INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 13, NO. 4, DECEMBER 2008, 251-263

Thermal-Aware Subarrayed Data Cache Microarchitectures

Jie Hu, Johnsy K. John, and Shuai Wang

***Abstract*—Designing thermal-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 cost, but also creates tremendous difficulties in chip leakage control and reliability. As 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 proposes to characterize the thermal behavior within data caches at the fine granularity instead of treating the data cache as a monolithic block. It develops detailed thermal models for a typical subarrayed data cache. Furthermore, this work proposes and evaluates a new subarray scheme, namely way-interleaved scheme, to improve the thermal behavior of subarrays. This optimized cache microarchitecture can be also combined with dynamic thermal management techniques to further improve the efficiency of the thermal management. The interaction between leakage control and thermal optimization is also investigated in the context of thermal-aware microarchitectural designs.**

***Index Terms*—Thermal management, data cache, thermal-aware microarchitecture, power density, interleaved subarray scheme, leakage**

1. INTRODUCTION

Continuous technology scaling down leads to an exponential increase in on-chip transistor integration density and high 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 effort. With a relatively slow supply voltage scaling, the on-chip power density exhibits an exponential increase as technology advances [1]. 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. As most microprocessors are designed for the worst-case operating temperature, though very rare, to prevent the chip from meltdown during the execution of some max-power applications, designing the thermal packaging to accommodate these peak power situations imposes a substantial extra manufacturing cost [2].

A preliminary version of this work was presented at the IEEE International Conference of Computer Design (ICCD-2005), San Jose, California, October 2-5, 2005.

Jie Hu and Shuai Wang are with Department of Electrical and Computer Engineering, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA (E-mail: jhu@njit.edu; sw63@njit.edu).

Johnsy K. John is with AMD Boston Design Center, 90 Central Street, Boxborough, MA 01719, USA (E-mail: johnsy.john@amd.com).

Since microprocessors run at average power consumption/temperature during majority of the time, techniques can be applied to prevent the microprocessor from running at the maximum power consumption (thermal emergency situation) without noticeable performance degradation because of the rare occurrence of thermal emergencies. Dynamic thermal management (DTM) [3][4] monitors chip-wide temperature at runtime and dynamically invokes power reduction schemes (e.g., dynamic voltage/frequency scaling, clock gating, speculation control [5]) 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. [4] also showed the strong connection between the effectiveness of DTM techniques and the accuracy of the on-chip temperature sensors.

DTM is critical for high performance microprocessors designed at new technologies. However, frequent DTM invocation at high temperatures has significant performance impact [3][4]. 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 handling thermal emergencies. Understanding the thermal behavior of these major components is essential towards such a thermal-aware microarchitectural design. Being major contributors to the on-chip transistor budget and die area, caches consume a large portion of the overall processor power dissipation, including both dynamic power and leakage power consumption [6][7]. 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 [4]. Due to the exponential effect of the temperature on subthreshold leakage, controlling the temperature in caches is of paramount importance in reducing cache leakage and avoiding thermal runaway emergency.

In this paper, we first study the thermal behavior of the data cache at the granularity of subarrays. Due to different cache configurations and access patterns/localities, different parts of the data cache experience quite different activities and thermal behavior [8], which however cannot be captured by conventional thermal studies [4]. With a detailed simulator modeled for the Alpha 21364 microprocessor, 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, and 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, an improved subarray scheme proposed in this paper shows 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. A further optimization of our cache subarray scheme 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.

To enable more detailed and comprehensive thermal analysis, we further propose to characterize the thermal behavior of the data cache at fine granularities instead of treating the data cache as a monolithic block. The proposed fine-grain characterization will expose to computer architects the detailed internal thermal behavior of the data cache and open new ways to more effective fine-grain thermal control and leakage management techniques. A fine-grain thermal model that captures the great details of the physical implementations of the data cache is developed for the proposed thermal characterization in this work. Our experimental results show that the temperature variation among different parts of the data cache is more than 10 degrees in Celsius. Another interesting finding is that even the power density does not have a strong correlation with the temperature at this fine granularity. For instance, the output drive circuits and post decoders of data subarrays possess the highest power density, however are not the thermal hotspots within the data cache. 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 optimization to improve the efficiency of thermal management.

The rest of this paper is organized as follows. In Section 2, we discuss the related work in microprocessor power and thermal optimization. The experimental setup for this work is presented in Section 3. We introduce the three subarray schemes and their detailed design for the data cache in Section 4. Subarray level thermal modeling and evaluation is presented in Section 5 and fine-grain thermal modeling and characterization is detailed in Section 6. Section 7 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 [9]. Pentium 4 employs global clock gating for its thermal management [10]. Lim et. al. [11] proposed a thermal-aware microprocessor consisting of a regular out-of-order superscalar pipeline and a secondary simple ultra low power in-order pipeline. The execution mode is switched from out-of-order to in-order in case that low power is required for mobile applications or the chip temperature exceeds a predefined threshold. Brooks and Martonosi [3] 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 studies use power numbers to predict the temperature. In [4][12], Skadron et. al. developed a microarchitecture level thermal model, HotSpot, for architectural studies of thermal management techniques. There is some previous work [13][14] on thermal management in SMP systems, which schedules the tasks to make use of the idle SMP nodes. Srinivasan et al. proposed the predictive dynamic thermal management by profiling multimedia applications [15]. All the studies target at the overall chip temperature or the localized temperature whose granularity is a functional unit, e.g., integer register file, integer execution unit, issue unit and L1 instruction cache.

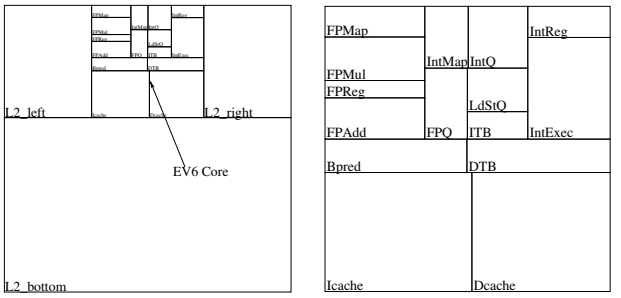
Due to the caches' large share in processor power consumption, managing cache power is critical for processor power management and thus has been the focus of many research efforts. Stage-skip pipeline [16] introduces a small decoded instruction buffer (DIB) to temporarily store decoded loop instructions that are reused to skip instruction fetching and decoding for power reduction. DIB is controlled by a special loop-evoking instruction and requires ISA modification. Loop caches [17][18] dynamically detect loop structures and buffer loop instructions or decoded loop instructions in an additional loop cache for later reuse. More generally, filter caches [19] 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 [20], controlling leakage is essential for cache power/thermal optimization. DRI i-cache [21] and cache decay [22] utilize gated-Vdd techniques to dynamically turn off a portion of the cache or a cache line for leakage reduction. Drowsy cache [20] utilizes a multiplexed supply voltage for the cache lines and periodically transitions all cache lines into a drowsy mode, where the supply voltage is reduced for significant leakage savings while maintaining the cache contents. A compiler-directed leakage management scheme [23] inserts special leakage control instructions based on compiler code analysis. At the circuit level, the asymmetric SRAM cell [24] achieves much lower leakage while storing a value of zero. Bitline leakage reduction by leaving bitlines open is proposed in [25].

