

Using Coherence Information and Decay Techniques to Optimize L2 Cache Leakage in CMPs

Matteo Monchiero¹
matteo.monchiero@hp.com
¹Exascale Computing Lab
HP Labs
Palo Alto, CA 94034

Ramon Canal²
rcanal@ac.upc.edu
²Dept. of Computer Architecture
Universitat Politècnica de Catalunya
Barcelona, Spain 08034

Antonio González^{2,3}
antonio.gonzalez@intel.com
³Intel Barcelona Research Center
Intel Labs - UPC
Barcelona, Spain 08034

Abstract—This paper evaluates several techniques to save leakage in CMP L2 caches by selectively switching off the less used lines. We primarily focus on private snoopy L2 caches. In this case, coherence must be enforced in all situations and specially when a line is turned off to save power. In particular, we introduce three techniques: the first one turns off the cache lines by using the coherence protocol invalidations, the second one is an implementation of a cache decay technique specific for coherent caches, the third one is a performance-optimized decay-based technique for coherent caches.

Experimental results, carried out by using accurate performance/thermal/energy models, show that appreciable power savings can be achieved by properly designing a leakage optimization technique. We target a CMP composed of 4 cores and 1 to 8 MB of total cache. For 4MB, the proposed techniques show a 13%, 30%, and 21% energy reduction, respectively, at the cost of 0%, 8%, and 2% performance loss. For other cache sizes the behavior is qualitatively similar.

Keywords—leakage; chip multiprocessor; CMP; Multicore; Cache Decay; Coherence;

I. INTRODUCTION

Static power is one of the most important sources of power consumption in modern processors. This is mainly due to the subthreshold and gate-oxide leakage currents which are intrinsically high in deep sub-micron devices. Efforts in high-k materials have reduced the gate-oxide leakage; nevertheless lower voltages increase dramatically subthreshold leakage. Both academic and industrial studies show that a large part of the power consumption of a state-of-the-art processor is due to the static component [1]–[3]. According to the ITRS [4], efficiently managing low-leakage devices and operation modes is a key challenge for next generation systems.

Multicore architectures have recently emerged as a well established trend in processor design. This approach – aka Chip Multiprocessor (CMP) – permits to efficiently deal with power/thermal issues dominating advanced processes and makes it easy to exploit thread level parallelism of modern applications.

In this context, the memory hierarchy design is critical to both performance and power. This paper faces the problem of optimizing leakage energy in the secondary (L2) caches of CMP systems. L2 caches are typically much larger than L1 caches, featuring larger static power contribution.

Many literature papers have proposed techniques to reduce static power in the caches of single-processor systems. Most of these techniques are based on the common idea of selectively turning off (or switch to a low-leakage state) a portion of the cache [5]–[8]. A SRAM cell may be switched off by using a transistor to gate the power supply or the ground path. A quite successful low-leakage cache microarchitecture, first introduced in [6], is based on the concept of *line decay* (or *cache decay*). According to this technique, a cache line is allowed to be on and inactive – i.e. not accessed – at maximum for a given time, named *decay time*. A counter is associated to each line. Once a time equal to the *decay time* has elapsed, if the line has not been accessed, the line is turned off. Otherwise, when the line is accessed, the counter is reset to its initial value (*decay time*). Cache decay has been mostly studied in the context of L1 caches and only recently has been considered also for L2 caches [9], [10].

This paper aims at proposing several leakage-aware schemes for secondary caches in CMPs. We focus on private snoopy L2 caches, since these present more interesting challenges. In this case, data coherence must be enforced when a line is turned off. In detail, the main contributions can be summarized as follows: i) a technique which uses the coherence protocol to switch off L2 cache lines; ii) an implementation of a cache decay technique for a private (snoopy) L2 cache; iv) a technique (*Selective Decay*), to improve the performance of the decay scheme by avoiding to decay lines which may generate ‘costly’ side-effects, such as misses in the primary caches.

This paper is organized as follows. Section II presents the related work. In Section III, we discuss the implementation of a turning off mechanism suitable for L2 caches of CMPs. The techniques explored in this paper, based on the mechanism of Section III, are described in Section IV. Section V, presents the experimental framework. Section VI, discusses the results. Finally, Section VII concludes the paper.

II. RELATED WORK

The problem of reducing the energy consumption of cache memories has been thoroughly faced in past years. A

plethora of techniques to minimize the dynamic consumption was proposed: subbanking, line buffers, filter cache, bitline segmentation, cache re-configuration, . . . We therefore focus on those works targeted at static power reduction.

High- V_{th} cells are typically used to build low-leakage circuit blocks [11], but leakage saving comes at the cost of slower speed. This approach can be adopted for caches as well. In particular, it is attractive for L2 caches since they are not frequently accessed. Nevertheless, the current trend makes L2 caches smaller, possibly backed up by a large L3 cache, and often tightly bound with the primary caches. We think this argument motivates the need for low-leakage techniques for high-speed L2 caches.

Powell *et al.* [5] discuss several *Gated- V_{dd}* techniques to turn off a portion of caches. They show that the leakage power of a gated cell can be reduced virtually to zero, at the cost of 8% slower access time and 5% area increase. They also propose a dynamically resizable instruction cache.

