Energy Efficient D-TLB and Data Cache using Semantic-Aware Multilateral Partitioning

Hsien-Hsin S. Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
leehs@ece.gatech.edu

Chinnakrishnan S. Ballapuram
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
chinnak@ece.gatech.edu

ABSTRACT

The memory subsystem, including address translations and cache accesses, consumes a major portion of the overall energy on a processor. In this paper, we address the memory energy issues by using a streamlined architectural partitioning technique that effectively reduces energy consumption in the memory subsystem without compromising performance. It is achieved by decoupling the d-TLB lookups and the data cache accesses, based on the semantic regions defined by programming languages and software convention, into discrete reference substreams — stack, global static, and heap. Their unique access behaviors and locality characteristics are analyzed and exploited for power reduction. Our results show that an average of 35% energy can be reduced in the d-TLB and the data cache. Furthermore, an average of 46% energy can be saved by selectively multi-porting the semantic-aware d-TLBs and data caches against their monolithic counterparts.

Categories and Subject Descriptors

B.3.2 [**Memory Structures**]: Design Styles—*Associative memories, Cache memories, Virtual memory.*

General Terms

Design, Experimentation, Performance.

Keywords

low-power cache, low-power TLB, energy optimization, multi-ported memory structures.

1. INTRODUCTION

Innovations in microarchitecture and prominent advances in semiconductor process technology enable designers to create more sophisticated and powerful microprocessors; at the same time, they also lead to increased energy consumption. Power optimization has become the first priority in developing microprocessors for all market segments from mobile devices to high-end servers. Among all components, the memory subsystem including address translations and cache accesses consumes a major portion of the overall power on a processor. The translation lookaside buffer (TLB) draws a considerable amount of power, as it is typically organized as a fully associative cache and is accessed for every instruction and data fetch. Also due in part to more transistors available on a processor die, the capacity of caches in contemporary microprocessors tends to increase substantially in order to accommodate new application workloads. On the other hand, the philosophy of exploiting more instruction-level parallelism in superscalar processors necessitates a multi-ported d-TLB and cache to enable multiple memory references in parallel, further exacerbating energy consumption. Without proper management, the memory hierarchy alone can create serious problems in power dissipation and thermal control.

The TLB, a content addressable memory to expedite address translation, stores address mapping information and provides virtual-to-physical translation for each virtual address being accessed. When a TLB lookup misses, the lookup is routed to main memory and the corresponding address mapping is then reloaded into the TLB. Since a TLB is typically arranged as a fully associative cache, the energy consumed for each TLB lookup is quite significant. Measured data [16, 17] from commercial processors such as Intel's StrongARM and Hitachi's SH-3 reported that as much as 17% on-chip power is consumed in the TLBs and the trend is increasing. Similarly, frequently and recently used data are stored in caches closer to the processor to alleviate long memory latency. Caches due to their large capacity can consume over 40% [21] of the overall processor power.

In this paper, we address these energy issues by using a streamlined memory architecture partitioning technique that reduces energy consumption in the memory subsystems without compromising performance. It is achieved by decoupling data TLB lookups and data cache accesses, based on the semantic regions defined by programming languages and software convention, into discrete reference substreams — stack, global static, and heap. Their unique access behaviors and locality characteristics are analyzed and exploited for energy reduction. Our results for the SPECint2000 and Mibench [13] benchmark suites show that an average of 35% dynamic power can be reduced in the d-TLB as well as in the data cache. In addition, this technique also alleviates the requirement for multi-porting the ever larger TLBs and caches in multi-issue machines.

ISLPED'03, August 25–27, 2003, Seoul, Korea.

Our experiments indicate that an average of 46% power can be saved by selectively multi-porting the semantic-aware d-TLBs and data caches against their monolithic counterparts. Design options in terms of number of TLB and cache ports for cost-effectiveness and energy efficiency are also evaluated and reported in the paper.

This paper is organized as follows. Section 2 describes the memory reference behavior and our motivation for this work. Section 3 describes semantic-aware d-TLBs, data cachelets and data address router. Section 4 discusses and analyzes the performance, energy, and area results of our scheme. We then discuss related work in Section 5. Finally, Section 6 concludes our work.