Different from the above work, our focus here is to analyze

TABLE I
PARAMETERS FOR THE SIMULATED ALPHA 21364 MICROPROCESSOR.

Processor Core	
Int Issue Queue	20entries
FP Issue Queue	15 entries
Load/Store Queue (LSQ)	64 entries
Active List	80 entries
Int Physical Register File	80 registers
FP Physical Register File	72 registers
Fetch/Decode/Commit Width	4 instructions per cycle
Issue Width	6 instructions per cycle (4 Int + 2 FP)
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 entries, full assoc., 30 cycle miss penalty



(a) Alpha EV7 floorplan (b) Alpha EV6 core

Fig. 1. The floorplan of Alpha EV7 (with the EV6 core) [4] used in this work.

TABLE II
HOTSPOT RELATED PARAMETERS.

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	

and gain the understanding of thermal behavior in a data cache and to design cache subarray schemes in a thermal-aware way, which is different from the very fine-grain/cacheline-level power density minimization approach [26]. 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. Furthermore, we study the leakage impact on subarray temperatures and the critical importance of leakage control for cache thermal management at deep sub-micron technologies.

3. EXPERIMENTAL SETUP

In this section, we discuss the experimental setup and benchmark selection for our work. We extend the original SimpleScalar simulator [27] with some radical modifications to model the Alpha 21364 microprocessor as close as possible. In our new simulator, the unified RUU structure is replaced by separate integer issue and floating-point issue queues, integer and floating-point physical register files, and the active list (functioning as the reorder buffer). The register renaming scheme is implemented as the one in MIPS R10000 [28]. There is no separate architectural/logical register file. Committing the current instruction frees the physical register that is being renamed to by the immediately previous instruction with the same destination/result logical register as the current instruction. The new simulator also implements the tournament branch predictor (the local predictor uses 2-bit counters) used in Alpha 21364 microprocessors. The detailed configuration of the simulated Alpha 21364 microprocessor is given in Table I.

The power model for this Alpha 21364 microprocessor simulator is derived from Wattch [29]. HotSpot [4] and HotLeakage [30] 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 [4] is used as the reference floorplan for this study, as shown in Figure 1. 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 [4] and some are listed in Table II.

We use a set of 10 integer benchmarks and 8 floating-point benchmarks from the SPEC2000 benchmark suite in this study. All benchmarks are 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. The selected SPEC2000 benchmarks and their reference inputs used for simulation are given in Table III.

4. THERMAL-AWARE CACHE SUBARRAYING

As a major chip transistor and power consumer, on-chip caches have a significant impact on the processor power efficiency and its overall thermal behavior. More importantly, the thermal behavior of the cache itself has a huge impact on cache leakage (subthreshold leakage) and soft error introduced reliability issues. In this section, we first review some basics of cache organizations. Then, we assume a primary simple subarray scheme and analyze possible thermal issues in the subarrayed cache. Based on this analysis, we propose an alternative subarray scheme that separates the subarrays to be accessed simultaneously during a cache transaction to avoid heat buildup from hot neighboring subarrays. Furthermore, we propose to distribute cache ways among subarrays and use subblock predecoding to limit a cache access to only one subarray. This subarray scheme significantly reduces the power consumption leading to optimized thermal behavior in the cache.

TABLE III
SPEC2000 BENCHMARKS AND REFERENCE INPUT SETS USED IN THIS WORK.

Benchmark	Ref Input
CINT2000 - Integer Benchmarks	
164.gzip	input.source 60
175.vpr	net.in
176.gcc	scilab.i
186.crafty	crafty.in
197.parser	2.1.dict -batch ref.in
252.eon	chair.*
253.perlbmk	splitmail.pl
255.vortex	lendian1.raw
256.bzip2	input.source
300.twolf	ref
CFP2000 - Floating Point Benchmarks	
172.mgrid	mgrid.in
173.applu	applu.in
177.mesa	mesa.in
178.galgel	galgel.in
179.art	c756hel.in,a10.img,hc.img
183.equake	inp.in
188.ammp	ammp.in
189.lucas	lucas2.in

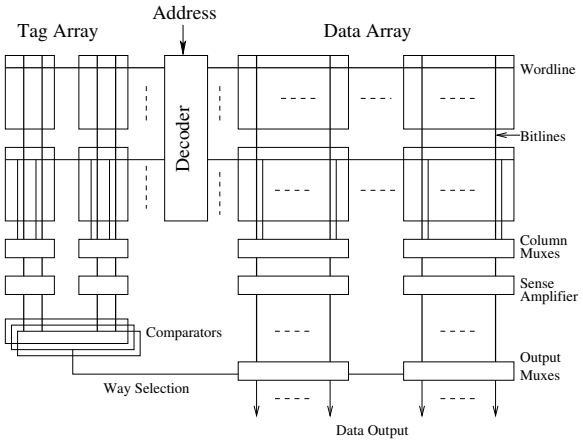


Fig. 2. A general view of cache organization.

4.1. Basics of Cache Organizations

A general view of the cache organization is shown in Figure 2. 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 [31], 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 cache size, cache block size, and set associativity. Similarly, the tag array is also configured by similar parameters, $Ntwl$, $Ntbl$, and $Ntspd$.

A cache access starts with the decoder decoding the index portion of the address. Only one wordline from the decoder is selected. The values stored in the SRAM cells along this selected wordline are read out through the two bitlines associated with each cell. Column multiplexers may be inserted to reduce the number of sense amplifiers needed. The number of selection lines for these column multiplexers is determined by the number of subarrays along the bitline (e.g., $Ndbl$ for the data array) and the number of sets mapped to a single wordline (e.g., $Nspd$ for the data array) [31].

After sense-amplifying the bitline signals, tag comparison is performed between tags read out from the tag array and the tag bits in the given address. A tag match signals a cache hit and selects the data from the hit way. Otherwise, a cache miss is signaled. Since the majority of the cache area and power consumption is contributed by the data array, we focus on the cache data array in the following discussion.

4.2. 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) [31]. 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 selects the bitlines from one among $Ndbl$ logical subarrays.

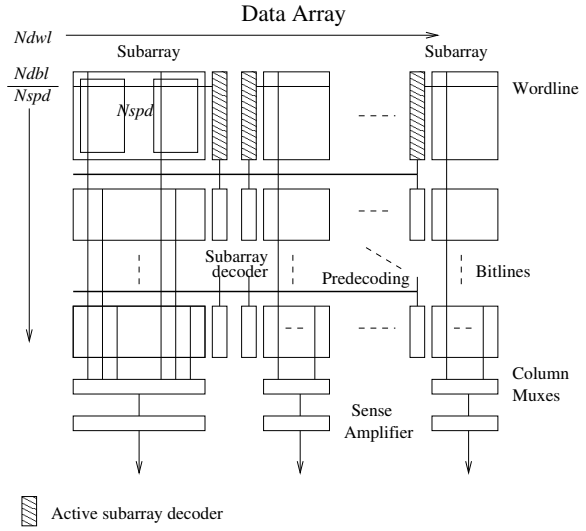


Fig. 3. A simple cache subarraying scheme (-org). Each subarray has $Nspd$ sets mapped to a single wordline.