In [7], Flautner *et al.* introduce the Drowsy Cache, whose idea is to put the cache lines into a state preserving, low-power drowsy mode. Other state-preserving techniques have been proposed in [9], [12]. This may be achieved by lowering the V_{dd} . It may involve two power supply voltages, as in [7], or a single one [12]. Regardless of the number of supply voltages, the noise margin of the cell is reduced. Since precise and stable threshold voltage is difficult to achieve in deep submicron processes, this makes the cells more susceptible to soft errors and metastability. Furthermore, in [9], several architectural techniques that exploit the data duplication across the different levels of cache hierarchy are proposed and compared to multi-level decay techniques both exploiting state-preserving and state-destroying mechanisms. For the reasons above, this work focuses only on state-destroying techniques (i.e. decay).

Kaxiras *et al.* [6], [13], propose two mechanisms of cache decay: *Fixed Decay Interval* and *Adaptive Decay Interval*. The first one assumes a fixed decay time for the whole cache and program execution. The latter increases the decay interval – for a single line – if a miss occurs soon after a line have been switched off. On the other hand, if a miss takes place a long time after turning off the line, the decay is decreased.

Zhou *et al.* [8] introduces another decay-based technique, named *Adaptive Mode Control*. In this case, a global decay interval exists for the whole cache, but this may be dynamically modified on the basis of a periodic sampling of the miss rate.

In [10], Abella *et al.* specifically target L2 caches. They propose a predictor which suggests a decay time by monitoring the *time between hits* for a cache line. Zhang *et al.* [14] use the compiler to determine the regions of code not used in the near future and thus deciding the lines to be turned-off. Finally, several works [15], [16] combine both state-preserving (drowsy) and state-destroying (decay) techniques in a single cache to optimize leakage reduction.

In this paper, we consider only state-destroying techniques

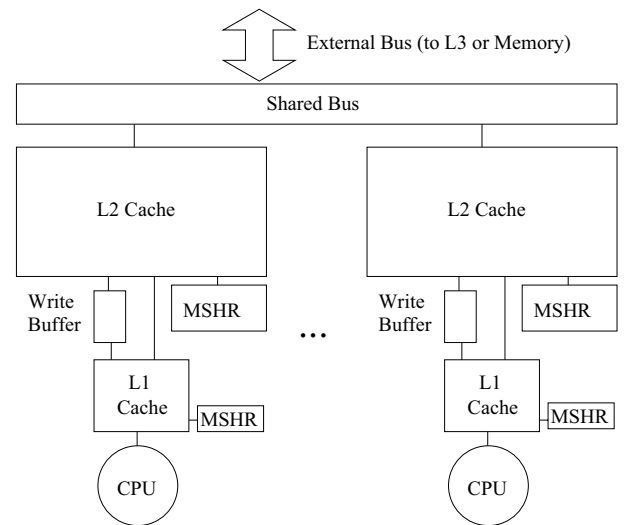


Figure 1. Private L2 CMP architecture

(i.e. the line is switched off). In particular, we assume *Gated- V_{dd}* [5], as enabling technology for any turn-off technique. In addition, We focus on fixed decay techniques. Plus, most of the work presented in this section is orthogonal to our approach and can be easily combined.

Self-invalidation [17], [18] was proposed to early invalidate active lines. In this way, the invalidation overhead under sequential consistency is reduced. The authors do not analyze the energy behavior of these techniques, which could be used in conjunction with some of our proposal to save leakage by exploiting protocol invalidates.

This work, differently from many previous work [6], [8], [10], uses a detailed temperature-dependent leakage model to carry out energy estimation.

Many literature works have appeared about power/performance optimization of multicore architectures. Most of them focus on the exploration of the design space, in terms of number of cores, processor complexity, and memory hierarchy [3], [19]–[21]. Dynamic Voltage and Frequency Scaling (DVFS), in the context of CMPs, has been discussed as well [20], [21].

Dynamic cache resizing have been proposed by Intel starting from the Centrino Duo [22]. Anyway, the resizing can be performed only in the C4 state, which corresponds to periods of very low utilization and idle residency.

This paper contribution differentiates from the previous ones. We face the problem of minimizing the static energy in CMPs caches, analyzing issues of cache coherence, and proposing a detailed evaluation of several leakage saving techniques.

III. IMPLEMENTING A MECHANISM TO TURN OFF L2 CACHE LINES IN A CMP

When dealing with a shared memory multiprocessor with coherent private-L2 caches, some issues may arise if a L2 line is simply eliminated. L2 caches, in a system like the one

Table I
SUMMARY OF THE VARIOUS SITUATIONS RELATED TO LINE STATE AND POSSIBILITY OF TURNING OFF

		Single processor or shared L2		Multiprocessor – private L2
		L1 is Write-Back	L1 is Write-Through	L1 is Write-Through
L2 line state	Clean	Turn off	Turn off, if no pending write	Turn off, if no pending write
	Dirty	Write back and turn off	Turn off, if no pending write, and write back	Turn off, but invalidate the upper level

sketched in Figure 1, are typically inclusive. This means that the lines in the L1 caches must be also in the L2. If this does not happen, any invalidation of a data residing only in the L1 will not be transmitted by the L2. Even if more sophisticated techniques can be devised [23], in this paper, we consider that L2 is inclusive, and to facilitate inclusion, the L1 cache is Write-Through. This solution stems for ease of design, while others may reveal additional complexity.

