An Integrated Approach to Reducing Power Dissipation in Memory Hierarchies

Jayaprakash Pisharath
Department of Electrical & Computer Engineering
Northwestern University
Evanston IL 60208
Phone: 1–847–467–2299
jay@ece.northwestern.edu

Alok Choudhary
Department of Electrical & Computer Engineering
Northwestern University
Evanston IL 60208
Phone: 1–847–467–4159
choudhar@ece.northwestern.edu

ABSTRACT

In recent years, both performance and power have become key factors in efficient memory design. In this paper, we propose a systematic approach to reduce the energy consumption of the entire memory hierarchy. We first evaluate an existing power-aware memory system where memory modules can exist in different power modes, and then propose on-chip memory module buffers, called Energy-Saver Buffers (ESB), which reside in-between the L2 cache and main memory. ESBs reduce the additional overhead incurred due to frequent resynchronization of the memory modules in a low-power state. An additional improvement is attained by using a model that dynamically resizes the active cache based on the varying needs of a program. Our experimental results demonstrate that an integrated approach can reduce the energy-delay product by as much as 50% when compared to a traditional non power-aware memory hierarchy.

Categories and Subject Descriptors

C.4 [**Performance of Systems**] – *design studies, measurement techniques, performance attributes*; B.3.2 [**Memory Structures**]: Design Styles – *cache memories, primary memory*; B.3.3 [**Memory Structures**]: Performance Analysis and Design Aids – *simulation*

General Terms

Algorithms, Design, Measurement, Performance

Keywords

Dynamic Cache, energy-delay product, Energy-Saver Buffers (ESB), integrated approach, power, RDRAM

1. INTRODUCTION

One of the major challenges that the hardware industry faces today is to cope with the immense power requirements of integrated circuits. Most of the studies and proposals for minimizing the power requirements of computing systems are directed towards processors and peripherals. Researchers recommend the deployment of different power modes based on the idleness of processors. For instance, the mobile Pentium III processor has five power management modes [23]. The RDRAM technology [12, 24] enables even memories to operate in any of the six low power modes. Similar methods have been proposed for minimizing the power consumption of disks and other peripherals [22]. These low-power modes of operation work well when the system is idle, but one has to pay the price for resynchronization overhead when the hardware goes to the active state from an idle one.

In this paper, we investigate the entire memory hierarchy to subsequently achieve considerable savings in terms of both delay and power dissipation. We efficiently mask the resynchronization costs associated with reactivating memory modules from a low-power state by introducing Energy-Saver Buffers. An effort to reduce the energy consumption is made at every level of the memory hierarchy. A reasonable power-performance tradeoff is also targeted for the proposed low-power design. Our experimental results demonstrate that such an integrated approach to designing a power-aware memory hierarchy can reduce the energy-delay product by almost 50% of the traditional memory hierarchies, while incurring a considerably shorter delay than other suggested power-aware schemes.

The rest of the paper is organized as follows. Section 2 focuses on some of the related work in the field of low-power memory design and the implications on our work. Section 3 outlines the design of a power-aware modular memory that forms the base model for our work. This section also covers the experimental setup. In Section 4, we present the design of our Energy-Saver Buffers (ESB), concentrating on mainly their functionality. Section 5 elaborates a dynamic cache design that can co-exist with our ESBs in the memory hierarchy. In Section 6, we discuss our results for a power-aware memory hierarchy that integrates our previous designs with a focus on ESB. Section 7 summarizes the overall results and the paper.

2. RELATED WORK

Memory is a very common candidate for low power design. Lebeck et al. proposed a power-aware page allocation scheme for the main memory [21]. In this work, the different operating modes of RDRAM are exploited to improve the energy-delay product. This dynamic scheme forms the basis for our work. Another approach by Delaluz et al. introduced novel techniques to exploit the low-power operating modes of DRAMs [11]. These techniques include heuristics that use fixed thresholds for detecting idleness and an adaptive threshold that attempts to adjust itself with the dynamics of a program.

CASES 2002, October 8–11, 2002, Grenoble, France.

Significant work has been done on designing low-power caches. An integrated architectural and circuit-level approach by Se-Hyun et al. aims at reducing leakage energy in instruction caches [28]. They introduced DRI i-cache, an i-cache that can dynamically resize and adapt itself to an application's requirements. Kamble and Ghose proposed architectural techniques to reduce power consumption in on-chip caches, and also provided analytical models for estimating energy dissipation of conventional caches as well as low-power caches [14, 17]. Compiler and hardware-based approaches to reduce cache misses, by Sherwood et al. [25], suggest dynamic reordering of pages in physically addressed caches. In [1], Albonesi proposed an on-demand cache resource allocation policy called selective cache ways. Selective cache ways technique gives the ability to disable a subset of 'ways' in a set associative cache during periods of modest cache activity, while the full cache may remain operational for more cache-intensive periods. A mechanism called Cache Decay, proposed by Kaxiras and Hu [19], exploits generational behavior of caches to reduce cache leakage power. Bahar et al. have studied power-performance tradeoffs for several power/performance sensitive cache configurations, which involve techniques like increasing cache size or associativity and including buffers along side L1 caches [3].