Figure 3 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 $Ndbl/Nspd$ 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 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, as shown in Figure 3) 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 to 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.

4.3. 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 4 presents such a separate-subarrayed cache.

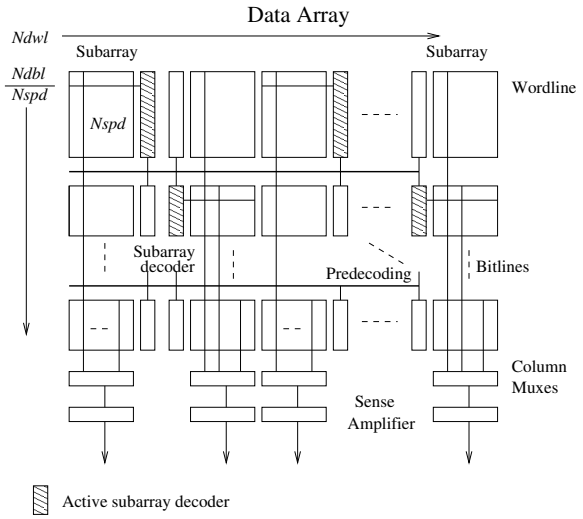


Fig. 4. A separated cache subarraying scheme (-sp).

The separated subarraying scheme in Figure 4 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.

The best thermal efficiency of this separated subarraying scheme is achieved when all the surrounding subarrays are idle during the access of the surrounded subarrays and also have lower temperature than the surrounded ones. However, this ideal situation may not be always attainable in the real world. For instance, multiple accesses (in a single clock cycle) to the data cache go to the first two rows of the subarrays in Figure 4, then all the subarrays on the first two rows are consuming dynamic power and are tightly neighbored. In this case, the separated subarraying has no benefit over the simple scheme. How often this happens again depends on the temporal and spatial locality of the cache access that is determined by the application behavior.

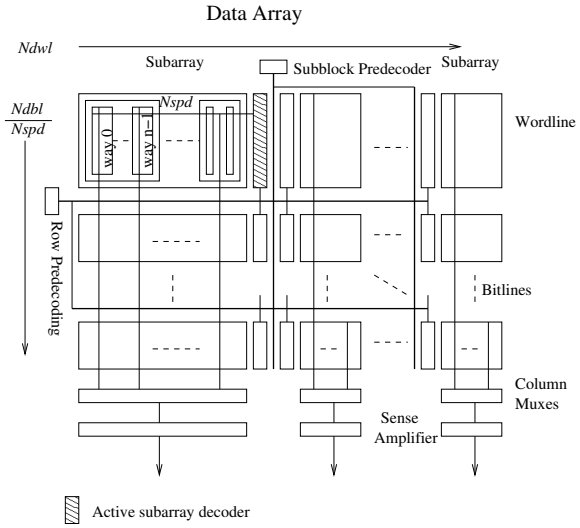


Fig. 5. A way-interleaved cache subarraying scheme (-il).

4.4. 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. The separate-subarrayed cache tries to move these accessed and hot subarrays away from each other for efficient heat diffusion to low temperature subarrays. 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. 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 [31]. 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 5 shows such an interleaved subarraying scheme. 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 that 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.

With moderate cache block sizes and set associativities, the subblock size can be maintained larger than the size of a word (e.g., $B/Ndwl$ = 64 / 4 = 16 bytes). In other words, this proposed way-interleaved subarraying scheme causes no cache bandwidth problem for common cache configurations in most microprocessors. Although the set associativity A is not in the expression calculating the subblock size, the optimal value of $Ndwl$ depends on A.

In the following sections, we evaluate the thermal efficiencies of these three subarraying schemes for a data cache similar to the one in Alpha 21364 microprocessors as well as technology impacts on these schemes.

5. Subarray-Level Thermal Modeling and Evaluation

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, as given in Table I. The optimal subarray parameters derived from Cacti 3.2 [31] are as follows, $Ndwl$ = 4, $Ndbl$ = 2, $Nspd$ = 1. 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 6 shows the subarray layouts of the data cache for the three different schemes discussed in Section 4.

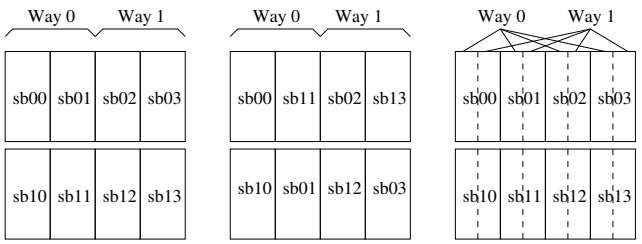


Fig. 6. Subarray layouts of the data cache for three subarraying schemes.

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, which are given in Table IV.

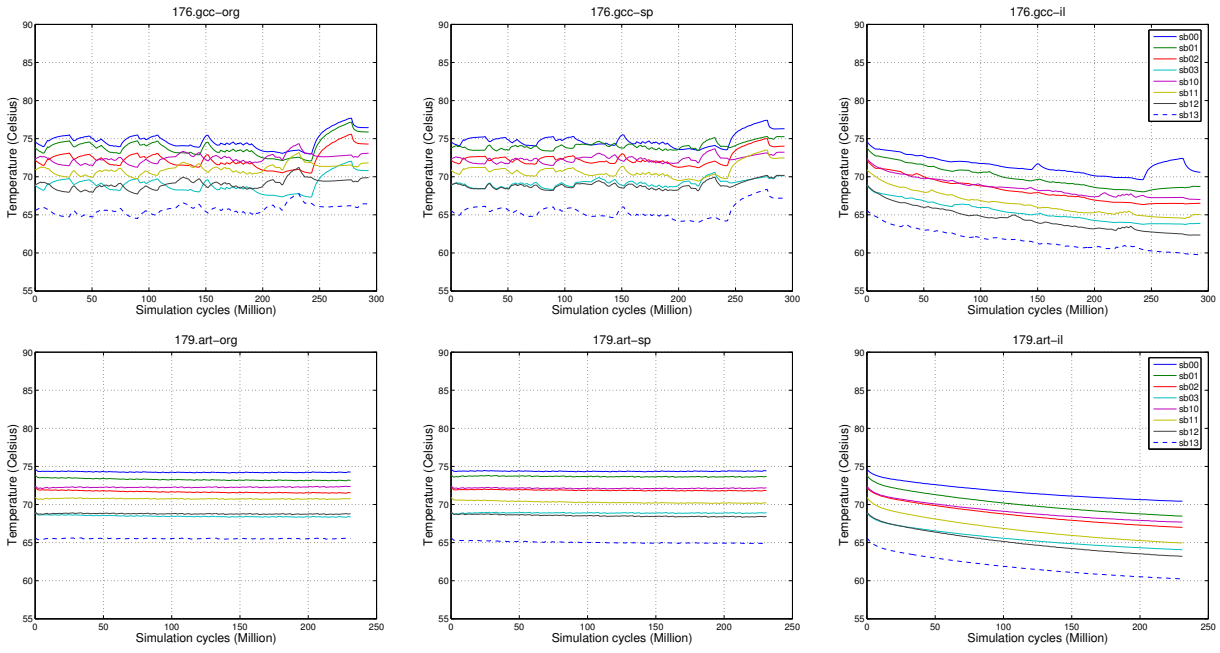
We first analyze the thermal behavior of the data cache with three different subarraying schemes: simple (-org), separated (-sp), and interleaved (-il), at the 130nm technology. Figure 7 shows the comparisons for a typical integer benchmark (*gcc*) and a typical floating-point one (*art*). The transient temperatures of each of the eight subarrays during the simulation are given for all three schemes.

