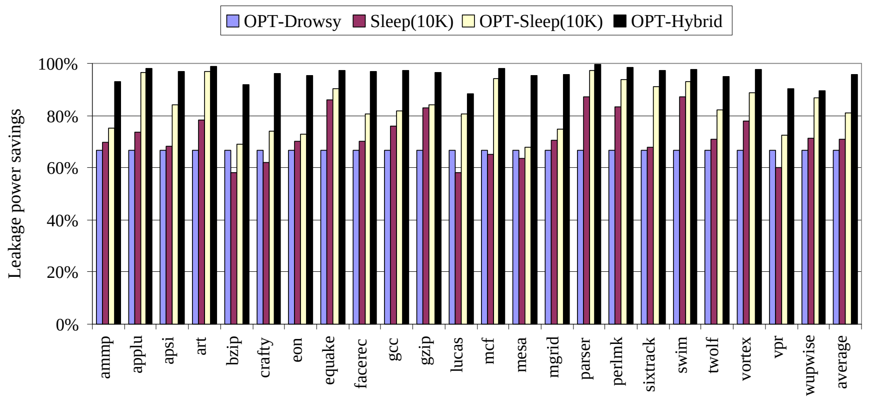
Fig. 9. Comparison of the hybrid method vs. the sleep-mode method for different sleep interval-lengths. The usefulness of applying the drowsy method to save leakage power decreases as the sleep length approaches the sleep-drowsy inflection point. For leakage power saving, the sleep mode plays a more important role in the L2 cache and the data cache than in the instruction cache. The L2 cache has larger sleep intervals than the data cache.

in the data cache for the leakage power saving than in the instruction cache.

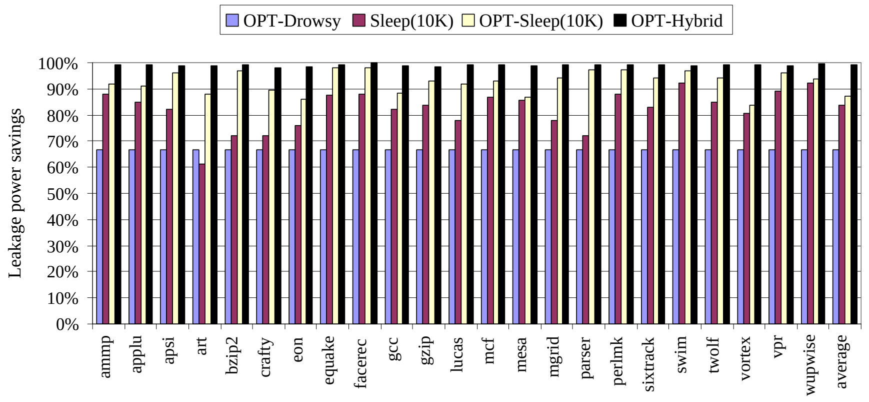
Finally, this figure also confirms that the small variances of the sleep-drowsy inflection point will not change our findings significantly.

4.2.3 *Exploring the Upper Bound.* With the assumption of perfect-prefetching, the upper bound of the leakage power saving was derived in Section 3 based on the two inflection points. We now explore the leakage-power-saving limits of the following methods assuming perfect knowledge of the future address trace:

— *OPT-Drowsy*: An optimal drowsy cache that has no performance penalty for waking up data (although there is a power penalty as discussed in Section 3).



(a) Instruction Cache



(b) Data Cache

Fig. 10. Comparisons of different leakage power saving schemes.

— *OPT-Sleep(10K)*: An optimal cache line sleeping technique that puts to sleep all intervals of a size greater than 10K with no performance penalty.

— *Sleep(10k)*[2]: Similar to the OPT-Sleep(10K) with the exception that instead of optimally turning off any cache line that has an interval larger than 10K, the line must now stay active for 10K and then may be optimally slept.

— *OPT-Hybrid*: The method that optimally combines drowsy and sleep modes based on the inflection points without any performance penalty.