Brooks et al. proposed a framework for analysis and optimization of power at the architectural level [8]. Wattch provides a power evaluation methodology within the portable and familiar SimpleScalar [2, 9] framework. Our model for power estimation is based on the Wattch model.

Our work differs from all of the preceding research in that we propose an integrated approach for building a memory model that optimizes power at each level of the memory hierarchy. As a result, we study the outcome of combining many of the existing power-aware schemes described above. A separate unique contribution involves the introduction of Energy-Saver Buffers (ESB), which are meant to reduce the power and delay overhead associated with the frequent resynchronization of memory modules in a low-power state.

3. POWER-AWARE PAGE ALLOCATION (Base Model)

Our base model follows the Power-Aware DRAM model (PADRAM) as suggested by Lebeck et al. [21] (Figure 1b). In this scheme, RDRAM [12, 24] functions in one of the four power modes, namely: active, standby, nap and power-down. The active state consumes the maximum energy during operation. A memory module will gradually go down to a lower-power state based on the number of cycles it remains inactive. The power-down state consumes the minimum energy during operation. A memory module in a low-power state is reactivated on a read or write operation.

For our base model, we use the sequential first touch page placement policy, which allocates pages sequentially in a single memory module until it gets completely filled, before moving on to the next module. Thus all the pages would restrict themselves to fewer memory modules than is the case for random placement, where pages are spread across all memory modules.

It is evident that this PADRAM scheme incurs a sizeable delay and an additional power dissipation during resynchronization of the memory modules. In Section 4, we propose on-chip Energy-Saver Buffers for smoothing out the high frequency of transitions between different power modes, in an attempt to improve upon this base model. In the rest of this section, we discuss both our experimental methodology and the results obtained after implementing the PADRAM scheme, with an emphasis placed on points of our interest.

3.1 Experimental Setup

For our experiments, we used the SimpleScalar/Arm architecture simulator [2, 9] modified to incorporate a detailed power model. The benchmarks used were binaries of applications from the SPECint2000 suite, Mediabench suite and some other custom applications. These binaries were built using GNU GCC version 2.95.1. Table 1 gives a summary of the applications used as benchmarks.

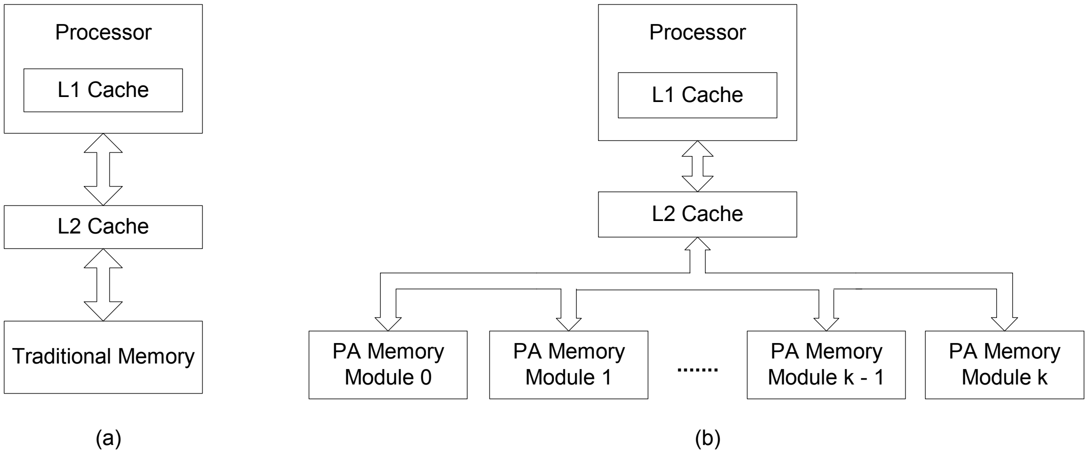


Figure 1. (a) Traditional memory without any power-aware schemes and (b) the equivalent power-aware (PA) DRAM implementation with power-aware memory modules. Cache shown here is a traditional cache, where all sets are active.

