Optimizing the Thermal Behavior of Subarrayed Data Caches

Johnsy K. John, Jie S. Hu, and Sotirios G. Ziavras
Electrical and Computer Engineering, New Jersey Institute of Technology
Newark, NJ 07102
{jkj2, jie.hu, sotirios.g.ziavras}@njit.edu

Abstract

Designing temperature-aware microarchitectures for microprocessors at new technologies is becoming a critical requirement due to the exponentially increasing on-chip power density. Extremely high power density, thus the very high on-chip temperature, not only significantly increases the packaging and cooling costs, but also creates tremendous difficulties in chip leakage control and reliability.

Being a major contributor to chip transistor budget and die area, caches account for a significant share of the overall processor power consumption, including both dynamic and leakage power. This work analyzes the thermal behavior of subarrays within a conventional data cache when running a set of applications from the SPEC2000 benchmark suite, and proposes two new subarraying schemes, namely, the separated scheme and the interleaved scheme, to improve the thermal behavior of subarrays in terms of more predictable behavior and reduced subarray temperatures. These optimizations can be also combined with dynamic thermal management (DTM) techniques to further improve the efficiency of thermal management. The impact of leakage control on the subarray thermal behavior is also evaluated.

1 Introduction

Continuous technology scaling down leads to an exponential increase in on-chip transistor integration density and an even faster operating frequency. While enjoying the potentially higher performance delivered from new technologies, we are facing new challenges such as increasing design complexity and chip thermal management. With a relative slow supply voltage scaling, the on-chip power density exhibits an exponential increase as technology advances [3]. This high on-chip power density in turn leads to very high chip temperature demanding much larger cooling capacity for the microprocessor designs, thus significantly increasing the costs of cooling systems and chip packaging [19].

Dynamic thermal management (DTM) [4][18] monitors chip-wide temperature at runtime and dynamically invokes power reduction schemes (e.g., dynamic voltage/frequency scaling, clock gating, speculation control [15]) to avoid thermal emergency when the temperature exceeds a pre-defined warning threshold (DTM trigger temperature). Therefore, the temperature is controlled before reaching the one designed for maximum cooling capacity. This also implies that lowering the maximum cooling capacity, thus significantly reducing the cooling cost, can be achieved by appropriately setting the DTM trigger temperatures. Skadron et. al. [18] also showed the strong connection between the effectiveness of DTM techniques and the accuracy of the on-chip temperature sensors.

It is our belief that a thermal-aware microarchitectural design for major components such as caches, register file, instruction issue queues, and functional units can further improve the effectiveness of DTM techniques in controlling thermal emergencies. Understanding the thermal behavior of these major components is essential towards such a thermal-aware microarchitecture design. In modern wide-issue superscalar microprocessors, multiported data cache is required to support multiple cache accesses per cycle since most load instructions are on the critical path, leading to high temperature in the data cache [18]. Due to the exponential effect of the temperature on subthreshold leakage, controlling the temperature in the data cache is of paramount importance in reducing data cache leakage.

In this paper, we study the thermal behavior of subarrays within the data cache, the technology impact of different subarray schemes, and the implications for more efficient power/thermal management in the data cache. Our experimental results using a set of SPEC2000 benchmarks show that 1) different subarrays in a conventional simple subarrayed data cache have very different thermal behavior during the course of simulation, 2) subarray temperature increases dramatically with this simple subarray scheme, at deep submicron technologies, which might lead to possible thermal emergency in these subarrays. By separating data subarrays that will be accessed simultaneously during a cache read/write, we propose an separated subarray scheme (-sp) that achieves more evenly distributed temperature among data subarrays and closer correlation with the initial temperature of each subarray, and also produces a more predictable thermal behavior. We further propose an interleaved scheme (-il) that employs cache way distribution among subarrays and the subblock predecoding to limit the cache access only to a particular subarray, thus reducing the dynamic power consumption and subarray temperatures. Our experimental results at different technologies (130nm and 70nm) indicate an even more severe thermal problem at future technologies, which requires a joint effort from both microarchitectural optimizations for major processor components and dynamic schemes for power (dynamic and leakage) optimizations to improve the efficiency of thermal management.