Figure 1 shows some more details of the system that we consider: Both L1 and L2 own a MSHR, which allows that multiple hits are served under a pending miss. The primary cache uses a Write Buffer to propagate writes. We assume L2 caches are snoopy and coherence is enforced on the shared bus – and propagated to the upper level. We consider a MESI snoopy protocol.

Any possible shared cache behaves as a coherent device. In this case, many issues existing for private caches do not arise.

The goal of this section is the definition of an architectural primitive capable of switching off a line of a secondary cache, without violating the consistency of the cache hierarchy. Issues related to when this should happen (the global scheme/algorithm used to turn off cache lines) are discussed in the next section.

We compare the multiprocessor case with the uni-processor. Table I compares the various situations that may arise, and whether a line may (or may not) be switched off. We assume that an external *turn-off* signal exists for each block and, when raised, it means to switch off that block. Depending on the state of the block in the L2, and the corresponding one in the L1, some conditions must be met. In detail, for the uni-processor:

- *L1 is Write-Back*. If the block in the L2 is clean, it may be evicted. The corresponding block in the L1 may be clean or dirty, but this does not make any difference. If clean, the block will be discarded. If dirty, the new value(s) will be possibly allocated in the L2 (depending on the write policy) – and thus the old values need not to be kept. Should the line size among L1 and L2 be different, the L2 (or the subsequent level of the memory hierarchy) should allow partial writes. If the block in the L2 is dirty, the L2 line can be turned off, and the newest copy of the dirty block (which can be either at L1 or L2) must be written back to the memory.
- *L1 is Write-Through*. If the block in the L2 is clean, it may be evicted. In this case, the corresponding block in the L1, should be clean. Nevertheless, the L1 write

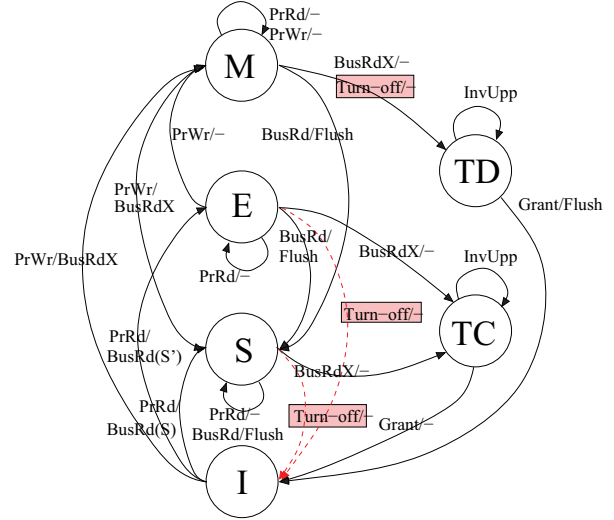


Figure 2. How to modify the MESI protocol to deal with external *turn-off* signal. *TC* and *TD* are transient states for a line being invalidated in the upper level. Labeling of the diagram edges is similar to [23]

buffer should be checked whether the L1 is writing that block to the L2.

Otherwise, if the L2 line is dirty, this means that the L1 line is dirty too, and the two values are coherent (unless there is a pending write). Also in this case the L2 block can be evicted (updating the memory), without touching the L1.

If the L2 cache is a snoopy cache (private-L2 multiprocessor), we must take care of the cache consistency when turning off a line. If the line is clean, this means that the corresponding line in the L1 is clean too (Write-through), so there is no problem in switching off the line. Otherwise, if the line is dirty, the line maybe turned off, but the L1 line must be evicted. This is because inclusion must be maintained.

Figure 2 shows a modified version of the MESI protocol which accounts for the possibility of turning off lines. The diagram includes two states, *TC* (*Transient Clean*) and *TD* (*Transient Dirty*), which are transient states for a clean or dirty line, being invalidated in the upper level. This extension belongs to the traditional implementation of multilevel protocols.

We marked with dashed lines additional edges, and we labeled with shadowed boxes additional conditions/actions. The turn-off signal may trigger a state transition only from a ‘stationary’ state, that is *Modified*, *Exclusive*, *Shared*. If the line is in any transient state, it must wait to reach the next

stationary state. Other transient states are not shown in the diagram, for the sake of clarity, but exist to deal with non atomic bus transactions [23]. Once this condition is satisfied, depending on the state of the line the following actions may take place:

- The line is *Modified*. Invalidation in the upper level is needed. So next state is *TD*. Once invalidation has been performed, the line can be switched off. This transition also causes a write-back to the memory.
- The line is *Shared* or *exclusive*. The line can be turned off.

The mechanism to physically turn off a line is implemented by using the valid bit, as explained in [6]. Once a line is invalidated, its valid bit goes to a low voltage level and triggers the circuitry to isolate the selected line from the power supply path. In the diagram of Figure 2, a line is effectively switched off when it goes to the *Invalid* state.

This technique may be easily extended to any coherence protocol, of course taking care of the different semantic of the states. For example, considering the *Owned* state of the MOESI, other copies must be invalidated before a line is turned off.