For the convenience of discussing the implementation of each technique, we define an access interval of a cache line as T_i. OPT-Drowsy puts the line into the drowsy

[2]The sleep(10K) is similar to the cache-decay scheme in [Kaxiras et al. 2001], in which the decay interval was set to be 10K cycles, and the extra leakage power consumed by the counter per cache line was taken into account.

mode during T_i, if T_i is greater than 6; while OPT-Sleep(10K) puts the cache line into the sleep mode during T_i if T_i is greater than 10K. We also studied Sleep(10K) to simulate the cache-decay scheme, whose decay interval is 10K. In this case, a cache line is put into the sleep mode for (T_i-10K) cycles if T_i is greater than 10K. The OPT-Hybrid is to put a cache line into the sleep mode during T_i if T_i is greater than 1057, and to put it into the drowsy mode if T_i falls into the range of (6, 1057]. When T_i is less than 6, all the above methods keep the cache line active to insure fast access time. When a sleep mode applied, the dynamic power consumption due to an induced miss was removed from the total leakage power savings.

Figure 10 depicts the percent of leakage power in comparison with a cache with all its cache lines constantly active for each benchmark application. Figure 10(a) shows that for the instruction cache, the limit of leakage power saving that OPT-Hybrid can achieve is 96.4%. It is 26% higher than Sleep(10K), 16% higher than OPT-Sleep(10K), and 30% higher than OPT-Drowsy. For the data cache (Figure 10(b)), the leakage power saving limit is 99.1%, which is 15% higher than the Sleep(10K), 12% higher than the OPT-Sleep(10K), and 33% higher than the OPT-Drowsy. The results indicate that while the initial Drowsy and Sleep techniques devised are quite effective, there is still far more potential left in these techniques. Indeed, the leakage power savings for the optimal case are so large that it is fair to say that leakage power would become an insignificant portion of the total overall power if these savings could be realized. All these savings could be realized with new policies for cache management. Of course realizing these optimal numbers requires perfect knowledge of the address trace and timing, which is not typically possessed by a management policy.

4.3 Limit Study for the Unified L2 Cache

In modern processors, a large number of the total on-chip transistors is consumed by caches, particularly L2 caches. Different from L1 caches that are optimized for performance, L2 caches are optimized for density and stability considering both yield and process parameter variation, and its memory-cell size is usually smaller than a L1 SRAM cell. The area overhead of implementing either drowsy or sleep technique in L2 caches is about 5-8%, and sleep mode may cause instability of L2 memory cells[Kim et al. 2004]. However, as machines and working-sets grow, the L2 caches are becoming increasingly performance critical, and integrated even closer to the processor (especially when a third level of hierarchy is added). In the future it is likely that L2 caches will be heavily accessed by multiple processors, and will be close enough to the logic that they may well be affected by heat dissipation problems. For example even today, the L2 cache covers 37% of the alpha 21364 chip area and contains 85% of the total devices [et al. 2002]. In addition, L2 caches have much larger miss penalty than L1 caches, because they have to go to main memory to fetch data when L2 misses happen. Thus, without careful attention to power, L2 caches may overwhelm the chip's power budget. In this section, we study the leakage reduction in L2 caches when both sleep and drowsy modes are employed.

For applications where the access frequency of L2 caches is small, using high-Vth transistors in memory cells will significantly reduce leakage power. Yet, for applications with large code and data footprints, more levels of cache hierarchy will