2. MOTIVATION

According to the convention of programming languages, a specific processor architecture typically partitions a virtual address space into several non-overlapped semantic regions including instruction, static data, and dynamic data. Static data are comprised of *global* and *read-only* data allocated at compile-time while dynamic data can be further decomposed into *stack*, a storage holding the activation record created at runtime during subroutine calls, and *heap* dynamically allocated by functions such as malloc() in C. These semantic data regions, created with demands of different purposes, demonstrate different characteristics which can be exploited with dedicated hardware.

To investigate this property, data memory reference patterns are analyzed. Using cjpeg from MiBench benchmark suite as an example, Figure 1 illustrates the footprint distribution of data memory accesses based on the semantic regions defined by software convention and programming languages. Each point in the figure represents a data reference hit at a particular memory address. For both sub-figures, the low order 12 bits of data address are plotted on the x-axis while the high order 20 bits on the y-axis. In other words, the y-axis shows a *virtual frame number (VFN)* given a 4KB (2^{12}) page is used. All the points drawn on the same horizontal line belong to the same virtual memory page. The top figure plots stack accesses while the figure below plots both the global static and heap accesses. Due to their wide separation in the VFN, they are plotted as two separate sub-figures.

The figures show that addresses within a particular semantic region form a *semantic band*. Obviously, the *heap band* is wider and denser than the *stack* and *global static bands*, as the number of unique heap pages accessed is substantially higher than those of stack and global static regions. Also note that only a very small number of stack virtual pages relative to the number of global static and heap pages is needed.

Based on the observation in Figure 1, we now examine the number of compulsory misses in the d-TLB and in the data cache for programs from MiBench and SPECint2000[1]. The MiBench suite, mimicking EDN's EEMBC benchmark suite, is comprised of representative embedded applications collected from networking, security, telecommunication, image processing, etc., while the SPEC2000 integer benchmark suite is often used for evaluating high performance systems. All programs were compiled into ARM binary, our target machine, with -O3 switch.

Figure 2 shows the number of compulsory d-TLB misses,

[1]Programs from MiBench were run to the end while four selected programs from SPECint2000 suite were run to 3 billion instructions for all simulated results shown in this paper.

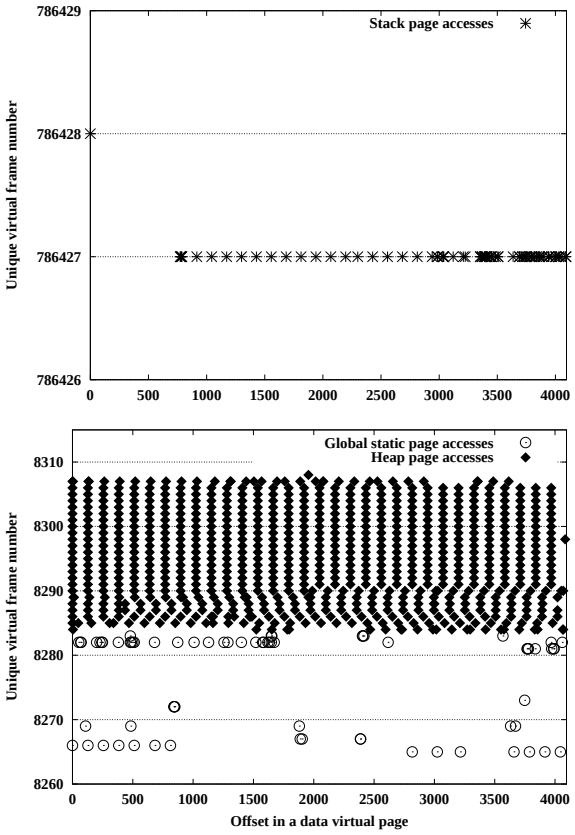

Figure 1: Dynamic address footprint distribution of semantic regions

i.e. the number of unique virtual data memory pages, accessed by a program from each semantic region. The memory page size is 4KB, a typical minimum page size used in modern operating systems. Note that data were plotted at log scale due to the large amount of heap references. Both benchmark suites show a similar trend of stack memory page utilization, which can be satisfied by no more than 5 pages in most of the programs. The number of global static pages is somewhat dependent on the behavior of a given application. Nonetheless, it is also significantly less than the number of heap pages averaging 15 pages.