Table 1. Benchmarks used for experiments

Benchmarks	Description	Instructions Executed (millions)
anagram	Anagram	425
bzip	Compressor	137
gcc	GNU GCC	235
grep	Text search algorithm	7
jpeg-encode	JPEG encoder	145
jpeg-decode	JPEG decoder	10
mpeg-decode	MPEG decoder	250
pegwit-keygen	Public key generation	27
pegwit-encode	Public key encryption	57
yacr2	VLSI router	70

For all measurements, a comparison is made with respect to a traditional memory hierarchy that does not incorporate any power-aware mechanisms (Figure 1a). The memory delay that gets added due to the introduction of new schemes is measured using their corresponding access times. For instance the access time for traditional memory hierarchy (as seen from the L1 cache) is given by:

$$t_{TM} = h_{L1} \cdot t_{L1} + m_{L1} \cdot (h_{L2} \cdot t_{L2} + m_{L2} \cdot t_{MEM}) \qquad (1)$$

where, h_{L1} and h_{L2} are the hit ratios of L1 and L2 caches, m_{L1} and m_{L2} are the miss ratios of L1 and L2 caches, and t_{L1}, t_{L2} and t_{MEM} are the times taken to access L1, L2 caches and memory respectively. The model used for evaluating power is a customized version of the Wattch [8] model. This architectural model has been incorporated within the existing framework of SimpleScalar. For our work, we developed an additional analytical framework to evaluate the power consumed by our proposed schemes. We measure the energy and access times for each of our selected benchmarks. We also evaluate the energy-delay product of the entire memory hierarchy for each of the proposed schemes. From these values, the relative performances of our new schemes are measured using simple ratios, with the energy consumed by additional buffers being calculated explicitly. The power consumed by the additional circuit, namely the extra logic gates that would be needed for our memory hierarchy implementations, works out to be a constant, and was found to be an insignificant factor when compared to the overall power values. For similar reasons, we also ignored the effect of leakage power. If the ratio of leakage power to the total power dissipation is very high, it would affect our memory hierarchy modifications. In that case, techniques to reduce leakage power must be applied in conjunction with all our proposed mechanisms.

3.2 Performance of Power-Aware Memory Scheme

We experimented with four variations of the PADRAM model, where we varied the threshold for memory state transitions (the number of instructions for which the memory remains in a particular energy state). Figure 2 depicts the energy ratio, memory delay ratio and energy-delay product ratio for each case when considering a 512MB memory having 16 modules each of size 32MB. Due to space limitations, all experimental results aren't presented here. *FFnap* and *FFstdby* denote the cases when the memory modules go directly to Nap and Stand-by states.

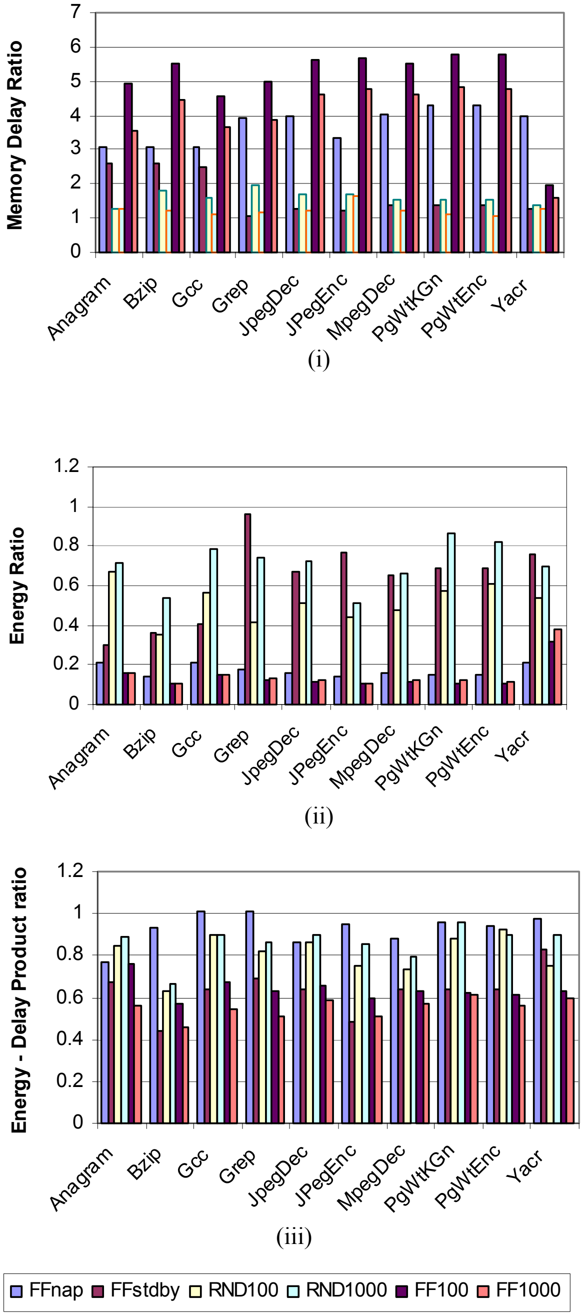


Figure 2. Memory delay (i), energy (ii) and energy-delay product ratios (iii) for a 512MB PADRAM scheme with sixteen 32MB modules. The base model is traditional cache and memory, where all modules are active. *FFnap* and *FFstdby* denote the cases when the memory modules go directly to nap and stand-by states. *FF100* and *FF1000* are the cases when sequential first touch policy (with 100 and 1000 instructions as the threshold for transition to the next power state) is used for replacement. *RND100* and *RND1000* denote the corresponding random page replacement policies.