Note that, depending on the semantic of the state, switching off a line may be more or less costly, possibly implying further invalidations. For what concerns MESI (see Figure 2), turning off a *Modified* line generates a write-back and invalidation in the upper level. On the other hand, *Shared/Exclusive* lines don't incur in any penalty.

IV. EXPLORED TECHNIQUES

We propose and explore three techniques based on the turning off mechanism illustrated in Section III, for a private-L2 CMP.

In details:

- *Turn off on Protocol Invalidation*. The base protocol is used (Figure 2 without extensions). A cache line is switched off when a line is invalidated. This technique does not incur in any performance loss, since the natural behavior of the cache is maintained.
- *Decay*. We implemented a fixed decay technique [6] on the top of our turn off mechanism. The diagram of Figure 2, with extensions, describes this technique. A line is turned off either because a line is invalidated or a *turn-off* signal is generated by the decay logic.
- *Selective Decay (SD)*. This is also a decay technique, but a line is let to decay on the transitions leading to a *Shared* or *Exclusive* state. The idea is to avoid costly decay in terms of performance. As we discussed in Section III, *Modified* lines, if turned off, need to invalidate the upper level. This may cause a non trivial performance loss, since it affects directly L1 performance. By activating the decay only on the selected transitions, we minimize the possibility that one of the decaying lines will goes to *Modified*.

V. EXPERIMENTAL SETUP

To accurately evaluate performance and power of CMPs architecture, we integrated a microarchitecture simulator modeling a CMP, power models, and a temperature modeling tool. In particular, we set up an accurate leakage estimation framework, accounting for temperature and physical characteristics of the chip.

The CMP simulator is SESC [24]. Each core is an out-of-order superscalar processor, with private L1 caches and private L2. The processor microarchitecture models the Alpha 21264 [25].

Inter-processor communication develops on a high-bandwidth shared bus (57 GB/s), pipelined and clocked at half of the core clock. The coherence protocol acts directly among the L2 caches, and it is MESI snoopy-based. At the same time, the protocol requires additional invalidates/writes between L1 and L2 caches, to ensure the data coherence. Memory ordering is ruled by a weak consistency model.

The power model integrated in the simulator is based on Watch [26] for processor structures, CACTI [27] for caches, and Orion [28] for buses. The thermal model is based on Hotspot-3.0.2 [29], while the leakage model is based on the work by Liao *et al.* [30].

Line decay techniques have been implemented assuming a hierarchical counter architecture [6]. We accounted for extra dynamic and leakage energy due to additional structures. Since Gated- V_{dd} needs 5% increased area, we consider this overhead when calculating the leakage energy. Furthermore, we considered one cycle penalty for caches employing decay [5]. Nevertheless, this comes up to be a not appreciable contribution to the total execution time. Following the methodology in [10], we have not taken into account the energy consumption due to the extra off-chip accesses. Nevertheless, we evaluate the extra bandwidth needed for each technique.

We selected 3 scientific applications from Splash-2 suite [31] (WATER-NS, FMM, and VOLREND), and 3 benchmarks from ALPbench [32] (mpeg2enc, mpeg2dec, facerec) as benchmarks. All benchmarks have been run up to completion and statistics have been collected on the whole program run, after skipping initialization. For thermal simulations, we used the power trace related to the whole benchmark simulation (dumped every 10000 cycles). We used standard data set for Splash-2, while we limited to 10 frames for the MPEG2. For facerec, we use the collection of images included in the distribution.

VI. EXPERIMENTAL RESULTS

This section presents the evaluation of the proposed techniques for L2-private 4-core CMPs, and 1, 2, 4, 8 MB total L2 cache¹.

We first analyze results related to average values across the benchmarks, and different total cache sizes. Figure 3(a) shows the occupation rate of the L2 caches for the different

¹That is, the size of every single L2 cache is 256KB, 512KB, 1MB, 2MB

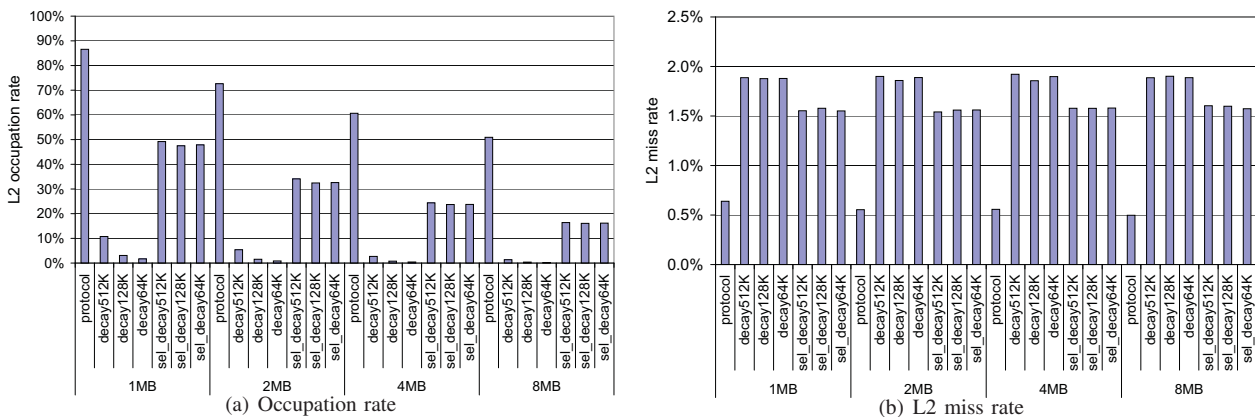


Figure 3. L2 occupation and miss rate. *protocol* is the architecture which uses the protocol to turn off lines; *decayN* is a fixed decay technique, and *sel_decayN* is the selective decay technique, where N is the decay time in K-cycles