Figure 3 shows the number of compulsory data cache misses given a 32-byte data cache line. Similar trends to the d-TLB are observed. It is worth noting that the SPECint2000 benchmarks demonstrate huge compulsory misses in the heap region compared to those shown in the MiBench. This suggests that data working sets tend to expand at a faster rate in the heap region than in the stack and global regions. We now quantify the distribution of dynamic data references in Figure 4.

In Figure 4, the number of accesses to each semantic region is normalized to the total number of data references. In addition to stack, global static, and heap, two additional regions are illustrated: text and environment (env) associated with data references. The reason data accessing text is explained as follows. Instead of using a global pointer to access global data, the ARM compiles the base addresses of all the global static data as part of the text space. Each global data is referenced in an indirect manner through two instructions. The first instruction, a PC-relative load, fetches the base address from *text region*, and the second

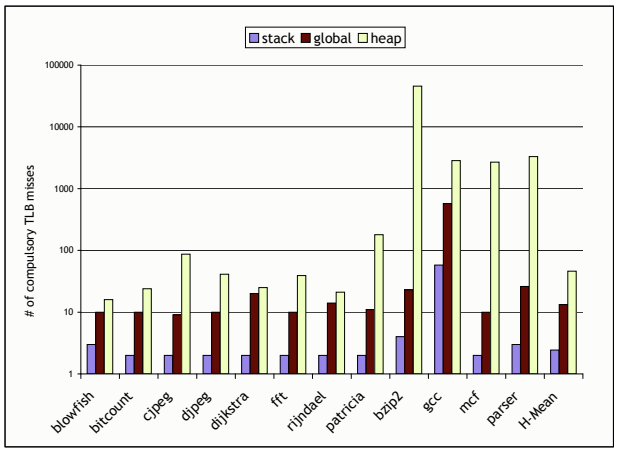

Figure 2: Compulsory data-TLB misses

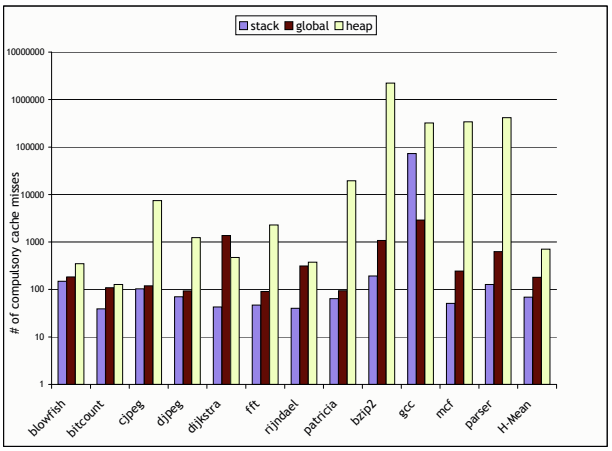

Figure 3: Compulsory data cache misses

one retrieves the actual value of the global variable based on the prior fetched address. This indirect load for global data not only degrades performance but also causes a potential data duplication inside both the d-TLB and data cache. This property can be further exploited by reorganizing the instruction-TLB for energy and space savings purposes, which is beyond the scope of this paper. The number of accesses to the env region, however, is rather insignificant as opposed to the other regions. The critical information conveyed in Figure 4 is that the majority (∼50%) of the dynamic memory accesses go to the stack. Figure 1, Figure 2, and Figure 3 combined indicate that although data references largely hit in the stack, these accesses concentrate on very few memory pages. This leads to a new opportunity for reorganizing the data memory architecture for energy and performance optimization.

3. SEMANTIC-AWARE MULTILATERAL PARTITIONING

From the analysis shown in Section 2, we propose a new partitioning scheme for memory architectures called Semantic-Aware Multilateral or *SAM* partitioning. The idea is to leverage the data reference characteristics to disperse data memory accesses into discrete SAM substreams and redirect each stream into their own exclusive architectural component. The SAM memory architecture, as depicted in Figure 5, exploits the locality and characteristics demonstrated in each semantic region by (1) reorganizing the first level TLB structure into two small structures — stack and global static micro-TLBs [7], while leaving the second level [9] for all data addresses and (2) splitting the first level cache into three cachelets. The SAM TLBs and the SAM Cachelets are hereafter referred to as *SAT* and *SAC*, respectively.