TABLE IV
Initial temperature of major components in the simulated microprocessor.

Component	Temperature (K)	Component	Temperature (K)
L2_left	322.2	Dcache00	347.8
L2_bottom	318.0	Dcache01	347.0
L2_right	324.9	Dcache02	345.4
Icache	348.6	Dcache03	342.1
ITLB	352.1	Dcache10	345.6
DTLB	347.0	Dcache11	344.1
Bpred	347.0	Dcache12	342.2
LdStQ	360.4	Dcache13	338.8
FPQ	347.4	IntQ	367.2
FPReg	341.8	IntReg	383.9
FPMap	337.4	IntMap	351.1
FPAdd	344.9	IntExec	362.9
FPMul	339.6		

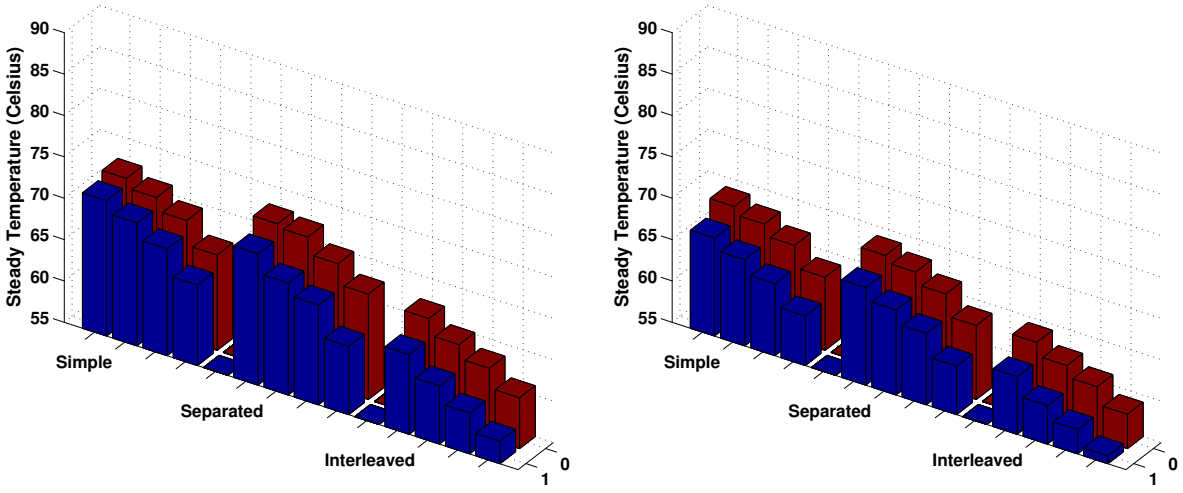
It is noticeable from Figure 7(a) that the integer benchmark *gcc* has very intensive variations in subarray temperatures during the simulation while the floating-point benchmark *art* intends to have very stable temperatures for all subarrays. This difference is mainly due to the very different data cache behavior of integer and floating-point benchmarks. In some benchmarks, such as *vpr* and *lucas* (results not shown), the subarray temperatures drop due to the lower average data cache accesses per cycle. By separating subarrays that will be accessed simultaneously into two different rows (as shown in Figure 6(b)), the separated scheme (-sp) (Figure 7(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. As a result, the subarray temperatures show a strong correlation with the initial temperatures and the thermal behavior is more predictable. 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 7(c), we achieve very nice thermal behavior in the data cache with all the subarray temperatures dropping continuously. This implies a less likely thermal emergency in the data cache. 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.

The steady subarray temperatures, an average for integer and floating-point benchmarks, respectively, are given in Figure 8 with the same floorplans as in Figure 6. Steady temperature results show that the interleaved scheme achieves substantially reduced subarray temperatures. The temperature reduction is even prominent for benchmarks with intensive data cache accesses. 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. Notice that, for all three schemes subarrays on the first row have much higher temperature than those



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

Fig. 7. Thermal behavior of subarrays with different schemes for Integer benchmarks (at 130nm technology).