The rest of this paper is organized as follows. In Section 2, we discuss related work in microprocessor power and thermal optimizations. Section 3 introduces three subarray

schemes and their detailed design for the data cache. We present our experimental setup in Section 4. In Section 5, we evaluate the thermal behavior of the subarray schemes and the technology impact. Section 6 concludes this work.

2 Related Work

Thermal aware microarchitectures and dynamic thermal management techniques are closely related to power management techniques. Once thermal management is triggered, some typical power control mechanisms are invoked to reduce the power consumption thus to lower the temperature in processor components. PowerPC G3/G4 microprocessors dynamically monitor the junction temperature of the processor through an on-chip thermal sensor and dynamically invoke power management, such as instruction cache throttling, when temperature reaches a threshold value [17]. Pentium 4 employs global clock gating for its thermal management [8]. Brooks and Martonosi [4] investigated the effectiveness of several typical power control techniques, such as clock frequency scaling, voltage and frequency scaling, decode throttling, speculation control, and instruction cache throttling, as dynamic response mechanisms in thermal management. Many previous works use power numbers to predict the temperature. In [18], Skadron et. al. developed a microarchitecture level thermal model, HotSpot, for architectural studies of thermal management techniques.

Due to the caches' large share in processor power consumption, optimizing cache power is critical for processor power management and thus has been the focus of many research efforts. Stage-skip pipeline [10] introduces a small decoded instruction buffer (DIB) to temporarily store decoded loop instructions that are reused to skip instruction fetching and decoding for power reduction. Loop caches [13][1] dynamically detect loop structures and buffer loop instructions or decoded loop instructions in an additional loop cache for later reuse. More generally, filter caches [12] use smaller level zero caches (between the level one cache and datapath) to capture tight spatial/temporal locality in cache access thus reducing the power consumption in larger level one caches.

As leakage is becoming a dominant part in cache power consumption at deep sub-micron technologies[7], controlling leakage is essential for cache power/thermal optimization. DRI i-cache [20] and cache decay [11] utilize gated-Vdd techniques to dynamically turn off a portion of the cache or a cache line for leakage reduction. Drowsy cache [7] utilizes a multiplexed supply voltage for the cache lines and periodically transitions all cache lines into a drowsy mode. A compiler-directed leakage management scheme [21] inserts special leakage control instructions based on compiler code analysis. At the circuit level, the asymmetric SRAM cell [2] achieves much lower leakage while storing a value of zero. Bitline leakage reduction by leaving bitlines open was proposed in [9].

Different from the above work, our focus here is to analyze and gain understandings of the thermal behavior in the data cache and to design cache subarray schemes in a thermal-aware way. Such a thermal-aware subarray scheme is not only thermal efficient by design, but is also able to support other dynamic thermal/power optimization schemes. Further, we study the leakage impact on subarray temperatures and the critical importance of leakage control for cache thermal management at deep sub-micron technologies.

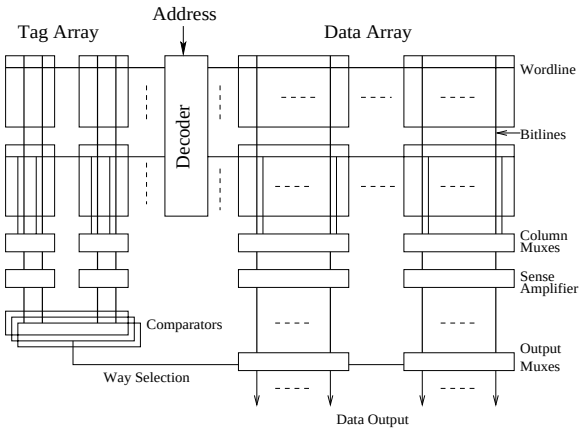


Figure 1. A general view of cache organization.