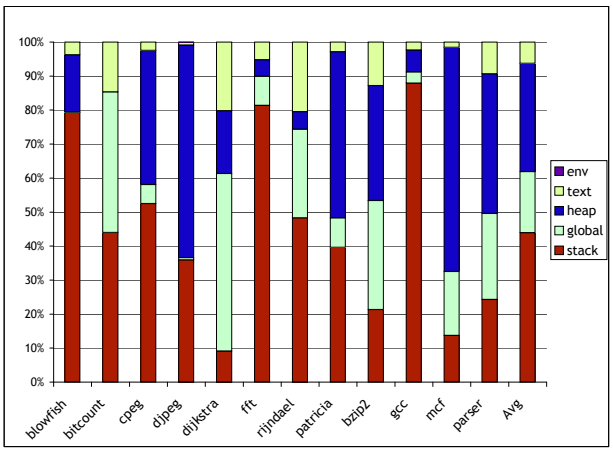

Figure 4: Dynamic data memory distribution for Mibench and SPECint2000

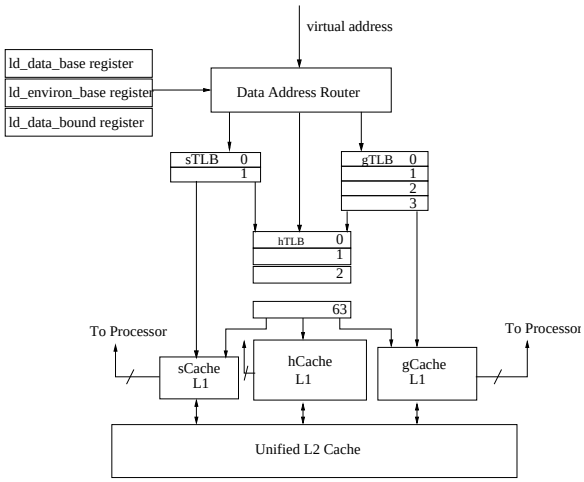

Figure 5: SAM memory architecture

3.1 Semantic-Aware Data TLBs

In the SAM partitioning approach, we implement a Data Address Router (DAR) that routes each d-TLB lookup to a stack TLB (sTLB), a global static TLB (gTLB), or a heap TLB (hTLB) for address translation based on the higher order bits of the virtual address. Note that the hTLB is also used as the second level (backing store) TLB for the sTLB and gTLB. When loading a program for execution, the system loader communicates the following virtual addresses: *ld_environ_base* (top of stack), *ld_data_base* (global static base), and *ld_data_bound* (heap base) to special hardware control registers. The DAR logic routes each address to its destination based on the ranges derived from the control registers. Instead of searching the entire fully associative TLB entries, energy can be saved by filtering out the lookups by stack and global static regions with less