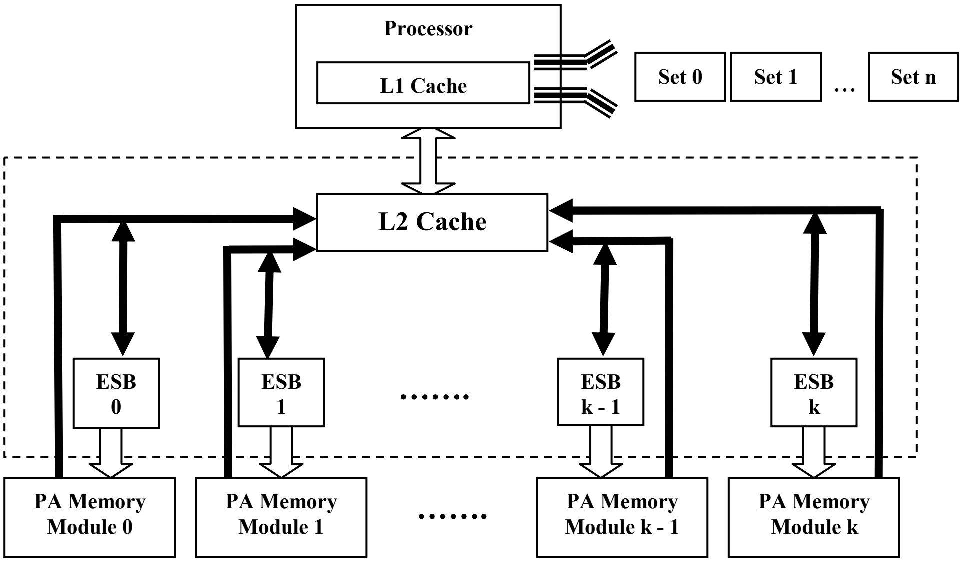


Figure 3. Schematic showing the addition of on-chip Energy-Saver Buffers to a power-aware memory hierarchy. Dotted line shows the chip boundary. Energy-Saver Buffers reside in the same chip as L2 cache. These buffers reduce the number of times the PA memory modules get reactivated.

FF100 is the sequential first touch scheme with 100 instructions as its threshold and *FF1000* is the scheme with 1000 instructions as its threshold. We use the PADRAM power model with the sequential first touch scheme for all the models that are used in subsequent sections, noting that the *FF1000* scheme performs the best in terms of energy-delay ratio on an average across all our selected benchmarks. The results are consistent with the conclusions arrived in the work done by Lebeck et al. in [21].

4. ON-CHIP ENERGY SAVER BUFFERS (ESB)

During a typical memory read/write operation, many memory modules of the power-aware memory get reactivated. Eventually, this would result in additional energy consumption and delay due to repeated resynchronization of the modules, which had gone back to a lower power state. Hence, to reduce the number of times the modules get reactivated, we propose onchip memory module buffers, called Energy-Saver Buffers (ESB), in between the L2 cache and the main memory. These buffers would be located on the same chip where the L2 cache resides (Figure 3). This technique would also reduce the dependence of activation of memory modules on the characteristics of programs since the traffic between L2 cache and each memory module still remains the same.

An Energy-Saver Buffer is associated with each memory module[1]. The ESB interacts with the memory module using the following protocol:

- If a memory read is initiated, before going to the memory module (which might be in a low-power state) the ESB of the corresponding module is searched for the word line. If the data is found, the delay and the power associated with the read operation are reduced since the ESB is located near the L2 cache.
- If the word-line is not found in the ESB, the corresponding memory module is activated for a read.
    - All the "dirty" lines in the ESB are first written back to the memory module.
    - All lines of the ESB are invalidated.
    - The data is read from the corresponding memory module to L2 cache.
- If a memory write is initiated and if the L2 cache is full, "dirty" data is written back to the ESB of the corresponding memory module and not to the memory module itself.
- If an ESB is full, the corresponding memory module is activated, the entire content of the ESB is flushed, and the ESB lines are invalidated.