3 Thermal-Aware Cache Subarraying

3.1 Basics of Cache Organizations

A general view of the cache organization is shown in Figure 1. The data array (on the right) and the tag array (on the left) form the two major parts in the cache. To achieve better cache access and cycle time, both the data and tag arrays may further be divided into subarrays to reduce the wordline or bitline delay. In the Cacti model [16], parameters $Ndwl$ and $Ndbl$ define how the data array is horizontally and vertically divided into subarrays. Parameter $Nspd$ defines how many sets are mapped to a single wordline. Thus, these three parameters define the organization of the data array. The optimal values of $Ndwl$, $Ndbl$, and $Nspd$ are determined by the cache size, cache block size, and the set associativity. Similarly, the tag array is also configured by similar parameters, $Ntwl$, $Ntbl$, and $Ntspd$.

3.2 A Simple Subarrayed Cache

Given a cache configuration and a particular process technology, the optimal values of parameters $Ndwl$, $Ndbl$, and $Nspd$ can be determined with respect to an optimizing function (e.g., for best access and cycle time) [16]. With these three parameters determined, a simple and fast subarraying scheme is to enable the decoders of all subarrays on the same horizontal row with a predecoding signal. Then, the column multiplexer will select the bitlines from one among the $Ndbl$ logical subarrays.

Figure 2 shows the data array implementing such a simple subarraying scheme. The bitlines of the data array are vertically divided into $Ndbl$ segments and the wordlines are horizontally divided into $Ndwl$ segments. Each subarray has $Nspd$ sets mapped to each wordline. Thus, the number of vertical segments is $Ndbl/Nspd$. During a cache access, the predecoding lines select/enable one row of subarrays among the $Ndbl/Nspd$ rows, followed by the selected subarray decoders decoding the remaining index bits in the address and accessing all the subarrays on this row. The performance

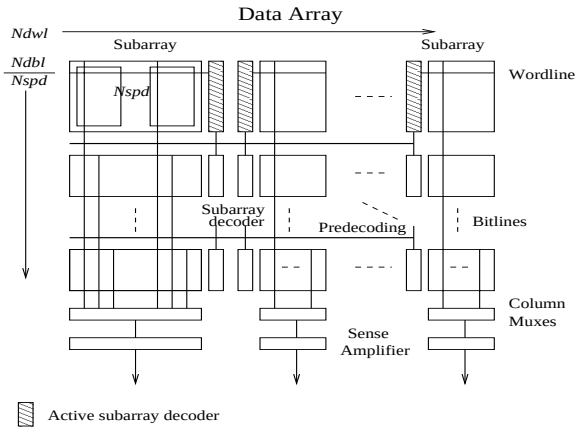


Figure 2. A simple cache subarraying scheme (-org). Each subarray has $Nspd$ sets mapped to a single wordline.

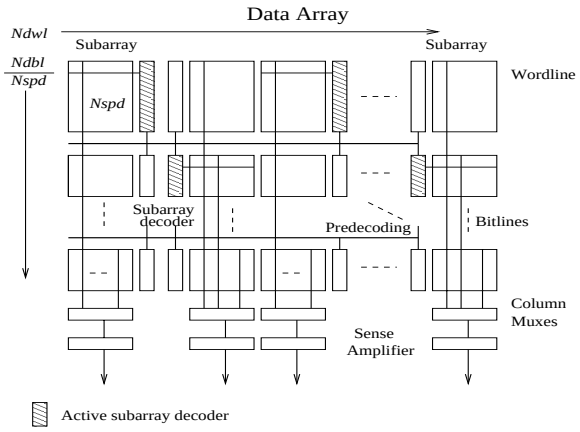


Figure 3. A separated cache subarraying scheme (-sp).

overhead of predecoding in this scheme should be minimum. However, from the thermal perspective, such a subarraying scheme is not efficient since the subarrays on a row are always accessed simultaneously and consume dynamic power. These subarrays have at most two idle subarrays (above and below them, see Figure 2) to spread heat to, which may lead to heat buildup on the accessed row due to hot neighboring subarrays. High temperature in these subarrays will cause significantly larger leakage than other low temperature subarrays and increase the overall cache leakage consumption. The increased leakage power may further cause temperature rise and possibly lead to thermal runaway. One possible solution to alleviate this thermal problem is to increase the number of idle subarrays with low temperature around an accessed subarray for fast heat diffusion.