techniques. This is an aggregate figure, calculated over all the L2 caches of the system. It indicates the average fraction of time that a cache line is turned-on during the execution of a program. For example, if the occupation rate is 60%, this means that the lines of the L2 caches have been turned on for the 60% of the time, on average.

A mathematical definition is as follows (given a leakage saving technique):

$$\frac{\sum_{j \in L2s} \sum_{i \in lines} on_cycles_{ij}}{\#L2s \times \#lines \times total_cycles}$$

where on_cycles_{ij} are the on-cycles (turned on line) of the i -th line of the j -th cache. $\#L2s$ is the number of L2 cache in the system, $\#lines$ is the number of lines per cache, and $total_cycles$ is the number of cycles to execute, that is, how long a cache line is active when no optimization technique is applied.

The architecture without any leakage optimization technique (occupation rate is 100% – always on) is assumed to be the baseline.

As Figure 3(a) shows, the *Protocol* technique behaves fairly well, especially, for larger caches. The L2 occupation ranges from 87% to 50%. Of course, since the workload is fixed for various cache sizes, the occupation rate decreases as the size increases. The *Decay* technique features low cache occupation, ranging from 10% to less than 1%. For this technique a quite large decay time (512K cycles) may be enough to achieve significant savings. The *Selective Decay* technique is in the middle, occupancy rates are on the range of 50% to 18%. In this case, because only some lines are allowed to decay (those going to *Shared* or *Exclusive* states), potential savings are reduced.

Figure 3(b) shows the aggregate miss rate for the L2 caches. You can see that the miss rate is quite low for all the configurations. This is because the private-L2 architecture features Write-Through L1s, making that the operations on the L2 are mostly writes. In addition, the miss rate is not sensitive to cache size, because the misses induced by the decay techniques (which are the largest component) do

not depend on the cache size, but mostly on the specific technique. On the other hand, for what concerns the *Protocol* technique, the miss rate slightly improves, since there are no extra-misses. Furthermore, when comparing the the three techniques for a given cache, it is easy to see that more aggressive the decay is, the higher the miss rate becomes: 0.5%, 1.5%, and nearly 2% for *Protocol* (and the baseline), *Selective Decay*, and *Decay*, respectively.

The increase of the memory bandwidth needed for each configuration is shown in Figure 4(a). This can be as large as 200% for the *Decay* technique and 8MB cache size. Bandwidth requirements may be non trivial when considering decay-based techniques, since these generate costly L2 misses. This trend is twice less for the *Selective Decay* technique, which may represent a cost-effective solution. On the opposite, if only relying on the protocol to turn off lines, no additional memory accesses are required.

In Figure 4(b), we consider the performance of the whole memory system, showing the variation of the Average Memory Access Time (AMAT). Decay-based techniques may worsen it by 10% on average. In this case, by using the *Selective Decay* a 10% better AMAT is achieved, even if this benefit seems to diminish as the cache size increases.

Finally, we evaluate the impact of the proposed techniques on system² energy and performance. In Figure 5(a), the energy reduction achieved by every technique is shown. Results are relative to unoptimized case (L2 caches always on). The amount of saved energy heavily depends on the size of the cache. This is because the optimized fraction (the L2 leakage) depends on the cache size too. For the *Decay* technique the energy savings are 9%, 17%, 30%, and 43% for each cache size from 1MB to 8 MB. The *Selective Decay* technique typically saves the 50-90% of the energy saved by *Decay*.

It is interesting to note that as the cache size increases, the two techniques perform similarly. Notice that the magnitude of the decay time is only slightly influential. The addi-