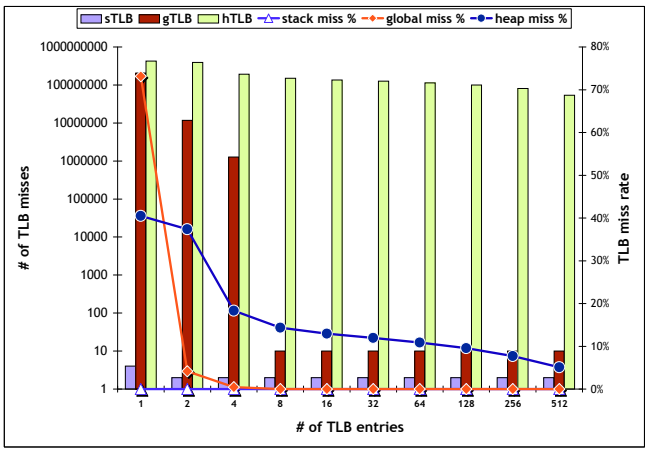

Figure 6: Semantic-aware TLB misses

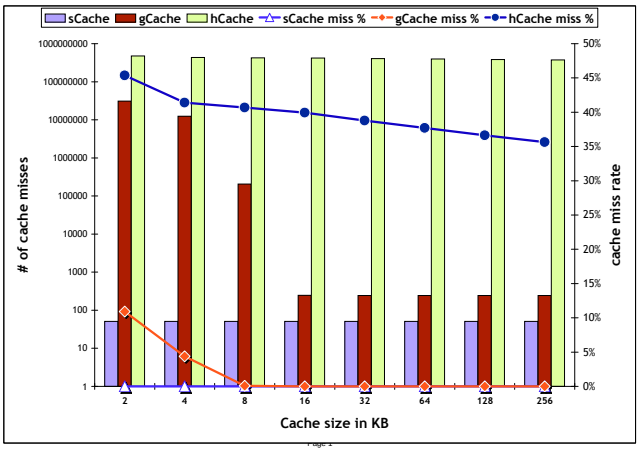

Figure 7: Semantic-aware cache misses

hardware. Lookups missing in the two-level TLB are directed to main memory. If the virtual address does not fall under the stack or global static region, it is directed to the heap TLBs. For example, the upper portion of virtual memory used by the kernel mode does not fall in the stack or global static region, thus it is directed to heap TLBs.

Using 181.mcf of SPECint2000 as an example for analysis, Figure 6 shows the number of TLB misses and miss rates when discrete SATs are implemented. The number of misses is plotted at log scale. The number of TLB entries on the x-axis is exponentially increased from 1 to 512 in order to measure the sensitivity of TLB miss behavior. Not surprisingly, the number of sTLB misses saturate when the number of entries reaches two, while the gTLB saturates at 8 entries. On the other hand, the hTLB misses do not drop as drastic as its counterparts, containing over 53 million misses even with 512 entries. This observation implies that under a unified TLB structure the more dynamic heap TLB lines could evict the more stable stack and/or global TLB lines, thereby leading to unnecessary TLB conflict misses that can be avoided by using a SAT implementation. Besides the advantage of eliminating TLB conflict misses among different semantic regions, the energy can be reduced due to a major portion of the memory access distribution being skewed towards the much smaller sTLB and gTLB. Moreover, multi-porting the micro-TLBs becomes more feasible under a given constraint of die area and access latency. All these advantages will be quantified later in Section 4.

3.2 Semantic-Aware Data Cachelets

Similar to the SATs, the first level data cache can be semantically partitioned into discrete semantic-aware cachelets (SAC). The energy saved could be substantial as energy dissipated in caches constitutes a major portion of the overall energy consumption. Multi-porting the ever-increasing caches for superscalar processors also exacerbates leakage dissipation. The SAC scheme provides an alternative solution that enables selective multi-porting for only the highly accessed SACs, e.g. stack cache.

Figure 7 shows a similar graph of Figure 6 for data caches. The size of each SAC varies from 2KB to 256KB. Again, stack demonstrate very stable working set size with respect to the other 2 semantic regions. Global misses saturate at a reasonable capacity whereas heap data appear to be less tractable. Therefore, the majority of the memory references can be captured by semantically partitioning the cache structure into a small stack cache (sCachelet), a small global static cache (gCachelet) and a larger cache (hCachelet) for accommodating all the rest, including heap, text, and env for the ARM ISA we simulate. This new SAC scheme can substantially reduce energy consumption while retaining the performance. For multi-issue machines, savings can be higher as only smaller sCachelet and gCachelet need to be multi-ported.

4. EXPERIMENTS AND ANALYSIS

Our simulation infrastructure is based on Simplescalar for the ARM ISA [3] for performance evaluation. We integrated Wattch [6] into the Simplescalar ARM model for energy simulation and made changes to enable our studies for the SAM memory architecture. We take the *clock-gating mode 3* from Wattch for all energy numbers in this paper. This mode is more practical as it considers a constant energy dissipated (i.e. leakage energy) even when an idle functional unit is clock-gated[2]. Table 1 lists the processor parameters used in our baseline machine model. The SAC sizes used in the following sections are identical across all benchmark programs. The SAT sizes which are changed slightly for different benchmarks will be discussed in Section 4.1.

4.1 Performance vs. Energy with SAM

We ran a wide spectrum of simulations to collect data for making best selection of the optimal TLB entries for each benchmark application. As discussed in Section 3, there exists a knee point after which the performance gain

[2]For the same functional unit, we assume the leakage power consumes 10% of the dynamic power.