3.3 Separating Subarrays

An alternative subarraying scheme, as a solution to the thermal problem for the simple scheme in the previous subsection, is to deneighbor or separate subarrays (on the same row) that will be accessed in parallel. How far a neighboring subarray can be put away depends on the two parameters $Ndbl$ and $Nspd$. To simplify the data output routing, we limit the subarray placement only within its column. Therefore, $Ndbl/Nspd$ defines the number of locations that a subarray can be placed into. The farther the two accessed subarrays are separated, the more efficient the heat dissipation should be. Figure 3 presents such a separate-subarrayed cache.

The separated subarraying scheme in Figure 3 puts the subarrays on the logical row into two neighboring physical rows in such a way that any two subarrays are not direct (horizontally or vertically) neighbors. This scheme also makes the accessed subarray have up to four possibly idle subarrays (up, bottom, left, and right) surrounding it, which helps quick temperature reduction for the hot subarray. However, the effectiveness of this separated subarraying scheme is limited in the presence of multiple cache accesses, where multiple rows of subarrays may be accessed in parallel.

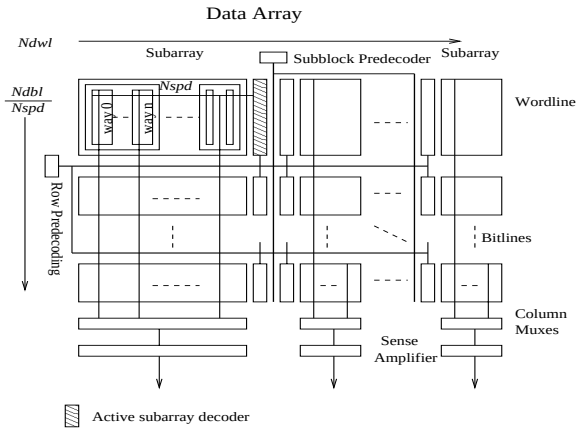


Figure 4. A way-interleaved cache subarraying scheme (-il).

3.4 Distributing Cache Ways among Subarrays

In both the simple and separated subarraying schemes, a cache read/write accesses all the subarrays on a selected logical row in the array. Notice that the data width between the CPU and data cache is usually a word (four or eight bytes) and is much smaller than the cache block size (e.g., 32 or 64 bytes). In the last stage of cache access (assuming a cache hit), the cache way selection from the tag comparison and block offset bits in the address together control the output multiplexers to route out the right data to the CPU [16]. We propose limiting a cache access only to activate one subarray rather than all subarrays on the same logical row by employing subblock predecoding (i.e., moving partially the word selection from the output multiplexer to an early predecoding stage) and cache way interleaving among subarrays (i.e, distributing a cache way among all subarrays on the row). Due to significantly reduced power consumption for each cache access, the new proposal has the potential to further improve the thermal efficiency of subarrayed caches.

Figure 4 shows such an interleaved subarraying scheme.

Processor Core	
Int/FP Issue Queue	20/20 entries
Load/Store Queue	64 entries
Active List	80 entries
Int/FP Physical Reg. File	80/72 registers
Fetch/Decode/Commit Width	4 instructions per cycle
Int/FP Issue Width	4/2 instructions per cycle
Function Units	4 IALU, 2 IMULT/IDIV, 2 FALU, 1 FMULT/FDIV/FSQRT, 2 Mem Read/Write ports
Branch Predictor	
Branch Predictor	tournament predictor PAg/GAg with GAg chooser
BTB	2048 entries, 2-way
RAS	32-entry
Memory Hierarchy	
L1 ICache	64KB, 2 ways, 64B blocks, 2 cycles
L1 DCache	64KB, 2 ways, 64B blocks, 2 cycles
L2 UCache	4MB, 8 ways, 128B blocks, 12 cycles
Memory	225 cycles first chunk, 12 cycles rest
I/DTLB	128 entrie, full assoc., 30 cycle miss penalty

Table 1. Parameters for the simulated Alpha 21364 microprocessor.