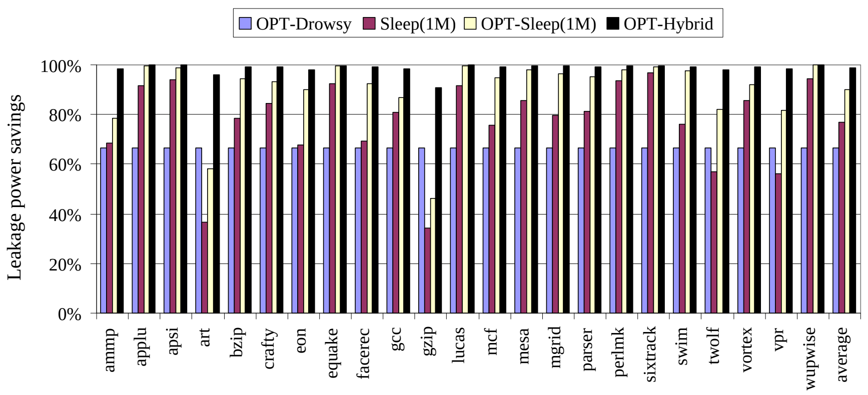


Fig. 11. Comparisons of different leakage power saving schemes for the direct-mapped L2 cache.

be needed to take advantage of locality, e.g. the Intel Madison processor [Madison] has L2 caches which are frequently accessed. In those applications, solely applying high-Vth transistors in memory cells will reduce leakage but may lead to significant performance impact, and dynamic management polices could be employed to trade off power and performance. In this work, we assume two level caches, while our optimal approach can be extended to study general multilevel caches. With the most advanced 70nm implementation technology, we calculated the optimal sleep-drowsy inflection point for the L2 cache as 20760 cycles, and the drowsy-active point as 6 cycles.

Figure 9(c) shows the average leakage power savings of all the benchmarks. It demonstrates how well the hybrid of sleep and drowsy modes can perform versus sleep mode. From the figure, we can see that if we can perfectly predict the access intervals of the L2 cache, there will be little benefit from using drowsy mode for those cache lines that are more frequently accessed than 20760 cycles. Figure 9(c) looks similar to Figure 9(b), however, they are different in scales. The L2 cache has much larger sleep-drowsy inflection point than the data cache(Figure 9(b)), since the miss penalty of the L2 cache is much larger than that of the on-chip data cache.

Figure 11 shows the comparison results of the four methods across all the benchmarks, *OPT-Drowsy*, *OPT-Sleep(1M)*, *Sleep(1M)*[3], and *OPT-Hybrid*. The figure shows that for the L2 cache, the limit of leakage power saving that OPT-Hybrid can achieve is 97.7%. It is 21% higher than Sleep(1M), 7.6% higher than OPT-Sleep(1M), and 31% higher than OPT-Drowsy, which indicates that there is still far more potential left in the existing techniques.

[3]The sleep(1M) is similar to the cache-decay scheme in [Kaxiras et al. 2001], in which the decay interval was set to be 1M cycles, and the extra leakage power consumed by the counter per cache line was taken into account.

	Technology	70nm	100nm	130nm	180nm
	Vdd (V)	0.9	1.0	1.5	2.0
	Vth (V)	0.1902	0.2607	0.3353	0.3979
I-Cache	OPT-Drowsy (%)	66.4	66.6	66.6	66.7
	OPT-Sleep (%)	95.2	85	80.6	61.5
	OPT-Hybrid (%)	96.4	93.7	91.3	67.1
D-Cache	OPT-Drowsy (%)	66.1	66.6	66.7	66.7
	OPT-Sleep (%)	98.4	96.9	95.3	63.2
	OPT-Hybrid (%)	99.1	98.1	97.3	67.3

Table II. Optimal leakage saving percentages with technology scaling down.

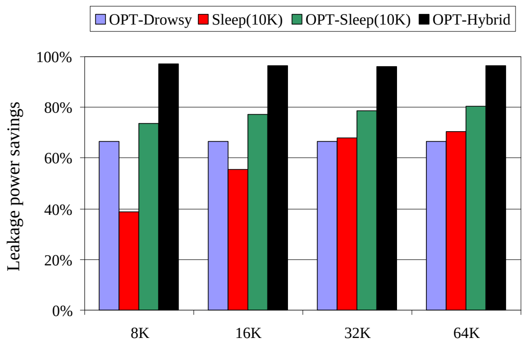
4.4 Empirical Study with the Generalized Model

Also, we evaluate the generality of the proposed parameterized model from two perspectives, one with different implementation technologies and the other with different cache configurations.