Execution Engine	out-of-order
Fetch/Decode Width	4 / 4
Issue/Commit Width	4 / 4
L1 cache hit latency	1 cycle
L2 cache hit latency	6 cycles
Memory latency	150 cycles
TLB hit latency	1 cycle
TLB miss latency	30 cycles
Cache property	Direct-mapped, 32B line
L1 Cache baseline	32KB
L1 s/g/hCachelet	8KB / 8KB / 16KB
L2 Cache	4-way 512KB, 32 bytes line

Table 1: Processor model parameters

Benchmark:	blowfish	bitcount	cjpeg	djpeg	dijkstra	fft	rijndael	patricia	bzip2	gcc	parser
dTLBbase	32	32	128	64	64	64	32	256	64	64	64
sTLB	2	2	2	2	2	2	4	2	4	4	4
gTLB	8	8	8	8	32	8	8	8	16	16	16
hTLB	16	32	128	64	32	64	32	256	64	64	64

Table 2: Cost-effective TLB configuration

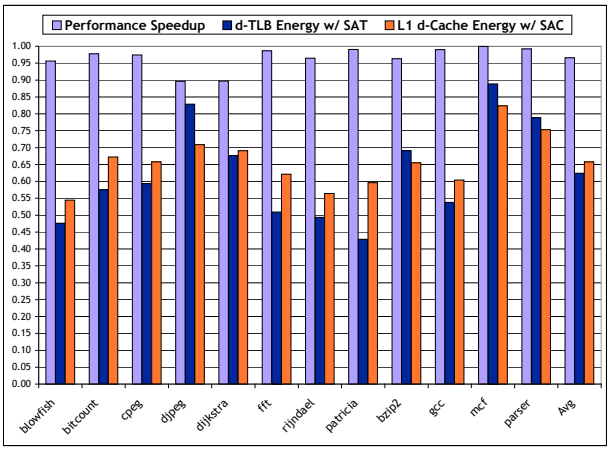

Figure 8: Design effectiveness of SAM (baseline=1.0)

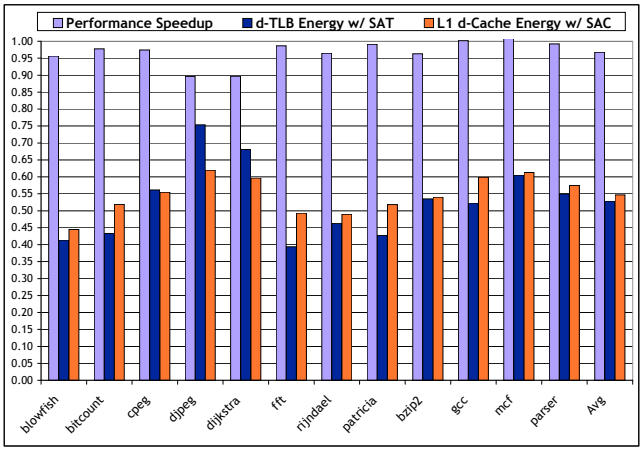

Figure 9: Multi-porting effectiveness of SAM (baseline=1.0)

Number of d-TLB entries			
base	sTLB	gTLB	hTLB
64	4	16	64
2 ports	2 ports	1 port	1 port
Data cache sizes			
base	sCachelet	gCachelet	hCachelet
32KB	8KB	8KB	16KB
2 ports	2 ports	1 port	1 port

Table 3: Multi-porting Configuration

is diminishing regardless how many more TLB entries are added. The rationale behind this is that one could configure the most cost-effective number of TLB entries for an application by V_{dd}-gating off the ineffective TLB entries. Table 2 summarizes the most effective number of TLB entries for each benchmark. In most of the cases, the SAT approach has slightly more TLB entries than the baseline without SAM partitioning except for the blowfish, in which the combined SAT size is slightly smaller than the baseline case. Using the configuration in Table 2 and other design parameters in Table 1, we generate the performance as well as the TLB and cache energy reduction[3] as shown in Figure 8. Similar trends are observed in this figure for both MiBench and SPECint2000, giving an average of 36% energy reduction in TLBs and 34% in caches with an average 4% performance loss.

4.2 Multi-Porting SAT and SAC

4.2.1 Performance vs. Energy

For contemporary applications and workloads, it is observed that one out of every two to three instructions is a memory operation. It implies that a multi-issue machine needs to access memory (dTLB and data cache) more than once per cycle, necessitating a multi-ported design. Multi-porting an ever-increasing monolithic TLB and cache aggravates energy consumption and increases die area. As shown in Figure 4, the average ratio of memory accesses to stack, global and heap is close to 2:1:1. In other words, we can multi-port the SAT and SAC by using this ratio. Figure 9 evaluates the performance and energy for such a design using a 4-wide SAM machine with the configuration listed in Table 3. The energy of a SAM design shows a 47% reduction in data-TLB and 45% reduction in the data cache with almost on-par performance of a baseline machine.