Each subarray might have $Nspd$ sets mapped to the same wordline. Each set within the subarray holds a subblock from each cache line (cache block) of A ways, where A is the set associativity. Assuming B is the cache line size in bytes, the subblock size (each way distributed among subarrays) is $B/Ndwl$. The row predecoder is kept unchanged as in the previous schemes. Row predecoding takes the higher $log_2(Ndbl/Nspd)$ index bits in the address to select a particular row of subarrays and their decoder for accepting the remaining index bits. In the mean time, the subblock predecoder uses the higher $log_2(Ndwl)$ block offset bits in the address to enable the subarray decoders on a particular column. These two predecoders together locate a single subarray and enable its subarray decoder for the current cache access. However, a cacheline replacement at cache misses needs to access all the subarrays on that row.

4 Experimental Setup

In this section, we discuss the experimental setup and benchmark selection for our work. We extended the original SimpleScalar simulator [6] with some radical modifications to model the Alpha 21364 microprocessor as close as possible. The detailed configuration of the simulated Alpha 21364 microprocessor is given in Table 1.

The power model for this Alpha 21364 microprocessor simulator is derived from Wattch [5]. HotSpot [18] and HotLeakage [22] are also incorporated into the processor simulator to profile the transient temperatures and the leakage power of the data cache. The Alpha EV7 floorplan (with its EV6 core) from [18] is used as the reference floorplan for this study. For the two technologies, 130nm and 70nm, we are evaluating here, the floorplan and the dimensions of the processor components are scaled accordingly from the 180nm technology. Other HotSpot related operating parameters are similar to those used in [18] and some are listed in Table 2.

We used a set of 10 integer benchmarks and 8 floating-point benchmarks from the SPEC2000 benchmark suite in this study. All benchmarks were compiled for the Alpha instruction set architecture with "peak" tuning. Each benchmark is first fastforwarded 1 billion instructions, then simulates the next 0.5 billion instructions in details.

HotSpot Parameters		
Technology	130nm	70nm
Clock Frequency	3GHz	5.6GHz
Supply Voltage	1.5V	1.0V
Ambient Air Temperature	45°C	
Package Thermal Resistance	0.8K/W	
Die	0.5mm thick, 15.9mm x 15.9mm	
Heat Spreader	Copper, 1mm thick, 3cm x 3cm	
Heat Sink	Copper, 7mm thick, 6cm x 6cm	

Table 2. HotSpot related parameters.

5 Experimental Results

In this section, we evaluate the effectiveness of proposed subarray schemes in optimizing the thermal behavior of cache subarrays at two different technologies, 130nm and 70nm. The simulated data cache has the same configuration as the one in Alpha 21364 microprocessors, which is given in Table 1. The optimal subarray parameters derived from Cacti 3.2 [16] are as follows, $Ndwl$ = 4, $Ndbl$ = 2, $Nspd$ = 1, $Ntwl$ = 1, $Ntbl$ = 4, and $Ntspd$ = 2. Since the tag array only accounts for around 3% of the total area, our evaluation of the subarray schemes focuses on the data array portion. Figure 5 shows the subarray layouts of the data cache for the three different schemes discussed in Section 3.

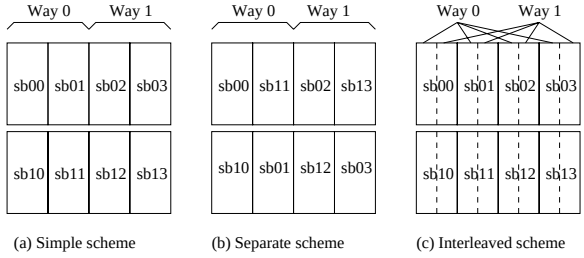
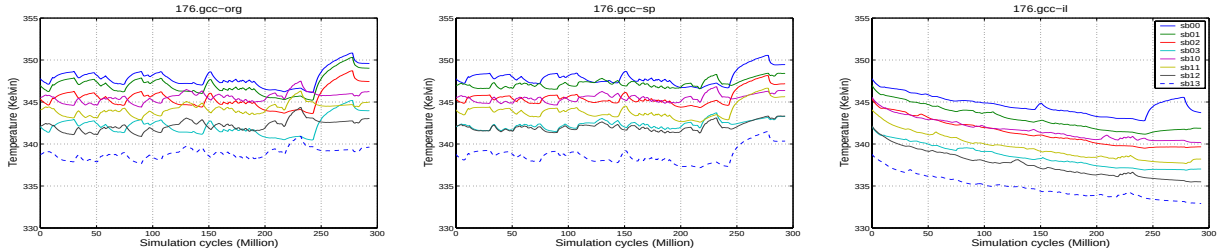


Figure 5. Subarray layouts of the data cache for three subarraying schemes.