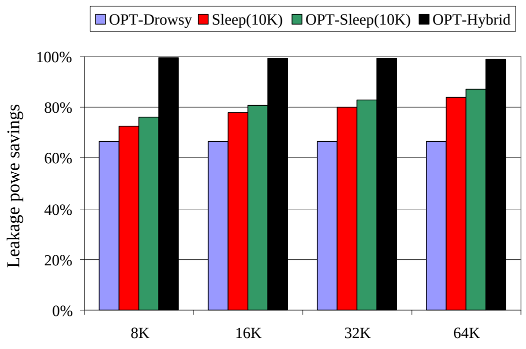
4.4.1 *Evaluating the Generality of the Parameterized Model with Different Implementation Technologies.* To show the generality of the parameterized model, we also study the L1 instruction and data caches with 100nm, 130nm and 180nm processes. Table II summarizes the optimal leakage saving percentages we can possibly achieve by using OPT-Drowsy, OPT-Sleep, and OPT-Hybrid methods for each of these technologies. Instead of using OPT-Sleep(10K) on intervals that are greater than 10K cycles, we study OPT-Sleep to figure out what is the best leakage power saving we can achieve by aggressively turning off all intervals that are greater than the sleep-drowsy inflection point. The OPT-Drowsy and OPT-Hybrid methods are the same as before. The results in the table are the average results over all the benchmark applications.

The table illustrates that the leakage savings for both the instruction and the data caches of using OPT-Hybrid increase with the technology scaling down from 180nm to 70nm. The increment of the possible leakage savings is due to the decrement of the sleep-drowsy inflection point. Moreover, the table shows that for the 180nm technology implementation, the drowsy mode plays a more important role in saving leakage power than the sleep mode does; while for the others, the sleep mode plays a leader role. This can be also attributed to the large difference of the sleep-drowsy inflection points. Finally, the table also reveals that more leakage savings can be possibly achieved with the technology scaling down, which leaves us more space for further improvement on leakage power savings.

4.4.2 *Evaluating the Generality of the Parameterized Model with Different Cache Configurations.* To further evaluate the generality of the parameterized model on studying the limits of leakage power reduction, we also conducted experiments on L1 caches with different sizes, while the rest of the configurations remain the same. Specifically, we studied 8KB, 16KB, and 32KB 2-way set associative instruction and data caches with one-cycle latency. Figures 12 shows the on-average results of different L1 caches. The results demonstrate that as the L1 cache sizes grow larger, the percentage of the leakage power saving for each scheme (OPT-Drowsy,



(a) L1 Instruction Caches



(b) L1 Data Cache

Fig. 12. Comparison of different leakage power saving schemes for L1 caches with different sizes.

Sleep(10K), OPT-Sleep(10K) and OPT-Hybrid) increases, and the drowsy mode plays a more important role for smaller caches than the sleep mode does. This is due to the fact that using smaller caches usually results in more frequent cache misses, which lead to many small intervals (less than 10K cycles) due to frequent replacements. Because of the existence of the small intervals, the drowsy mode shares a significant portion in reducing leakage power.

In addition to study the leakage problem with different L1 cache sizes, we also experimented with another configuration that has a different L2 cache configuration. This configuration includes a 32KB 2-way set associative instruction cache with a single-cycle hit latency, a 32KB 2-way set associative data cache with a two-cycle hit latency, and a unified 1M 2-way set associative L2 cache with five-cycle hit latency. The rest of the configuration, such as the main memory, replacement policy and implementation technology, are the same as the previous configurations.