4.2.2 Area and Access Latency Estimation

Table 4 evaluates the transistor areas of different SAC schemes using CACTI 3.0 [15]. As shown in the table, selective dual-porting SAC effectively reduces the access time by as much as 50% and transistor area by 32 to 41%. Also note the leakage energy (not shown in the table) can be saved proportionally as the transistor count is reduced.

5. RELATED WORK

Many memory partitioning schemes at an architectural level were proposed for energy reduction. These techniques can be classified into two categories: horizontal partitioning and vertical partitioning. Vertical partitioning techniques such as line buffers [12, 22] and filter caches [18] attempt to capture cache lines in a small intermediate structure to exploit transient data locality at a cost of performance loss. Sub-banking [12], selective cache ways [1], and PSAC [14] partition the caches into horizontal segments and only enable the partitions that are being accessed at runtime. The HP3000 Series II [5] featured a stack cache with FIFO replacement policy as an extension to main memory in the absence of a data cache. The CRISP processor [4] performed register allocation on the 32-entry stack cache at runtime to avoid the overheads of procedure calls. Cho et al. [8] proposed a decoupled stack cache to alleviate

[3]We do not use E*D product as the metric for our analysis as the E(nergy) component highly depends on what resources are included into the evaluated system, in which D(elay) component is measured. The argument is presented by Lee et al. in [19].

Cache Model	32KB Unified	8KB sCachelet	8KB gCachelet	16KB hCachelet	Total SAC area	Area savings
R/W ports	2	2	1	1		
Access time (ns)	1.125	0.826	0.692	0.816		
Area in mm^2	5.304	1.393	0.616	1.095	3.104	41.5%
Model	64KB Unified	16KB sCachelet	16KB gCachelet	32KB hCachelet	Total SAC area	Area savings
Access time (ns)	1.630	0.949	0.816	0.948		
Area in mm^2	8.942	2.555	1.095	2.246	5.897	34.1%
Model	128KB Unified	32KB sCachelet	32KB gCachelet	64KB hCachelet	Total SAC area	Area savings
Access time (ns)	2.238	1.125	0.948	1.196		
Area in mm^2	17.24	5.304	2.246	4.115	11.666	32.3%

Table 4: Access time and die area comparison

the performance degradation in multi-ported caches. Lee and Tyson proposed region-based caching [20], a preliminary concept for semantic-aware data caches. Collins et al. [10] studied a pointer cache to enhance indirect loads. Their mechanism recognizes heap address accesses dynamically based on a method in [11] and inserts them into the pointer cache.

More recently, Ashok et al. [2] used a compiler-managed address speculation to enable tagless cache accesses for power reduction. Kadayif et al. [17] proposed a mechanism that reduces power of the i-TLB by adding a register called Current Frame Register (CFR) for address translation. Instead of looking up the i-TLB for each instruction, the processor fetches the translated address from the CFR unless there is a memory page change.

6. CONCLUSIONS AND FUTURE WORK

This paper presents a streamlined Semantic-Aware Multilateral (SAM) memory design technique that effectively reduces the energy consumption in memory, including data-TLB and data cache with minimal performance impact. Due to the unique memory reference characteristics, our SAM approach reduces dynamic energy by redirecting the majority number of accesses to the much smaller SA-TLB (SAT) and SA-Cache (SAC). It is found that the number of SAT entries in the stack can be as few as 2 to cover almost all the stack references. We discuss a design of data address router that splits and routes each access to the corresponding SAT and SAC. Using our technique, we show that an average of 35% energy can be reduced in the d-TLB and in the data cache. The effect of multi-porting SAT and SAC is also investigated. As our experiments indicated, both energy and transistor area can be substantially reduced by 46% and 41% respectively, by multi-porting only heavily accessed SAC and SAT instead of multi-porting its ever-increasing monolithic counterpart. This technique is also orthogonal to prior vertical and horizontal partitioning schemes. Prior art such as filter caches or line buffers can be easily implemented on top of our SAM scheme to acquire more energy savings.