From the results in Section 6, it will be shown that the overall energy consumption of a memory system with ESBs is significantly less than a memory configuration without any buffers, even after considering the additional energy consumed by ESBs. Moreover, the protocol ensures that data present in the ESB is the most recently updated copy. Thus the protocol ensures consistency and maximum utilization of the activation of memory modules, amortizing the overhead incurred due to resynchronization.

The ESBs are implemented as logically separated buffers within a single chip along with the L2 cache (owing to their small size – see Table 2). The cache controller is responsible for a harmonious

[1] A unified ESB for all memory modules (similar to the Victim Buffer proposed in [16]) was also considered for our experiments. As can be seen in Table 3, utilizing a Victim Buffer in place of an ESB increased the energy consumption significantly while demonstrating only slight performance improvements.

operation between L2 cache and ESBs since they co-exist in the same chip. The L2 cache has exclusive read/write access to ESBs. For read/write operations directed to the memory, the bus is accessed. Since L2 cache and ESBs are placed on the same chip, the bus and the arbitration logic remain the same. Thus existing hardware technologies can be used for implementing the new power-aware memory hierarchy and hence, addition of ESB is an economic solution. The modeling of ESBs for our experiments is explained in Section 6.

5. DYNAMIC CACHE

The proposed ESB can co-exist either with the traditional caches or in a system using any of the existing (or new) power-aware caching schemes. In this section, we illustrate this fact by establishing a scheme that allocates cache sets depending on varying needs of the program at a given instance. Note that similar approaches were examined in [1, 5]. The memory and hence the cache access patterns are carefully studied and then the number of active cache sets are dynamically increased or decreased (enabled or disabled) [28] according to the access patterns. Cache access patterns are similarly studied in [18, 19]. As proposed in the model by Se-Hyun et al. [28], the unused portion of the cache is provided with a gated V_{dd} to reduce the energy consumption. Figure 4 shows a flowchart describing the protocol for our dynamic cache operation.

Initially the program starts with a minimal number of sets, which we term as "active sets". The cache access pattern is then carefully followed. The Active cache Miss-cycle Counter (AMC) keeps track of the number of miss cycles for the current active sets. The Critical Miss-cycle Counter (CMC) keeps track of variations in the AMC. Depending on the value of the CMC, the cache is dynamically increased or decreased (more sets are enabled or some of the currently enabled active sets are disabled). For removing a currently active set, the cache access patterns for each individual set are studied and one is removed according to a Least-Recently Used (LRU) algorithm. In either case, the cache data and the tag bits undergo realignment [28].

The graphs in Figure 5 depict the memory delay, energy, energy-delay product ratios of the memory hierarchy comprising of dynamic cache and PADRAM without any ESB. The next section elaborates the experimental results obtained after integrating ESB as a part of the power-aware memory.

6. ANALYSIS OF INTEGRATED APPROACH

The ESBs are implemented in our experiments as multiple fully-associative caches. Hence the modeling of power and delay for ESB is the same as that for a fully associative cache. In our experiments we found that two different configurations of a direct-mapped L1 cache [15], one of size 128K with 32 sets and one of size 256K with 64 sets performed well in terms of energy-delay product when used in conjunction with a 512 MB PADRAM using the *FF1000* page replacement scheme (Table 3). Consequently these predominant configurations were employed as a basis for comparison when adding our ESBs, using combinations as shown in Table 2. Note that for all of our results explained in this paper, the only L2 cache configuration considered is a unified 2-way 1M cache.