Figure 13 shows the comparison results of the smaller L2 cache for different leakage power saving techniques. For the purpose of comparison with the previous cache

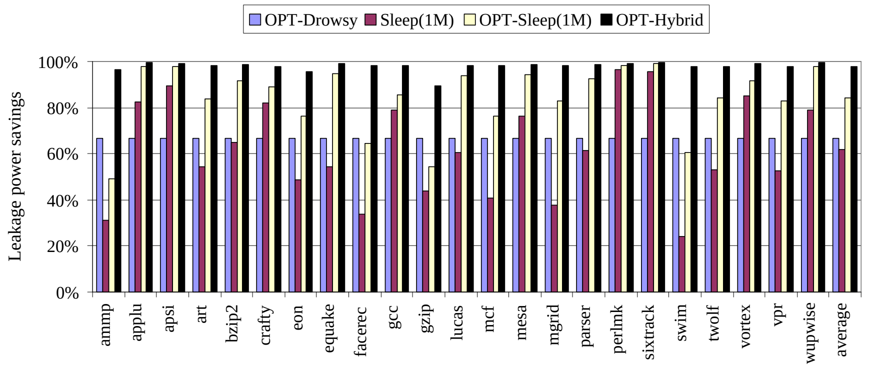


Fig. 13. Comparisons of different leakage power saving schemes for the new 2-way associative L2 cache.

configuration, the results here also include the results of all benchmarks and their average. Note that due to the different cache configurations, the baseline of the total leakage power consumption of the new configuration (Figure 13) is different from that of the previous configuration (Figure 10), even though the y-axis also indicates the percent of total leakage power of which all lines remain constantly active. In our experiments, we found that the total power averaged over all benchmarks for the previous configuration was twice that of the new configuration.

From Figure 10 and Figure 13, we can make three observations. First, for OPT-Drowsy, Sleep(1M) and OPT-Sleep(1M), there are still much room left for further exploring circuit and architectural techniques to achieve the maximal leakage power savings that OPT-Hybrid provides. Second, the percentages of leakage power savings for Sleep(1M) and OPT-Sleep(1M) on the new L2 cache are smaller than those on the previous L2 cache. Third, the results of *ammp* and *mcf* clearly show that OPT-Drowsy achieves more leakage power savings than Sleep(1M) and OPT-Sleep(1M) with the new configuration, while OPT-Drowsy saves less leakage power than either Sleep(1M) or OPT-Sleep(1M). The explanation for the above observations is that the new cache configuration has smaller L1 caches than the previous configuration, which will generally result in a larger number of accesses to the L2 cache and smaller access intervals for the L2 cache lines. Since most of the power savings are from the long intervals, the new cache configuration consequently only achieves less power savings than the previous configuration. This confirms that the parameterized model can be generalized to provide limits of leakage power reduction for different cache configurations.

We would also like to mention that we had an initial exploration [Meng et al. 2005] of using a form of prefetching, such as next-line and stride-based techniques, to approximate the optimal. The goal of prefetching is to accurately predict future access patterns so that they can optimistically fetched from memory before their use. We propose that prefetching can optimistically re-fetch data that has been either turned off for sleep mode or put into a drowsy state. In our trial study, we delivered the information of what is the best those employed prefetching techniques can help us to approach the optimal, and we thus did not take into consideration the overhead of prefetching. Our evaluation of the potential usefulness of next-line and stride-based prefetching toward reducing leakage power on the L1 instruction and data caches shows that if prefetching can be used to guide sleep mode and drowsy mode is used the other times then the leakage power dissipation will be within a factor of 2.5 from the optimal.

5. CONCLUSIONS

Leakage power dissipation is quickly becoming a major concern in designing high performance processors. In this paper we explore the limits to which known circuit level techniques can be combined and employed to save cache leakage power using new management methods and protocols. In addition, we developed a parameterized model to determine the optimal leakage savings while the implementation technology changes over time. We find that it is possible with perfect knowledge of the future address trace to reduce the amount of power dissipated by the instruction cache down by a factor of 5.3 from known techniques (2 for the data cache, and 10 for the unified L2 cache). At this level, the leakage power of the cache would become a less serious problem. Through the evaluation of generality of the parameterized model, we found the model is robustly applicable to different caches, which will provide helpful guidance for further researches in cache-level leakage power reduction.

ACKNOWLEDGMENTS

The authors would like to thank reviewers for their valuable feedback on the manuscript.