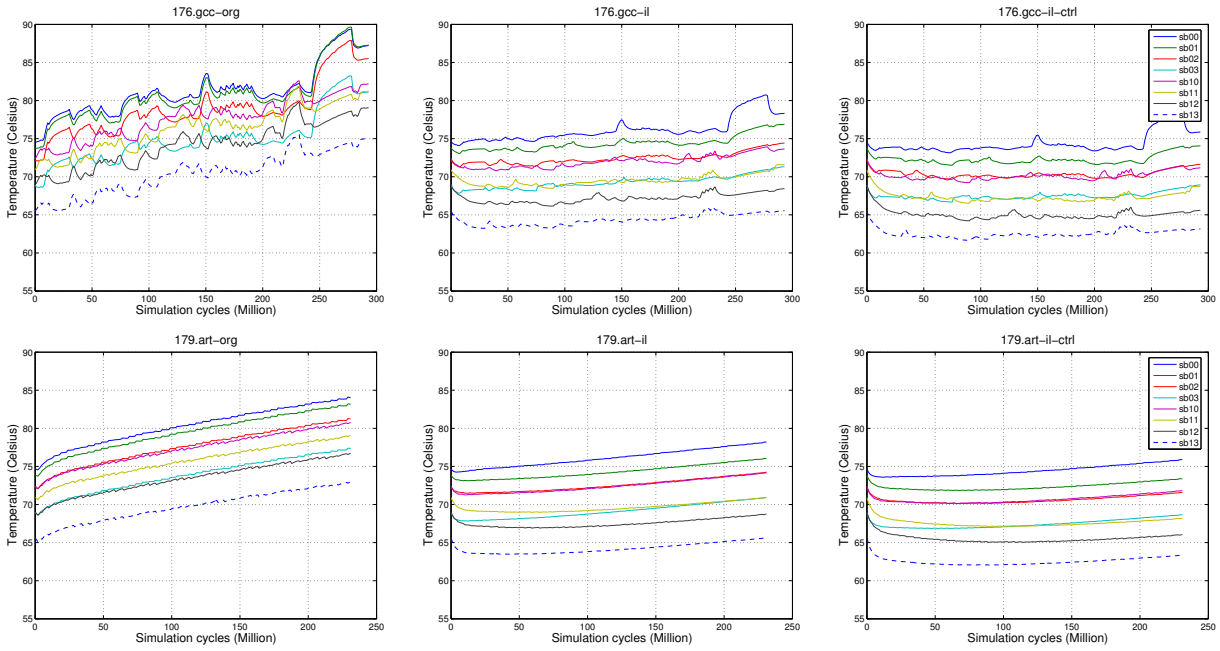


(a) Average for integer benchmarks (b) Average for floating-point benchmarks

Fig. 8. Steady subarray temperature comparison at 130nm technology

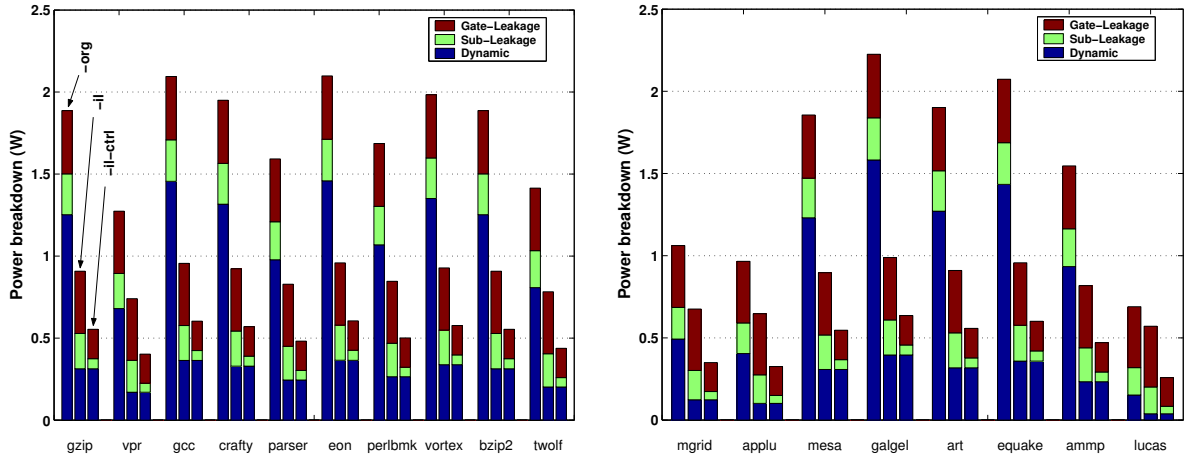
on the second row, and subarrays on the right have lower temperature than those on the left. This can be explained by the microprocessor floorplan in Figure 1. The data cache is neighboring with L2 cache (low temperature) on the bottom and the right side. Since the separated scheme has less impact on subarray temperature reduction, it is not considered in our analysis at the 70nm technology.

For the 70nm technology, the increasing power density creates much higher temperature in subarrays, as shown in Figure 9 for the two example benchmarks. With the simple subarraying scheme (-org), all subarray temperatures are sharply increasing (Figure 9(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 9(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 [30] with a temperature feedback from HotSpot [4] 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 10. The large gate leakage is partially due to the 1.0V supply voltage used in this evaluation. The exponential function of temperature for subthreshold leakage makes the situation even worse. Consequently, leakage optimization will play a critical role in processor thermal management at new technology generations. Since cache decay is more sensitive



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

Fig. 9. Thermal behavior of data cache subarrays at 70nm technology

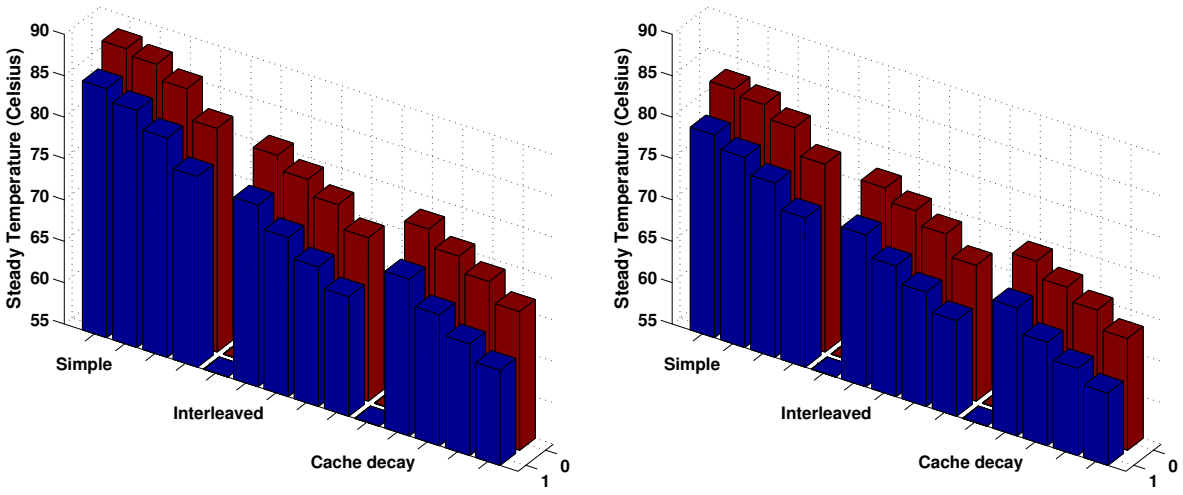


(a) Integer benchmarks (b) Floating-point benchmarks

Fig. 10. 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).

to temperature variations than the drowsy cache scheme [32], 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 [22]. Figure 9(c) shows the impact of leakage optimizations on the dynamic thermal behavior of cache subarrays. At the 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 11.

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 10. 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 dramatically reduces the subthreshold leakage (gate leakage) 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 reduced subarray temperature.



(a) Average for integer benchmarks

(b) Average for floating-point benchmarks

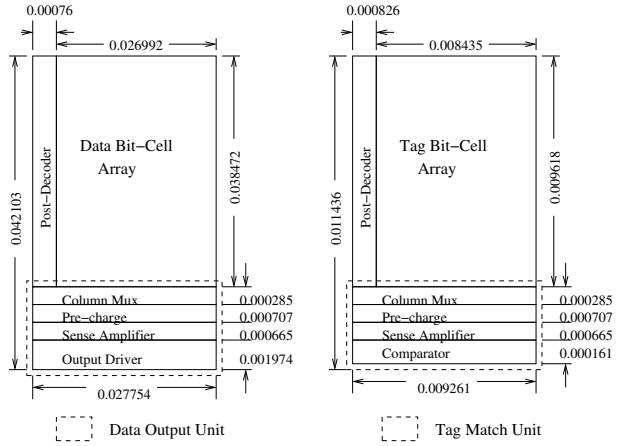
Fig. 11. Steady subarray temperature comparison at 70nm technology

6. Fine-Grain Thermal Modeling and Characterization

6.1. Fine-Grain Thermal Models for the Data Cache

To enable a more detailed and comprehensive thermal analysis of the data cache, we propose to model the thermal behavior of the data cache at fine granularity that captures the physical implementation details. We utilize the Cacti cache model [31] to derive the detailed geometric sizes and power distribution of the data cache at the macro-block level. According to Cacti [31], both the data and tag arrays are divided into subarrays to achieve better cache access and cycle time, by segmenting the bitlines and wordlines. Each subarray is largely an autonomic unit where both the bit-cell array and peripheral circuits are activated during the access to the subarray. In the following discussion, we use the data cache (64KB, 2-way associative, 64B cacheline size) from Alpha 21364 processors as an example to demonstrate our fine-grain thermal modeling. The cache is modeled at the 70nm technology. From Cacti, the data array of this data cache is divided into eight subarrays ($Ndbl$=2, $Ndwl$=4, and $Nspd$=1) and its tag array is divided into four subarrays ($Ntbl$=2, $Ntwl$=2, and $Ntspd$=1).