²The system is composed of cores L1, L2 and system bus

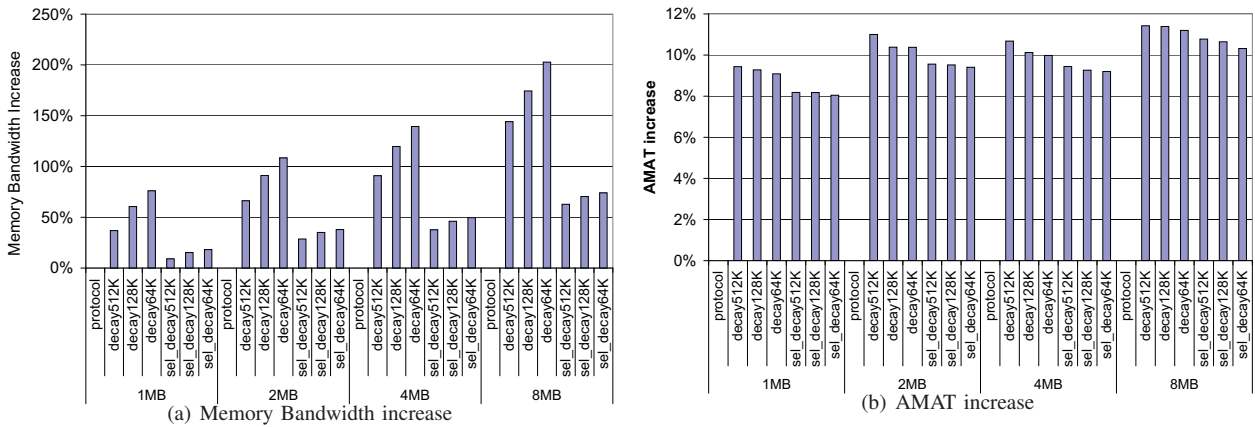


Figure 4. Variation of the Memory Bandwidth and AMAT, w.r.t. the unoptimized case

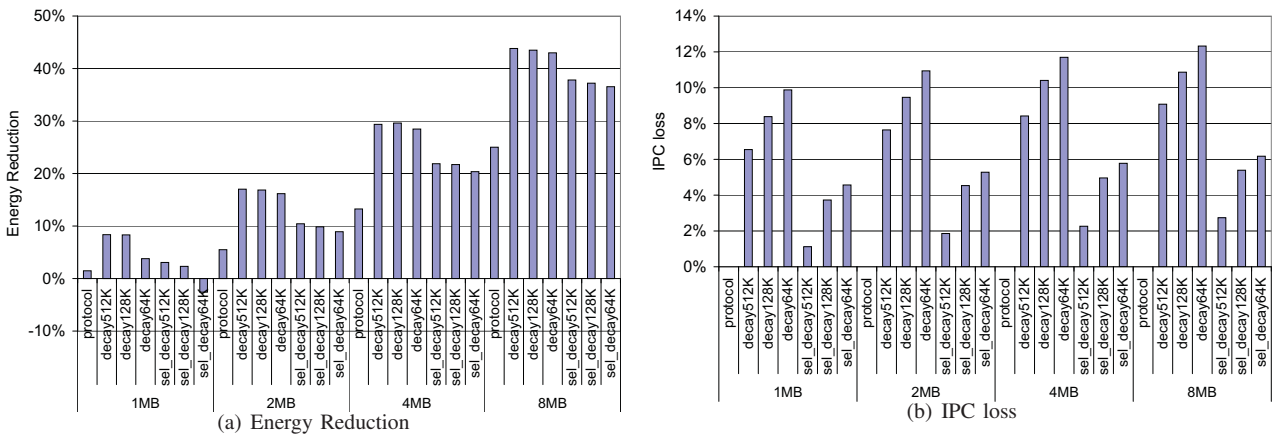


Figure 5. Variation of Energy and IPC

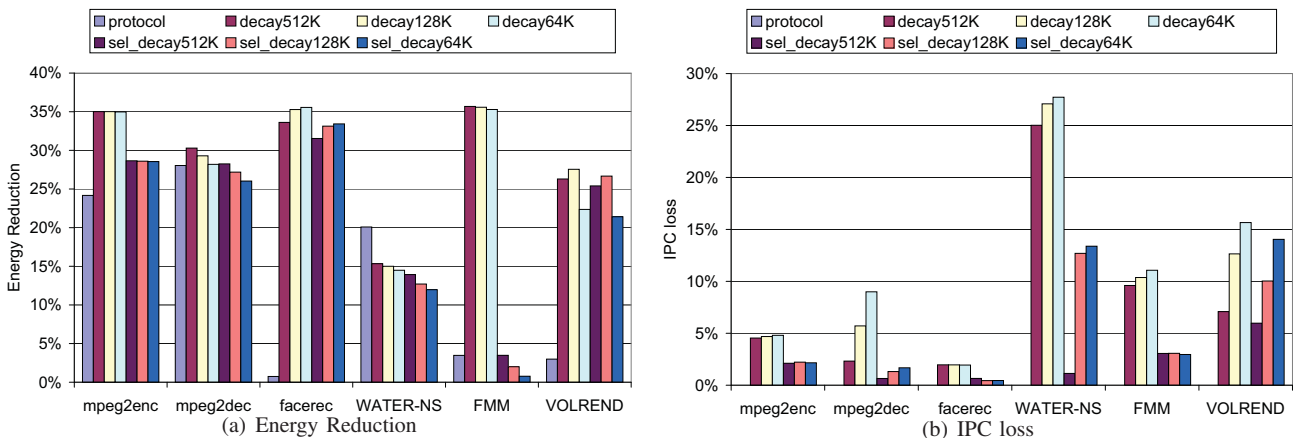


Figure 6. Energy and IPC for different benchmarks, given 4MB total cache size

tional saving is compensated by increased dynamic energy. Moreover, for *Selective Decay* 4K-decay time/1MB-cache size, the optimized architecture consumes more than the baseline. This happens because of the steady increase of dynamic energy for some benchmarks, when the decay time is reduced to 64K-cycles.

The *Protocol* technique features lower energy savings: approximately half with respect to *Decay*, but still significant, especially for larger caches. For example for 4MB, it is 13%.