REFERENCES

Agarwal, A., Li, H., and Roy, K. 2002. Drg-cache: a data retention gated-ground cache for low power. In *Proceedings of Design Automation Conference (DAC 2002).*

Amrutur, B. S. and Horowitz, M. A. 2001. Fast low-power decoders for rams. *IEEE Journal of Solid-State Circuits 36,* 10, 1506–1515.

Azizi, N., Najm, F. N., and Moshovos, A. 2003. Low-leakage asymmetric-cell sram. *IEEE Transactions on Very Large Scale Integration Systems 11,* 4 (Aug.).

Bai, R., Kim, N. S., Mudge, T., and Sylvester, D. 2005. Power-performance trade-offs in nanometer-scale multi-level caches considering total leakage. In *Proceedings of Design, Automation and Test in Europe (DATE 2005).*

Belady, L. 1966. A study of replacement of algorithms for a virtual storage computer. *IBM Systems Journal 5,* 2, 78–101.

Desikan, R., Burger, D., Keckler, S. W., and Austin, T. M. 2001. Sim-alpha: a validated execution driven alpha 21264 simulator. Tech. Rep. TR-01-23, Department of Computer Sciences, University of Texas at Austin.

electronics, S. 2002. Ddr2sdram datasheet mr16r1622.

et al., J. G. 2002. Power and cad considerations for the 1.75mbyte, 1.2ghz l2 cache on the alpha 21364 cpu. In *Proceedings of GLSVLSI'02.* New York, NY.

Flautner, K., Kim, N., Martin, S., Blaauw, D., and Mudge, T. 2002. Drowsy caches: simple techniques for reducting leakage power. In *Proceedings of International Symposium on Computer Architecture (ISCA 2002).* Anchorage, AK.

Hanson, H., Agarwal, Hrishikesh, M., Keckler, S., and Burger, D. 2001. Static energy reduction techniques for microprocessor caches.

HEO, S., BARR, K., HAMPTON, M., AND ASANOVIC, K. 2002. Dynamic fine-grain leakage reduction using leakage-biased bitlines. In *Proceedings of International Symposium on Computer Architecture (ISCA 2002)*. Anchorage, Alaska.

H.KIM AND ROY, K. 2002. Dynamic vt sram's for low leakage. In *Proceedings of ACM International Symposium on Low Power Design (ISLPED 2002)*.

HU, J. S., NADGIR, A., VIJAYKRISHNAN, N., IRWIN, M. J., AND KANDEMIR, M. 2003. Exploiting program hotspots and code sequentiality for instruction cache leakage management. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED 2003)*. Seoul, Korea.

HU, Z., KAXIRAS, S., AND MARTONOSI, M. 2002. Let caches decay: Reducing leakage energy via exploitation of cache generational behavior. *ACM Transaction on Computer Systems*.

ITRS. International technology roadmap for semiconductors. http://public.itrs.net.

KAXIRAS, S., HU, Z., AND MARTONOSI, M. 2001. Cache decay: exploiting generational behavior to reduce cache leakage power. In *International Symposium on Computer Architecture (ISCA 2001)*. Gőteborg, Sweden.

KESSLER, R. 1999. The alpha 21264 microprocessor'. In *Proceedings of IEEE Micro*. 24–36.

KIM, N., FLAUTNER, K., BLAAUW, D., AND MUDGE, T. 2002. Drowsy instruction cache: leakage power reduction using dynamic voltage scaling and cache sub-bank prediction. In *ACM/IEEE International Symposium on Microarchitecture (MICRO 2002)*. Istanbul, Turkey.

KIM, N., FLAUTNER, K., BLAAUW, D., AND MUDGE, T. 2004. Circuit and microarchitectural techniques for reducing cache leakage power. *IEEE Transaction on Very Large Scale Integration Systems 12,* 2 (Feb.), 167–184.

KIM, N. AND MUDGE, T. 2004. Single vdd and single vt super-drowsy techniques for low-leakage high-performance instruction caches. In *Proceedings of International Symposium on Low Power Electronics and Design(ISLPED 2004)*. Newport Beach, CA.