For a close look into the internal cache structures, we first breakdown the data subarrays into major functional blocks: bit-cell array, post-decoder, column multiplexer, pre-charge circuit, sense amplifier, and data output driver. The geometric dimensional sizes of these macro blocks are given in Figure 12(a). In general cache models, column multiplexer, pre-charge circuit, sense amplifier, and data output driver are stacked together at the bottom of the bit-cell array, and due to their small sizes in height, are thus combined into a single block as the data output unit in our fine-grain thermal model. In this modeled data cache, the output unit accounts around 9% of the die area of the data subarray. Similarly, the tag subarray is composed of tag bit-cell array, post-decoder, and tag match unit, as shown in Figure 12(b). The tag match unit is a composition of column multiplexer, pre-charge circuit, sense amplifier, and comparators, which contributes a share of 16% to the tag subarray area. However, due to its small dimension,



(a). Data subarray

(b). Tag subarray

Fig. 12. Layout of data cache data and tag subarrays (unit size: cm).

only about 9% of the data subarray in area, the tag subarray is treated as a single block in our thermal modeling without further dividing into smaller blocks. This is also to maintain our fine-grain thermal modeling at reasonable block sizes.

Built upon subarrays, a cache subblock consists of a number of subarrays, which form a one-dimensional or two-dimensional array. Subarrays within a subblock share the address bus and connect to part of the data bus. As shown in Figure 13, the coming-in address bus lines are routed to each subarray after the predecoder. For example, in subblock 0, the address bus after predecoder is shared between two subarrays, on the left or on the right. The data coming out from each subarray within the subblock is combined to form the required data to be sent out. According to Cacti model, the data cache (from Alpha 21346 processor) is constructed as two subblocks and each subblock consists of four subarrays, in between located data and address routings, which is given in Figure 13. Subblock 0 holds both two ways of sets with index from 0 to 255 and subblock 1 contains the rest half of the cache sets from 256 to 511. According to this organization, any

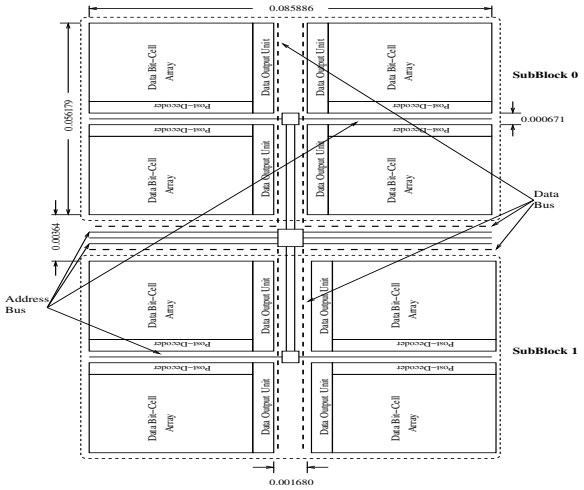


Fig. 13. Subblock-level view of the data cache (configuration from Alpha 21364).

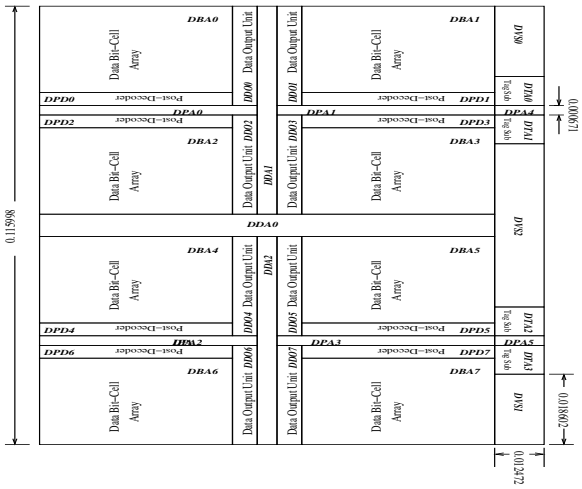


Fig. 14. Modeled floorplan of the data cache (configuration from Alpha 21364).

access to this data cache only requires to activate one subblock, determined by the set index of the fetch address. In contrast, the subblock of the tag array is much simpler and only consists of two tag subarrays and address routing. Similarly, with the interleaved subarray scheme,(-il) only a single subarray in the selected subblock of the data array will be accessed in order to reduce the per-access power consumption.

For better thermal behavior, we locate the tag array on the right of the data array, as shown in the modeled floorplan in Figure 14. Notice that this fine-grain (physical) subarray layout is slightly different from the one in Figure 6) used in the subarray-level thermal models, in order to closely match the real layout of the data cache in Alpha 21364 processors. These two subblocks share the data/address routing part DDA0. Compared to the data array, the tag array is relatively small. Tag subarrays DTA0 and DTA1 belong to tag subblock 0 and hold tags for way 0 and way 1 of sets in data subblock 0. Similarly, tag subarrays DTA2 and DTA3 hold tags for data in subblock 1. On both sides of the tag array, there is one empty void space, DVS0, DVS1, and DVS2. This fine-grain

TABLE V
MACRO BLOCKS IN THE FLOORPLAN OF THE MODELED DATA CACHE

Macro Blocks	Block Ids
Tag subarrays	DTA0, DTA1, DTA2, DTA3
Data subarray bit-cell arrays	DBA0, DBA1, DBA2, DBA3, DBA4, DBA5, DBA6, DBA7
Data subarrays post-decoders	DPD0, DPD1, DPD2, DPD3, DPD4, DPD5, DPD6, DPD7
Data subarray output units	DDO0, DDO1, DDO2, DDO3, DDO4, DDO5, DDO6, DDO7
Post-decode address routing	DPA0, DPA1, DPA2, DPA3, DPA4, DPA5
Data/address routing	DDA0, DDA1, DDA2
Void space	DVS0, DVS1, DVS2

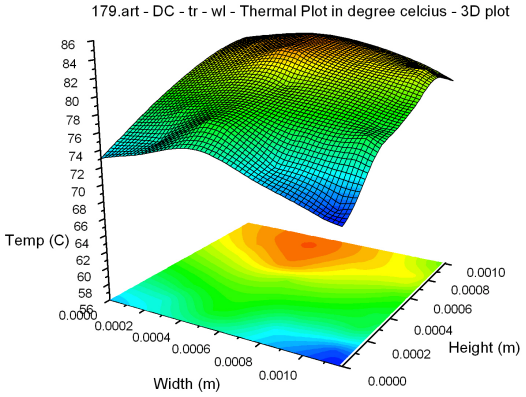


Fig. 15. Steady temperature map of the data cache with simple subarray scheme at 70nm technology (for benchmark 179.art).

thermal model tracks 40 macro blocks within the data cache to provide accurate and detailed cache thermal behavior. Table V categories the block ids for quick reference.

The dynamic power consumption of macro blocks such as tag subarray, data bit-cellarray, data post-decoder, and data output unit are directly derived from the Cacti model. At cache level, the dynamic power comes from three major components: tag array for 17.0%, data and address routing for 3.0%, and data array accounting for 80.0%. Within each data subarray, the output unit consumes more than half, 54.8% of the dynamic power per access, while the bit-cell array shares only 11.7% of the power consumption. Data routing power is distributed among DDA0, DDA1, and DDA2, where DDA0 accounts for 50%, and DDA1 and DDA2 each accounts for another 25% of the data routing power. The current model does not include the power consumption in post-decoder address routing (DPAs). The void-space blocks DVS0, DVS1, and IVS2 do not have any power consumption. A similar way is used to distribute the leakage power that is provided by HotLeakage [30].