Figure 5(b) shows the IPC loss for each technique. You can observe that, unlike energy, the decay time has a remarkable impact on the IPC (40-50%). So, larger decay

time might be a better choice from the Energy-Delay point of view. The *Selective Decay* technique features improved IPC with respect to *Decay*. If comparing *Decay* 512K-decay time (less aggressive), and *Selective Decay* 64K-decay time (most aggressive), *Selective Decay* achieves 75% lower IPC penalty than decay, while featuring 25% less energy saving (see 4MB-L2, in Figure 5(b)).

In Figure 6(a) and Figure 6(b), energy reduction and IPC loss, for each simulated benchmark, are reported. For the sake of conciseness, we focus on 4MB total cache systems. Following considerations are qualitatively similar for other cache configurations.

Since the characteristics of the benchmarks are different for what concerns the L2 usage, results are pretty heterogeneous. You may observe that the *Protocol* technique is sometimes as good as the decay-based ones. This happens for *mpeg2dec* (only slightly worse than *Decay*), while it performs better for *WATER-NS*. The *Selective Decay* achieves typically the same saving as *Decay*, except for *mpeg2enc* and *FMM*. For the latter it is clearly outperformed by *Decay*.

IPC (Figure 6(b)) seems worsen much more for scientific benchmarks (*WATER-NS*, *FMM*, and *VOLREND*) than from multimedia ones. The selected scientific benchmarks may feature more complex cache access patterns, difficult to capture with simple decay techniques.

The effects of varying the decay time are also different. For example, larger decay improves significantly IPC for *VOLREND* and *mpeg2dec*, while it has a smaller effect for the others. Moreover the effects on the energy may result in an increase (e.g. *mpeg2dec*) or a decrease (*facerec*), depending on the dynamic energy overhead.

In conclusion, the *Protocol* technique may be a cost effective choice for a subset of the considered applications (25-35% energy reduction, and no performance loss). When targeting multimedia applications, the *Selective Decay* provides nearly the same energy saving as the more aggressive *Decay* – less than 5% worse – at the cost of a minimal IPC loss (less than 2% on average). On the other hand, for scientific applications, decay techniques may lead to appreciable IPC penalty (up to more than 25%). Nevertheless, a careful choice of a leakage saving technique may lead to 15-35% energy savings.

VII. CONCLUSIONS

This paper explored several techniques to reduce the leakage energy in CMPs. We adapted the coherence protocol to cope with power decisions such as turning off a line. Furthermore, we adapted existing decay techniques to work properly on a CMP. In this paper, we presented three leakage saving techniques: i) *Turn off on Protocol Invalidation (Protocol)*: this is based on the coherence protocol to turn off the lines invalidated by the protocol itself; ii) *Decay*; this is an implementation of a cache decay technique, compatible with the CMP cache hierarchy; iii) *Selective Decay*: another decay-based technique, but performance-optimized, since it aims at avoiding costly decays in terms of performance.

The evaluation showed that for a total of 4MB L2 cache CMP, these techniques feature 13%, 30%, and 21% energy reduction, respectively, at the cost of 0%, 8%, and 2% performance loss. While the performance penalty is approximately the same for different cache sizes, the energy reduction is higher for larger caches – up to 25%, 44%, and 38%, respectively, for 8MB. When analyzing the behavior of each scheme for a mix of scientific and multimedia benchmarks, we found that *Protocol* and *Selective Decay* feature 25%/35% energy reduction at the cost of minimal performance loss (0% or 2%, respectively), for most of the applications. Especially when targeting multimedia applications, *Selective Decay* seems the best solution for Energy-Delay.

ACKNOWLEDGMENT

This work has been supported by the Generalitat de Catalunya under grant 2005SGR00950 and the Spanish Ministry of Science and Innovation under grant TIN2007-61763.

REFERENCES

- [1] A. Grove, “Changing vectors of Moore’s law – Keynote speech,” in *IEDM’02: Int. Electron Devices Meeting*, 2002. [Online]. Available: http://www.intel.com/pressroom/archive/speeches/grove_20021210.htm
- [2] Y. Meng, T. Sherwood, and R. Kastner, “Exploring the limits of leakage power reduction in caches,” *ACM Trans. Archit. Code Optim.*, vol. 2, no. 3, pp. 221–246, 2005.
- [3] M. Monchiero, R. Canal, and A. Gonzalez, “Design space exploration for multicore architectures: A power/performance/thermal view,” in *ICS06: Proceedings of the 20th annual international conference on Supercomputing*. New York, NY, USA: ACM Press, 2006.
- [4] ITRS, “The international technology roadmap for semiconductor: 2005.” [Online]. Available: <http://www.itrs.net/Common/2007ITRS/Home2007.htm>
- [5] M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, “Gated-vdd: a circuit technique to reduce leakage in deep-submicron cache memories,” in *ISLPED ’00: Proceedings of the 2000 international symposium on Low power electronics and design*. New York, NY, USA: ACM Press, 2000, pp. 90–95.
- [6] S. Kaxiras, Z. Hu, and M. Martonosi, “Cache decay: exploiting generational behavior to reduce cache leakage power,” in *ISCA ’01: Proceedings of the 28th annual international symposium on Computer architecture*. New York, NY, USA: ACM Press, 2001, pp. 240–251.
- [7] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, “Drowsy caches: simple techniques for reducing leakage power,” in *ISCA ’02: Proceedings of the 29th annual international symposium on Computer architecture*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 148–157.