KIM, N. S., AUSTIN, T., BLAAUW, D., MUDGE, T., FLAUTNER, K., HU, J., IRWIN, M., KANDEMI, M., AND VIJAYKRISHNAN, N. 2003. Leakage current: Moore's law meets static power. *Computer 36,* 12 (Dec.).

LEE, D., BLAAUW, D., AND SYLVESTER, D. 2004. Gate oxide leakage current analysis and reduction for vlsi circuits. *IEEE Transactions on Very Large Scale Integration Systems 12,* 2 (Feb.).

LI, L., KADAYIF, I., TSAI, Y., VIJAYKRISHNAN, N., KANDEMIR, M., IRWIN, M. J., AND SIVASUBRAMANIAM, A. 2003. Managing leakage energy in cache hierarchies. *Journal of Instruction-level Parallelism 5*.

LI, Y., PARIKH, D., ZHANG, Y., SANKARANARAYANAN, K., SKADRON, K., AND STAN, M. 2004. State-preserving vs. non-state-preserving leakage control in caches. In *Proceedings of the 2004 Design, Automation and Test in Europe Conference(DATE 2004)*.

LIU, D. AND SVENSSON, C. 1993. Trading speed for low power by choice of supply and threshold voltages. *IEEE Journal of Solid State Circuits 28,* 1 (Jan.).

LIU, J. AND CHOU, P. 2004. Optimizing mode transition sequences in idle intervals for component-level and system-level energy minimization. San Jose, CA.

Madison. Intel madison processor. http://www.intel.com.

MENG, Y., SHERWOOD, T., AND KASTNER, R. 2005. On the limits of leakage power reduction in caches. In *Proceedings of International Symposium on High-Performance Computer Architecture(HPCA-11)*. San Francisco, CA.

POWELL, M., YANG, S., FALSAFI, B., ROY, K., AND VIJAYKUMAR, T. N. 2001. Reducing leakage in a high-performance deep-submicron instruction cache. *IEEE Transactions on Very Large Scale Integration Systems 9,* 1 (Feb.).

RABAEY, J., CHANDRAKASAN, A., AND NIKOLIC, B. 2002. *Digital Integrated Circuits A Design Perspective(2nd Edition)*. Prentice-Hall.

SANKARANARAYANAN, K. AND SKADRON, K. 2004. Profile-based adaptation for cache decay. *ACM Transactions on Architecture and Code Optimization 1,* 3 (Sep.).

SHERWOOD, T., PERELMAN, E., HAMERLY, G., AND CALDER, B. 2002. Automatically characterizing large scale program behavior. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2002)*. San Jose, CA.

SHIVAKUMAR, P. AND JOUPPI, N. 2001. Cacti 3.0: An integrated cache timing, power, and area model. Tech. Rep. WRL-2001-2, HP Labs Technical Reports. Dec.

VELUSAMY, S., SANKARANARAYANAN, K., PARIKH, D., ABDELZAHER, T., AND SKADRON, K. 2002. Adaptive cache decay using formal feedback control. In *Proceedings of 2002 Workshop on Memory Performance Issues in conjunction with ISCA-29*. Anchorage, Alaska.

ZHANG, W., HU, J. S., DEGALAHAL, V., KANDEMIR, M., VIJAYKRISHNAN, N., AND IRWIN, M. J. 2002. Complier-directed instruction cache leakage optimization. In *Proceedings of International Symposium on Microarchitecture (MICRO 2002)*. Istanbul, Turkey.

ZHANG, Y., PARIKH, D., SANKARANARAYANAN, K., SKADRON, K., AND STAN, M. R. 2003. Hotleakage: An architectural, temperature-aware model of subthreshold and gate leakage. Tech. Rep. Tech. Report CS-2003-05, Department of Computer Sciences, University of Virginia. Mar.

ZHOU, H., TOBUREN, M. C., ROTENBERG, E., AND CONTE, T. M. 2003. Adaptive mode control: a static power-efficient cache design. *ACM Transcactions on Embedded Computing Systems 2,* 3 (Aug.).