6.2. *Fine-Grain Thermal Behavior of the Data Cache*

For experimental evaluation of this fine-grain thermal model for analyzing the detailed thermal behavior, the data cache floorplan in Figure 14 (from the Cacti model) is scaled into the target area in Alpha 21364. In this evaluation, we only show the results from benchmark *179.art* for the 70nm technology

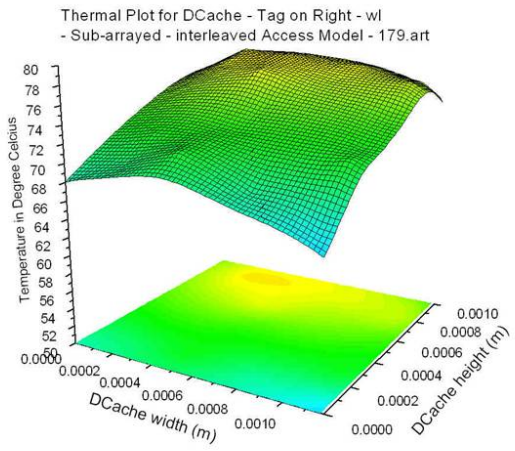


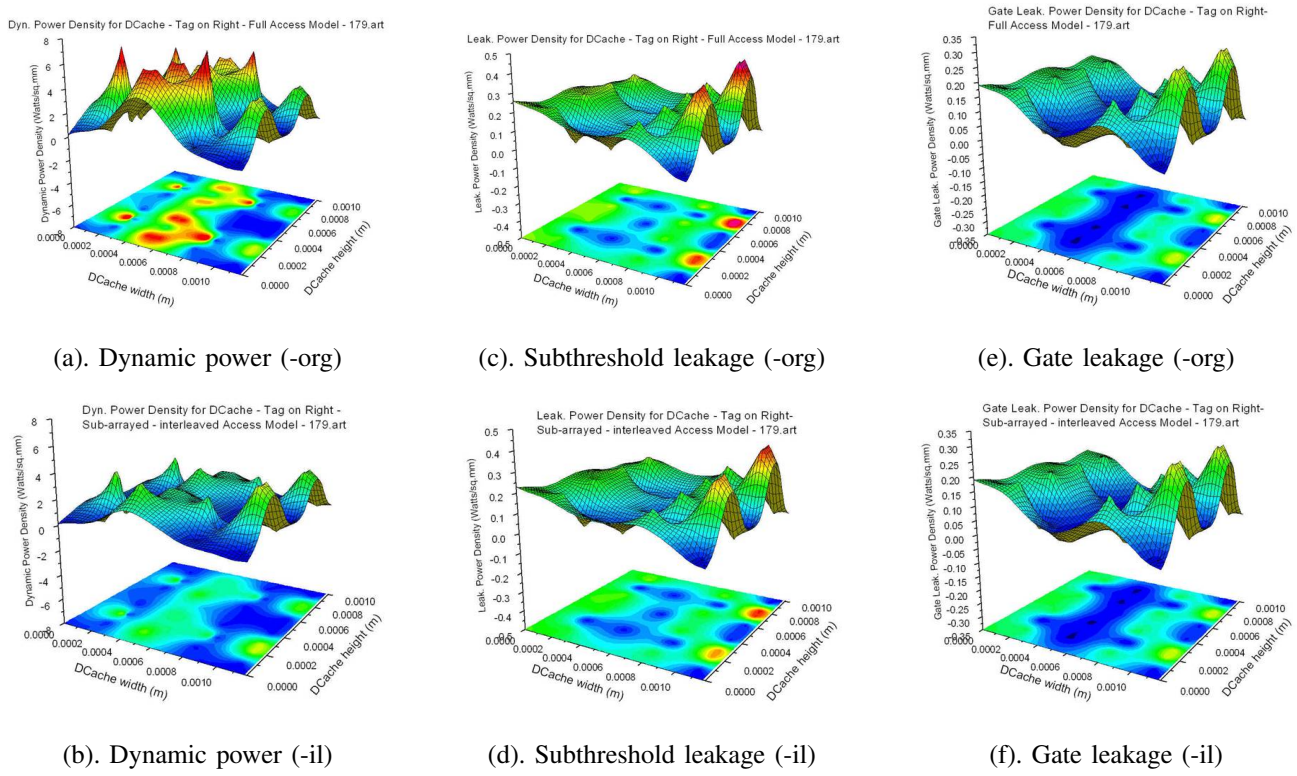
Fig. 16. Steady temperature map of the data cache with the interleaved subarray scheme at 70nm technology (for benchmark 179.art).

due to the space limit. With our new fine-grain thermal models, we are able to show a complete thermal map of the steady temperature in the data cache with the simple subarray scheme (-org), as given in Figure 15. The highest temperature (84.6°C) appears in the top portion of data/address routing block DDA1 that is between data output unit blocks DDO0 and DDO1. Temperature reduces gradually as a block's location is moving away to the bottom, left, and right sides from the hotspot. With our proposed way-interleaved scheme, the temperature of the hotspot in the data cache is reduced by 5.2°C to 79.4°C, as shown in Figure 16. Note that the location of the hotspot has slightly moved to the top-right corner of bit-cell array block DBA0 and the thermal map has the same moving trend as the one in Figure 15. From these two figures, it seems that most data cache accesses fall into subblock 0 (consisting of the top four subarrays) in the simple scheme and into subarray 0 in the interleaved scheme for benchmark *179.art*. Since the data cache is neighboring to the relatively hot DTLB from the top and to the relatively cool L2 cache modules from the right and bottom, the above speculation of the cache access activity may or may not be true, depending on the significance of heat diffusion in this scenario.

To further understand the thermal behavior within the data cache under heat diffusion between neighboring blocks, in Figure 17 we plot the dynamic power, subthreshold leakage, and gate leakage power densities for all modeled macro blocks of the data cache in both the simple and interleaved schemes. Figure 17(a) shows that in the simple scheme (-org), post-decoders (DPDs) have the highest dynamic power density, 7.52 Watts/sq. mm for DPD4/DPD5/DPD6/DPD7 and 6.32 Watts/sq. mm for DPD0/DPD1/DPD2/DPD3, followed by data output units, 6.17 Watts/sq. mm for DDO4/DDO5/DDO6/DDO7, and then 5.18 Watts/sq. mm for DDO0/DDO1/DDO2/DDO3. However, except for data output units DDO0 and DDO1 that are close to the hotspot (top portion of DDA1), all other high-power-density blocks exhibit consistently lower temperature. This dynamic power density distribution together with the thermal distribution in Figure 15 suggests that there is a weak correlation between the power density and the temperature of the modeled macro blocks when a strong heat-diffusion effect is ushered in. This in turn reemphasizes the importance of floorplan optimizations at multiple granularity levels to the thermal-aware designs. Figure 17(b) verifies the effectiveness of our proposed interleaved subarray scheme (-il) in achieving a low power data cache design: the dynamic power density of most macro blocks has been reduced by about half, except DDAs, and tag subarrays DTAs, which also brings down the hotspot temperature by 5.2°C. However, the interleaved scheme can only deliver a 10% reduction in subthreshold leakage power density over the simple scheme, as shown in Figures 17(c) and (d). A comparison between Figures 17(e) and (f) also indicates the very limited impact of the interleaved scheme on gate leakage optimization since leakage power consumption is largely independent of the access activity. This again suggests the critical need of leakage control schemes in addition to the thermal-aware subarrays schemes for effective thermal optimizations and dynamic thermal management in the data cache.