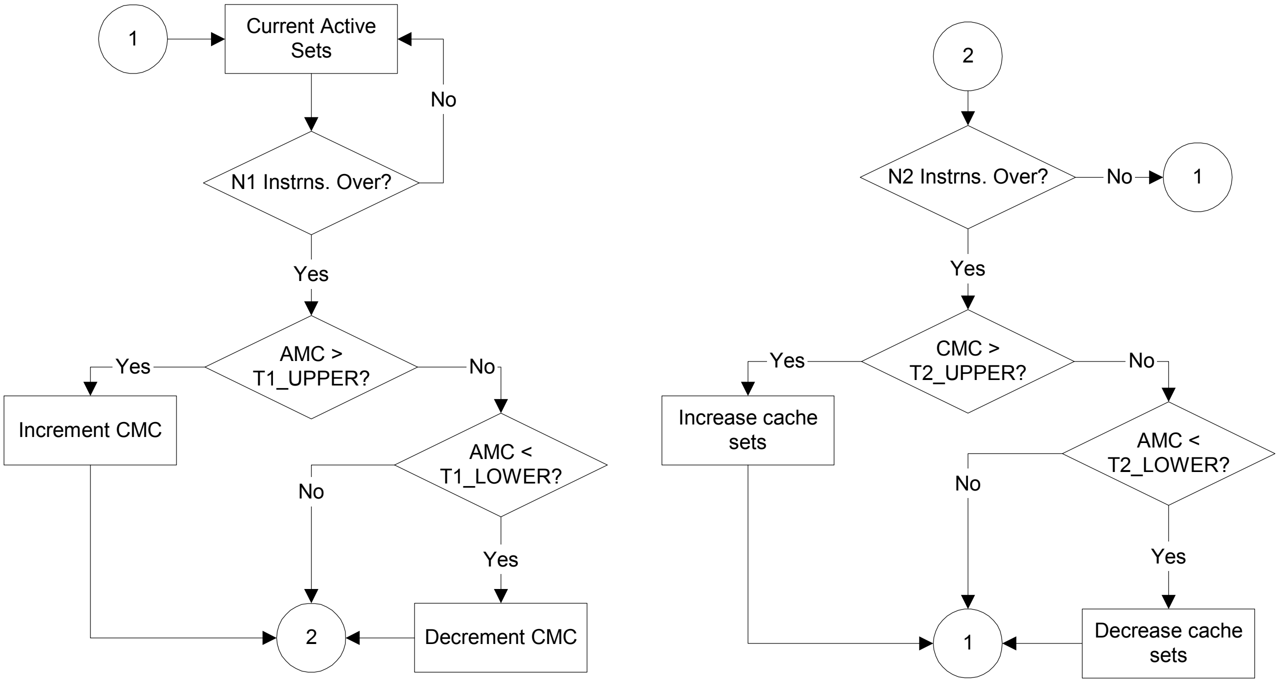


Figure 4. Flowchart showing the operation of dynamic cache. In dynamic caches, the caches resize periodically depending on the memory access pattern of a given application and hence the varying needs of the application. This is achieved by disabling unused portions of the cache.

Table 2. Combinations considered for Dynamic Cache, Energy-Saver buffer, PADRAM combination

Case	Dynamic L1 Cache Size	Total Energy-Saver buffer Size (# word blocks on each of 16 modules)
a	128K with 32 sets	4K (64)
b	256K with 64 sets	4K (64)
c	128K with 32 sets	8K (128)
d	256K with 64 sets	8K (128)

As can be seen from Figure 6, when adding the ESBs to our power-aware memory hierarchy, there is a large savings in terms of energy consumption that is offset by an increase in memory delay, when compared to the traditional memory hierarchy. This increased delay can be attributed to the dynamic caches, as when the number of cache sets used during a program's execution is reduced, there is an increased likelihood that data would be accessed directly from the memory. That is, in equation (1), the hit ratios h_{L1}, h_{L2} decrease. As can also be seen from Figure 6, the power consumption decreases significantly when utilizing dynamic caches since for long periods of time during program execution, many sets remain unused (experimental results also proved that the program itself did not require them). Consequently, for our integrated approach, the average energy consumption is 27% of the traditional model.

As is demonstrated in Figure 7, the addition of the ESB to the model without any buffers (a hierarchy with just dynamic cache and PADRAM) leads to an overall improvement in terms of energy consumption, memory delay, and energy-delay product. This result can be more deeply investigated by examining the access time equation for the memory model utilizing Energy-Saver Buffers:

$$t_{NEW} = h_{L1} \cdot t_{L1} + m_{L1} \cdot [h_{L2} \cdot t_{L2} + m_{L2} \cdot (h_{ESB} \cdot t_{ESB} + m_{ESB} \cdot t_{MEM})] \quad (2)$$

where h_{ESB}, m_{ESB}, t_{ESB} are the hit ratio, miss ratio and access time of ESBs.

The results showed that the hit ratio (h_{ESB}) is substantial. The hits increased as most of data being sought was already present in ESB. When the size of ESB was increased, the memory delay decreased further by as much as 13% (of the model with just dynamic cache and PADRAM without any buffers). This implies that an optimal value of the buffer size has the potential to reduce the delay significantly. Also, the average energy consumption reduces by 25% (with respect to the model with just the dynamic cache and PADRAM without any buffers). Consequently, the average energy-delay product also decreases by 26% for cases (a) and (b), while 36% for cases (c) and (d).

7. CONCLUSION

Given the large number of experiments and design alternatives presented (and other experiments not presented here due to space limitation), we discuss the overall results in a concise manner in this section. Table 3 summarizes our results.

Recalling the results derived in the earlier sections, PADRAM performs better than traditional memory. A hierarchy with dynamic cache increases the energy-delay product by 50%.