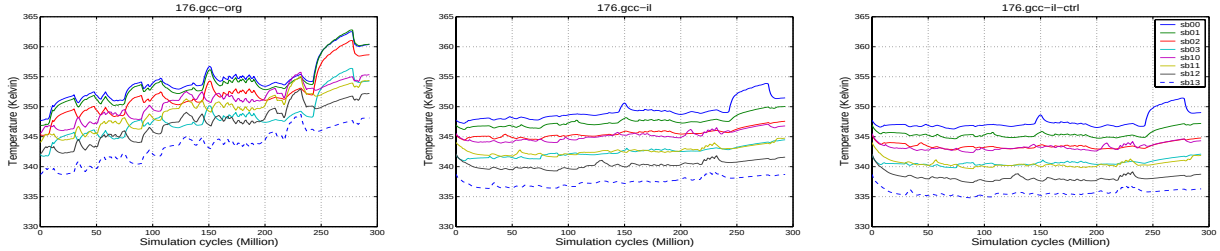
We use HotSpot to update transient temperatures every 100K cycles. The initial temperatures of major processor components and subarrays in the data cache for this limited study are derived from the steady temperatures after one sample simulation.

We first analyze the thermal behavior of the data cache with three different subarraying schemes: simple (-org) scheme, separated (-sp) scheme, and interleaved (-il) scheme, at the 130nm technology. Figure 6 shows the comparisons for a typical integer benchmark (*gcc*). The transient temperatures of each of the eight subarrays during the simulation are given for all three schemes. It is noticeable from Figure 6 (a) that the integer benchmark *gcc* has very intensive variations in subarray temperatures during the simulation. In some benchmarks, such as *vpr* and *lucas* (results not shown), the subarray temperatures drop due to the lower average data



(a) Simple scheme (-org) (b) Separated scheme (-sp) (c) Interleaved scheme (-il)

Figure 6. Thermal behavior of subarrays with different schemes for Integer benchmarks (at 130nm technology).



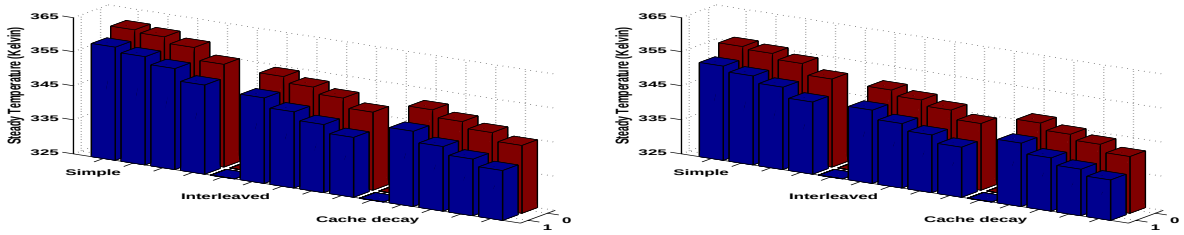
(a) Simple scheme (-org) (b) Interleaved scheme (-il) (c) (-il) scheme with cache decay

Figure 7. Thermal behavior of data cache subarrays at 70nm technology.

cache accesses per cycle. By separating subarrays that will be accessed simultaneously into two different rows (as shown in Figure 5 (b)), the separated scheme (-sp) (Figure 6 (b)) sees more evenly distributed subarray temperatures and decreased subarray temperature variations. This is due to the evenly distributed spatial accesses to these cache subarrays. However, the separated scheme is not effective in lowering the subarray temperatures. This could be explained by the overall still high dynamic power consumption in the data cache. With interleaved subarrays, as shown in Figure 6 (c), we achieve very nice thermal behavior in the data cache with all the subarray temperatures dropping continuously. As the interleaved scheme employs both subarray row predecoding and subblock predecoding, each cache access only activates one subarray which dramatically reduces the dynamic power consumption and improves heat diffusion to the surrounding idle subarrays. Steady temperature results (not presented here) show that the interleaved scheme achieves substantially reduced subarray temperatures. On the average, the interleaved scheme reduces the subarray temperatures by 6.5 degrees for integer benchmarks and 5.1 degrees for floating-point benchmarks, compared to the simple scheme. Since the separated scheme has less impact on subarray temperature reduction, it is not considered in our analysis at the 70nm technology.