Future work involves using the compiler to provide architectural hints in the ISA for the number of stack and static pages used in a program. The architecture can use this information to dynamically configure, e.g. with a V_{dd} gating technique, the most cost-effective number of dTLB entries for an application to optimize the energy consumption. We can further semantically partition the stream of data addresses present in the text region, a code generation convention in ARM, to another semantic aware structure for mitigating aliasing (i.e. conflict misses) and achieving additional benefits in energy and performance.

7. ACKNOWLEDGEMENTS

This work is sponsored by a grant from the Center for Experimental Research in Computer Systems (CERCS), as part of its Industry Affiliates Program at Georgia Tech.

8. REFERENCES

[1] D. H. Albonesi. Selective Cache Ways: On-Demand Cache Resource Allocation. In *MICRO-32*, 1999.

[2] R. Ashok, S. Chheda, and C. A. Moritz. Cool-Mem: Combining Statically Specuative Memory Accessing with Selective Address Translation for Energy Efficiency. *ASPLOS-X*, 2002.

[3] T. M. Austin. Simplescalar 4.0 Release Note. http://www.simplescalar.com/.

[4] A. D. Berenbaum, D. R. Ditzel, and H. R. McLellan. An Introduction to the CRISP Architecture. In *Proceedings of the Spring COMPCON*, 1987.

[5] R. P. Blake. Exploring a Stack Architecture. *IEEE Computer*, May 1977.

[6] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. *ISCA-27*, 2000.

[7] J. B. Chen, A. Borg, and N. P. Jouppi. A Simulation Based Study of TLB Performance. In *ISCA-19*, 1992.

[8] S. Cho, P.-C. Yew, and G. Lee. Decoupling Local Variables Accesses in a Wide-Issue Superscalar Processors. In *ISCA-26*, 1999.

[9] J.-H. Choi, J.-H. Lee, S.-W. Jeong, S.-D. Kim, and C. Weems. A Low power TLB structure for Embedded Systems. *IEEE Computer TCCA Letter*, January 2002.

[10] J. Collins, S. Sair, B. Calder, and D. Tullsen. Pointer Cache Assisted Prefetching. In *MICRO-35*, 2002.

[11] R. Cooksey, S. Jourdan, and D. Grunwald. A Stateless, Content-Directed Data Prefetching Mechanism. *ASPLOS-X*, 2002.

[12] K. Ghose and M. B. Kamble. Reducing Power in Superscalar Processor Caches using Subbanking, Multiple Line Buffers and Bit-Line Segmentation. In *ISLPED'99*, 1999.

[13] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. MiBench: A Free, Commercially Representative Embedded Benchmark Suite. In *IEEE 4th Workshop on Workload Characterization*, 2001.

[14] M. Huang, J. Renau, S.-M. Yoo, and J. Torrellas. Cache Decomposition for Energy-Efficient Processors. In *ISLPED'01*, 2001.

[15] N. Jouppi. CACTI 3.0. http://research.compaq.com/wrl /people/jouppi/CACTI.html, 1999.

[16] T. Juan, T. Lang, and J. J. Navarro. Reducing TLB Power Requirements. In *Proceedings of the International Symposium on Low Power Electronics and Design*, 1997.

[17] I. Kadayif, A. Sivasubramaniam, M. Kandemir, G. Kandiraju, and G. Chen. Generating Physical Addresses Directly for Saving Instruction TLB Energy. In *MICRO-35*, 2002.

[18] J. Kin, M. Gupta, and W. H. Mangione-Smith. Filtering Memory References to Increase Energy Efficiency. *IEEE Transactions on Computers*, Vol. 49, No. 1, 2000.

[19] H.-H. S. Lee, J. B. Fryman, A. U. Diril, and Y. S. Dhillon. The Elusive Metric for Low-Power Architecture Research. In *Workshop on Complexity-Effective Design in conjunction with ISCA-30*, 2003.

[20] H.-H. S. Lee and G. S. Tyson. Region-Based Caching: An Energy-Delay Efficient Memory Architecture for Embedded Processors. In *CASES'00*, 2000.

[21] J. Montanaro and et al. A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor. *Digital Technical Journal*, Vol.9 No.1, November 1996.

[22] C.-L. Su and A. M. Despain. Cache Design Trade-off for Power and Performance Optimization: A Case Study. In *Proceedings of the International Symposium on Low Power Design*, 1995.