- [8] H. Zhou, M. C. Toburen, E. Rotenberg, and T. M. Conte, "Adaptive mode control: A static-power-efficient cache design," *Trans. on Embedded Computing Sys.*, vol. 2, no. 3, pp. 347–372, 2003.
- [9] L. Li, I. Kadayif, Y.-F. Tsai, N. Vijaykrishnan, M. T. Kandemir, M. J. Irwin, and A. Sivasubramaniam, "Leakage energy management in cache hierarchies," in *PACT '02: Proceedings of the 2002 International Conference on Parallel Architectures and Compilation Techniques*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 131–140.
- [10] J. Abella, A. Gonzalez, X. Vera, and M. F. P. O'Boyle, "Iatac: A smart predictor to turn-off l2 cache lines," *ACM Trans. Archit. Code Optim.*, vol. 2, no. 1, pp. 55–77, 2005.
- [11] A. Chandrakasan and R. W. Brodersen, *Low-Power CMOS Design*. Wiley-IEEE Press, 1998.
- [12] N. S. Kim, K. Flautner, D. Blaauw, and T. Mudge, "Single-vdd and single-vt super-drowsy techniques for low-leakage high-performance instruction caches," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 54–57.
- [13] Z. Hu, S. Kaxiras, and M. Martonosi, "Let caches decay: reducing leakage energy via exploitation of cache generational behavior," *ACM Trans. Comput. Syst.*, vol. 20, no. 2, pp. 161–190, 2002.
- [14] W. Zhang and J. Hu, "Compiler-directed instruction cache leakage optimization," in *Proc. of the 35th annual ACM/IEEE International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 2002.
- [15] T. S. Y. Meng and R. Kastner, "On the limits of leakage power reduction in caches," in *HPCA '05: Proceedings of the 11th International Symposium on High-Performance Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2005.
- [16] M. Ghosh and H. Lee, "Virtual exclusion: An architectural approach to reducing leakage energy in caches for multiprocessor systems," *2007 International Conference on Parallel and Distributed Systems*, vol. 2, pp. 1–8, Dec. 2007.
- [17] A.-C. Lai and B. Falsafi, "Selective, accurate, and timely self-invalidation using last-touch prediction," in *ISCA '00: Proceedings of the 27th annual international symposium on Computer architecture*. New York, NY, USA: ACM Press, 2000, pp. 139–148.
- [18] A. R. Lebeck and D. A. Wood, "Dynamic self-invalidation: reducing coherence overhead in shared-memory multiprocessors," in *ISCA '95: Proceedings of the 22nd annual international symposium on Computer architecture*. New York, NY, USA: ACM Press, 1995, pp. 48–59.
- [19] M. Ekman and P. Stenstrom, "Performance and power impact of issue-width in chip-multiprocessor cores," in *ICPP'03: Proceedings of the 2003 International Conference on Parallel Processing*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 359–369.
- [20] J. Li and J. Martinez, "Power-performance implications of thread-level parallelism on chip multiprocessors," in *ISPASS'05: Proceedings of the 2005 International Symposium on Performance Analysis of Systems and Software*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 124–134.
- [21] Y. Li, B. Lee, D. Brooks, Z. Hu, and K. Skadron, "CMP design space exploration subject to physical constraints," in *HPCA '06: Proceedings of the 12th International Symposium on High Performance Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2006.
- [22] A. Naveh, E. Rotem, A. Mendelson, S. Gochman, R. Chabukswar, K. Krishnan, and A. Kumar, "Power and thermal management in the intel core duo processor," *Intel Technology Journal*, vol. 10, no. 2, pp. 109–122, 2006.
- [23] D. E. Culler, A. Gupta, and J. P. Singh, *Parallel Computer Architecture: A Hardware/Software Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997.
- [24] J. Renau, B. Fraguera, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos, "SESC simulator," January 2005, <http://sesc.sourceforge.net>.
- [25] R. E. Kessler, "The Alpha 21264 microprocessor," *IEEE Micro*, vol. 19, no. 2, pp. 24–36, March/April 1999.
- [26] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proc. of the 27th International Symposium on Computer Architecture*. ACM Press, 2000, pp. 83–94.
- [27] P. Shivakumar and N. P. Jouppi, "CACTI 3.0: An integrated cache timing, power, and area model," Western Research Laboratory, Compaq, Tech. Rep. 2001/2, 2001.
- [28] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Proc. of the 35th annual ACM/IEEE International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society Press, 2002, pp. 294–305.
- [29] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 94–125, 2004.
- [30] W. Liao, L. He, and K. Lepak, "Temperature and supply Voltage aware performance and power modeling at microarchitecture level," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1042–1053, July 2005.
- [31] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," in *ISCA '95: Proceedings of the 22nd annual International Symposium on Computer Architecture*. New York, NY, USA: ACM Press, 1995, pp. 24–36.
- [32] M.-L. Li, R. Sasanka, S. V. Adve, Y.-K. Chen, and E. Debes, "The alpbench benchmark suite for complex multimedia applications," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC-2005)*. Washington, DC, USA: IEEE Computer Society, 2005.