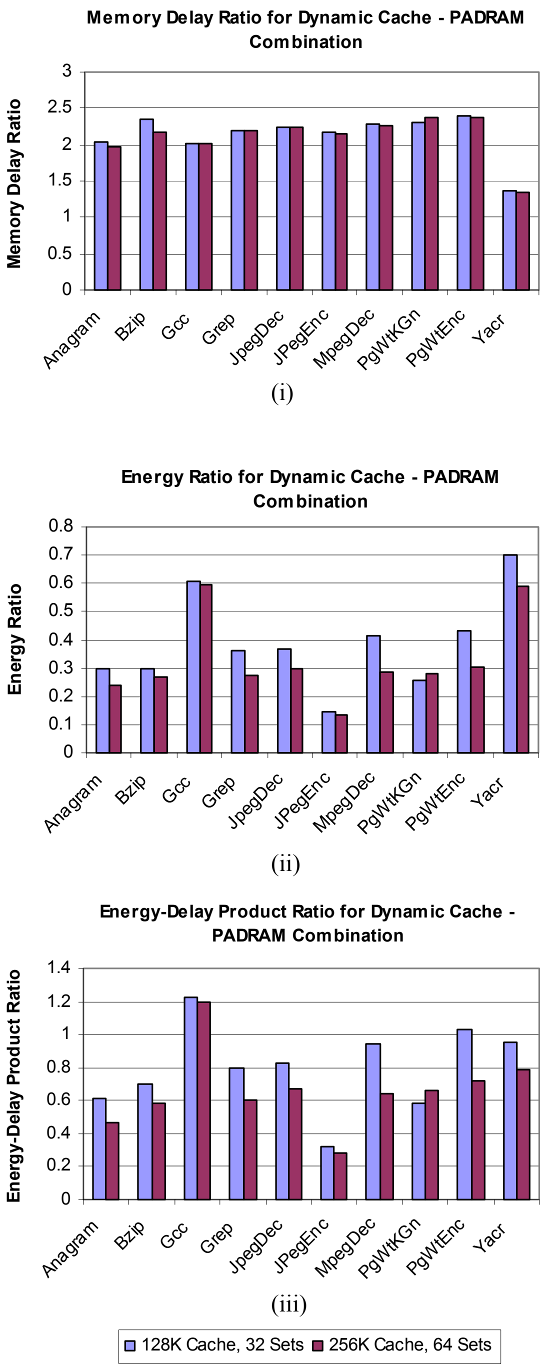
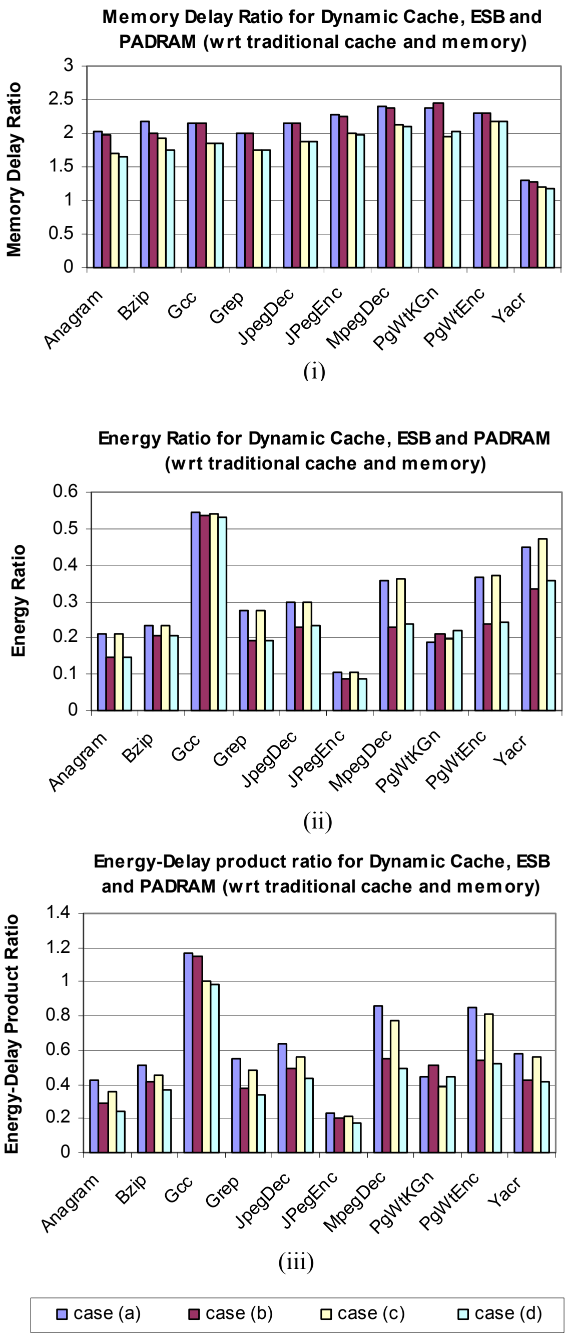


Figure 5. Memory delay ratio (i), energy ratio (ii) and energy-delay product ratio (iii) for dynamic cache and PADRAM combination (no ESB). The base model is traditional cache and memory without any buffers. The AMC and CMC thresholds (lower, upper) are (20, 50) and (4, 8). The instruction threshold (N1, N2) is (100, 1000). The average energy-delay product is 73% of traditional memory hierarchy. The energy consumed is just 35% of the traditional hierarchy.