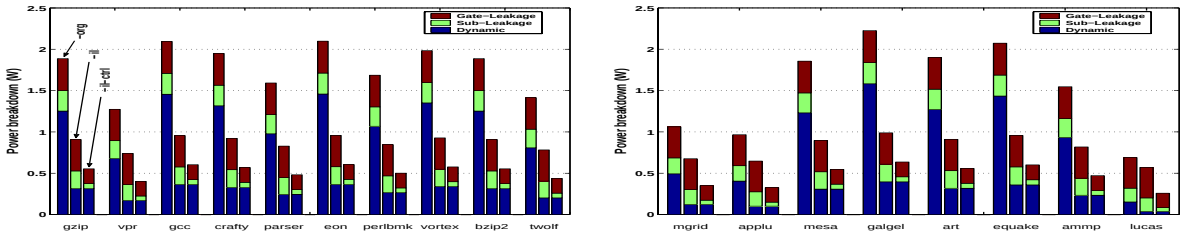
For the the 70nm technology, the increasing power density creates much higher temperature in subarrays, as shown in Figure 7. With the simple subarraying scheme (-org), all subarray temperatures are sharply increasing (Figure 7 (a)). Although the interleaved subarray scheme does not deliver the same thermal behavior as at the 130nm technology, it effectively slows down temperature rising in subarrays (Figure 7 (b)). One major reason of this thermal behavior is the significantly increased cache leakage power at sub-micron technologies. From our results, the leakage power (including both subthreshold and gate leakage calculated by HotLeakage [22] with a temperature feedback from HotSpot [18] every 100K cycles) in the cache with the interleaved subarraying scheme is around 70% of the overall cache power at 70nm technology, as shown in Figure 9. The large gate leakage is partially due to the 1.0V supply voltage used in this evaluation. Since cache decay is more sensitive to temperature variations than the drowsy cache scheme [14], we exploit the cache decay scheme for leakage control in the context of the data cache thermal optimization. The decay interval is 8K cycles as suggested in [11]. Figure 7 (c) shows the impact of leakage optimizations on the dynamic thermal behavior of cache subarrays. At 70nm technology, the interleaved scheme reduces the subarray temperatures by 8.2 degrees and 6.6 degrees on the average for the integer and floating-point benchmarks, respectively. With cache decay applied, the temperature reduction is increased to 11.4 degrees and 9.7 degrees for the integer and floating-point benchmarks, respectively, as shown in Figure 8.

To show how the subarraying schemes optimize the cache power and consequently the thermal behavior, we break down data cache power consumption (at the 70nm technology) into three part: dynamic power, subthreshold leakage, and gate leakage, for both integer and floating-point benchmarks, as shown in Figure 9. The interleaved subarray scheme achieves a significant dynamic power reduction, 75% on the average, which leads to substantially reduced subarray temperature and optimized cache thermal behavior. Applying cache decay, the subthreshold leakage (gate leakage) is dramatically reduced, by 75% (53%) on the average for all benchmarks, which further improves the subarray temperature reduction. Notice that this subthreshold leakage reduction is due to the joint effects of the cache decay scheme and the reduced subarray temperature.



(a) Average for integer benchmarks (b) Average for floating-point benchmarks

Figure 8. Steady subarray temperature comparison at 70nm technology. "0" and "1" correspond to the top and bottom rows in Figure 5.



(a) Integer benchmarks (b) Floating-point benchmarks

Figure 9. Data cache power breakdown at the 70nm technology for the simple scheme (-org), the interleaved scheme (-il), and the interleaved scheme with cache decay applied (-il-ctrl).