7. 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 analyze 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 is 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 propose 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. We further develop a fine-grain thermal model that captures physical implementation details of the modeled data cache to enable more comprehensive full-cache thermal behavior analyses as well as to explore the correlationship among per-access power consumption, heat diffusion, power density, and the resulting-in 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. For future work, we plan to extend our fine-grain thermal modeling and analysis for other major processor components such as register files and issue queues, to derive thermal-aware microarchitectures for them, to explore thermal-oriented multi-granularity floorplanning, and to exploit the interaction between thermal-aware microarchitectural designs and dynamic thermal management schemes to achieve an overall thermal-efficient processor microarchitecture for future generation processors.



(a). Dynamic power (-org) (c). Subthreshold leakage (-org) (e). Gate leakage (-org)

(b). Dynamic power (-il) (d). Subthreshold leakage (-il) (f). Gate leakage (-il)

Fig. 17. Dynamic, threshold leakage, and gate leakage power density comparisons between data caches with the simple subarray scheme (top row) and the interleaved subarray scheme (bottom row) at 70nm technology (for benchmark 179.art) (unit size: Watts/sq. mm).

REFERENCES

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

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

[3] David Brooks and Margaret Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture (HPCA'01)*, 2001.

[4] Kevin 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.

[5] 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.

[6] J. Montanaro and et al. A 160-mhz, 32-b, 0.5-w cmos risc microprocessor. *Digital Technical Journal, Digital Equipment Corporation*, 9, 1997.

[7] R. Bechade and e. al. A 32b 66mhz 1.8w microprocessor. In *Proc. of International Solid-State Circuits Conference*, 1994.

[8] J. K. John, J. S. Hu, and S. G. Ziavras. Optimizing the thermal behavior of subarrayed data caches. In *Proceedings of the IEEE International Conference on Computer Design (ICCD)*, Oct. 2005.

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

[10] Stephen H. Gunther, Frank Binns, Douglas M. Carmean, and Jonathan C. Hall. Managing the impact of increasing microprocessor power consumption. *Intel Technology Journal*, Q1, 2001.

[11] Chee How Lim, W. R. Daasch, and G. Cai. A thermal-aware superscalar microprocessor. In *Proceedings. International Symposium on Quality Electronic Design*, pages 517–522, 2002.

[12] W. Huang, K. Sankaranarayanan, K. Skadron, R. J. Ribando, and M. R. Stan. Accurate pre-rtl temperature-aware design using a parameterized geometric thermal model. *IEEE transaction on Computers*, 57(9):1277–1288, Sep. 2008.

[13] A. Merkel, F. Bellosa, and A. Weissel. Event-driven thermal management in smp systems. In *Proceedings of the Second Workshop on Temperature-Aware Computer Systems (TACS05)*, June 2005.

[14] M. D. Powell, M. Gomaa, and T. N. Vijaykumar. Heat-and-run: Leveraging smt and cmp to manage power density through the operating system. In *Proceedings of International Conference on Architectural Support for Programming Language and Operating System (ASPLOS04)*, Oct. 2004.

[15] J. Srinivasan and S. V. Adve. Predictive dynamic thermal management for multimedia applications. In *Proceedings of International Conference on Supercomputing (ICS03)*, June 2003.

[16] M. Hiraki et al. Stage-skip pipeline: A low power processor architecture using a decoded instruction buffer. In *Proc. International Symposium on Low Power Electronics and Des ign*, 1996.

[17] 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. International Symposium on Low Power Electronics and Des ign*, 1999.

[18] T. Anderson and S. Agarwala. Effective hardware-based two-way loop cache for high performance low power processors. In *IEEE Int'l Conf. on Computer Design*, 2000.

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

[20] 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.

[21] 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. International Symposium on High Performance Computer Architecture*, Jan. 2001.

[22] 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.

[23] 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 Annual International Symposium on Microarchitecture*, Istanbul, Turkey,, November 2002.

[24] N. Azizi, A. Moshovos, and F. N. Najm. Low-leakage asymmetric-

cell sram. In *Proc. the 2002 International Symposium on Low Power Electronics and Design*, Monterey, CA, 2002.
[25] 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.
[26] J. C. Ku, S. Ozdemir, G. Memik, and Y. Ismail. Thermal management of on-chip caches through power density minimization. In *Proceedings of IEEE/ACM International Symposium on Microarchitecture*, Nov. 2005.
[27] 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 Au stin, 2000.
[28] K. C. Yager. The MIPS R10000 superscalar microprocessor. *IEEE Micro*, 16(2):28–40, April 1996.
[29] 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.
[30] 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.
[31] P. Shivakumar and N. Jouppi. Cacti 3.0: An integrated cache timing, power, and area model. Technical report, Compaq Western Research Lab, August 2001.
[32] 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 Design, Automation and Test in Europe (DATE) Conference*, pages 22–27, Feb. 2004.

Jie Hu received his B.E. degree in Computer Science and Engineering from Beijing University of Aeronautics and Astronautics, China, in 1997, his M.E. degree in Signal and Information Processing from Peking University, China, in 2000, and his Ph.D. degree in Computer Science and Engineering from the Pennsylvania State University - UP in 2004. He has been an Assistant Professor in the Electrical and Computer Engineering Department at New Jersey Institute of Technology since 2004. His research interests are in the areas of computer architecture, power-aware systems design, power-efficient memory hierarchy, high-performance microprocessors, complexity-effective processor microarchitecture, power-efficient reliable systems, compiler optimizations for performance and power consumption, and reconfigurable computing architecture. He is a member of ACM, ACM SIGARCH, IEEE, and IEEE Computer Society.

Johnsy K. John received his B.Tech in Electronics and Communication Engineering from Mahatma Gandhi University, Kottayam, India, in 1999, and his M.S. in computer Engineering from New Jersey Institute of Technology (NJIT), Newark, NJ, in 2003. He is currently a Design Engineer with AMD Boston Design Center, working as a part of Instruction Fetch Micro-architecture team for AMD's Next Generation Microprocessors. Before joining AMD, he was an intern at Qualcomm Processor Design Center, Austin, Texas. He was also a PhD Candidate in the Department of Electrical and Computer Engineering, NJIT and is currently taking a break from the academic research work for his industry experience on microprocessors.

Shuai Wang received his B.S. degree in Computer Science from Nanjing University, China, in 2003. Currently, he is a Ph.D. candidate in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, and is a member of the Computer Architecture and Parallel Processing Lab (CAPPL). His research interests include power/thermal-aware systems design, reliable circuits and systems, reconfigurable computing architectures, and embedded systems. He is a student member of IEEE.