6 Conclusions

In this work, we focus on designing thermal-efficient data caches, a major processor component in terms of power consumption and transistor/die area budget. First, we analyzed the thermal behavior of subarrays within a conventional data cache. The simple subarray organization implies possible thermal emergency due to heat buildup involving neighboring subarrays. Then, an optimized subarraying scheme was proposed to avoid this heat accumulation by separating subarrays (to be accessed simultaneously) into different physical rows. However, its effectiveness is limited due to the large power consumption when accessing multiple subarrays in parallel. Finally, we proposed a way-interleaved subarraying scheme utilizing additional subblock predecoding to activate only one subarray during a cache access. Our evaluation results show that the interleaved scheme achieves superior cache thermal behavior and significantly reduces cache temperature. As leakage is becoming a dominant part of cache power consumption at deep sub-micron technologies, leakage control mechanisms must be also employed in order to exploit the thermal efficiency of the interleaved subarraying scheme.

References

[1] T. Anderson and S. Agarwala. Effective hardware-based two-way loop cache for high performance low power processors. In *IEEE ICCD*, 2000.

[2] N. Azizi, A. Moshovos, and F. N. Najm. Low-leakage asymmetric-cell sram. In *Proc. the 2002 ISLPED*, Monterey, CA, 2002.

[3] S. Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4):23–29, 1999.

[4] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proc. 7th HPCA'01*, 2001.

[5] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proc. International Symposium on High-Performance Computer Architecture*, 2000.

[6] D. Burger, A. Kagi, and M. S. Hrishikesh. Memory hierarchy extensions to simplescalar 3.0. Technical Report TR99-25, Department of Computer Sciences, The University of Texas at Austin, 2000.

[7] K. Flautner, N. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: Simple techniques for reducing leakage power. In *Proc. the 29th International Symposium on Computer Architecture*, Anchorage, AK, May 2002.

[8] S. H. Gunther, F. Binns, D. M. Carmean, and J. C. Hall. Managing the impact of increasing microprocessor power consumption. *Intel Technology Journal*, Q1, 2001.

[9] S. Heo, K. Barr, M. Hampton, and K. Asanovi. Dynamic fine-grain leakage reduction using leakage-biased bitlines. In *Proc. the 29th International Symposium on Computer Architecture*, Anchorage, AK, May 2002.

[10] M. Hiraki et al. Stage-skip pipeline: A low power processor architecture using a decoded instruction buffer. In *Proc. ISLPED*, 1996.

[11] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: Exploiting generational behavior to reduce cache leakage power. In *Proc. the Int'l Symposium on Computer Architecture*, 2001.

[12] J. Kin et al. The filter cache: An energy efficient memory structure. In *Proc. International Symposium on Microarchitecture*, December 1997.

[13] L. H. Lee, B. Moyer, and J. Arends. Instruction fetch energy reduction using loop caches for embed ded applications with small tight loops. In *Proc. ISLPED*, 1999.

[14] Y. Li, D. Parikh, Y. Zhang, K. Sankaranarayanan, M. R. Stan, and K. Skadron. State-preserving vs. non-state-preserving leakage control in caches. In *Proc. the 2004 DATE* , pages 22–27, Feb. 2004.

[15] S. Manne, A. Klauser, and D. Grunwald. Pipeline gating: Speculation control for energy reduction. In *Proc. the 25th Annual International Symposium on Computer Arch itecture*, pages 132–141, June 1998.

[16] G. Reinman and N. Jouppi. An integrated cache timing and power model. Technical report, Compaq Western Research Lab, 1999.

[17] H. Sanchez et al. Thermal management system for high performance PowerPC microprocessors. In *Proceedings of COMPCON 97*, San Jose California, 1997.

[18] K. Skadron et al. Temperature-aware microarchitecture. In *ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture*, pages 2–13, San Diego, California, 2003.

[19] V. Tiwari et al. Reducing power in high-performance microprocessors. In *35th Design Automation Conference*, 1998.

[20] S.-H. Yang, M. D. Powell, B. Falsafi, K. Roy, and T. N. Vijaykumar. An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance i-caches. In *Proc. HPCA* , 2001.

[21] W. Zhang, J. S. Hu, V. Degalahal, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. Compiler-directed instruction cache leakage optimization. In *Proc. the 35th Micro* , Istanbul, Turkey,, November 2002.

[22] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. Technical report, Dept. of Computer Science, Univ. of Virginia, 2003.