Memory Delay Ratio for Dynamic Cache, ESB and PADRAM (wrt traditional cache and memory)

(i)

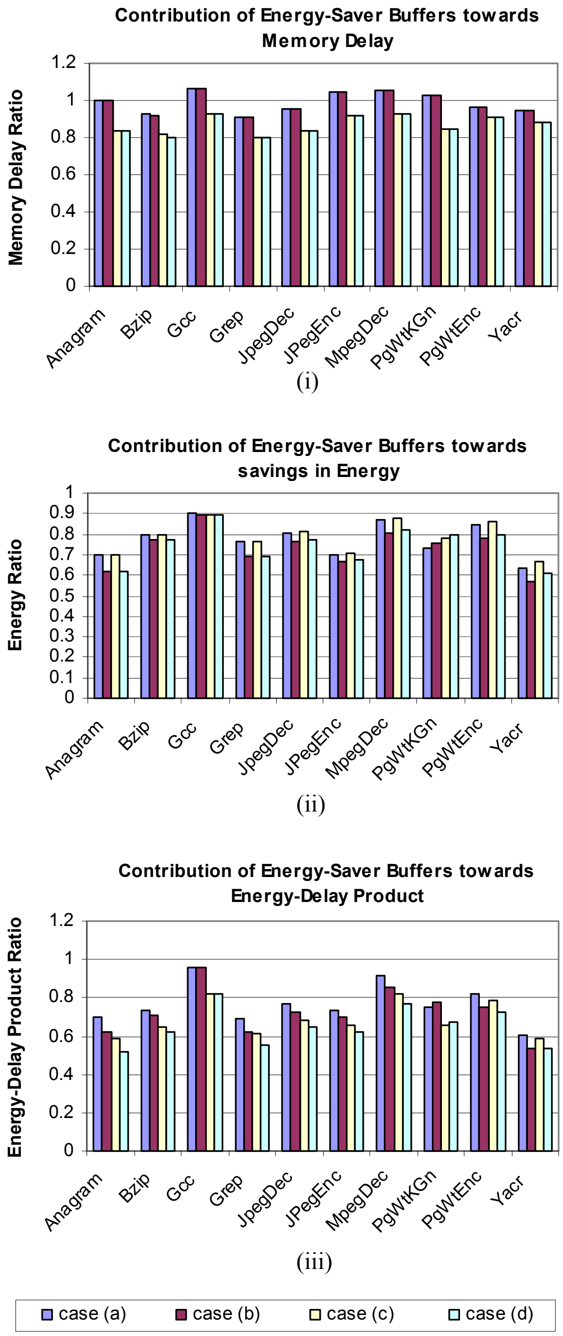
Energy Ratio for Dynamic Cache, ESB and PADRAM (wrt traditional cache and memory)

(ii)

Energy-Delay product ratio for Dynamic Cache, ESB and PADRAM (wrt traditional cache and memory)

(iii)

Figure 6. Memory delay ratio (i), energy ratio (ii) and energy-delay product ratio (iii) for dynamic cache, ESB and PADRAM Combination. The base model is traditional cache and memory without any buffers. The average energy consumption decreases by 70% of the traditional model. The average energy-delay product is 55%. The results show that the model with Energy-Saver Buffers performs better than the model with just PADRAM and dynamic cache (Figure 5).


Contribution of Energy-Saver Buffers towards Memory Delay

(i)

Contribution of Energy-Saver Buffers towards savings in Energy

(ii)

Contribution of Energy-Saver Buffers towards Energy-Delay Product

(iii)

Figure 7. The contribution of Energy-Saver Buffers towards the overall performance. The base model is dynamic cache and PADRAM combination without any intermediate buffers. ESBs bring down the memory delay by 13% of traditional hierarchies on an average (i). The addition of Energy-Saver Buffers has decreased the energy consumption by 25% further, when compared to the model without it (ii). The energy-delay product also reduces considerably (iii).

When using both dynamic cache and PADRAM in the hierarchy, the energy-delay product dropped by 30% of the case without them. On-chip ESBs were then added in-between dynamic cache and PADRAM to further improve the energy-delay product. A hierarchy with dynamic cache, ESBs and PADRAM established a 50% cutback in the energy-delay product, when compared to a traditional hierarchy.

From the results shown in Table 3, it is clear that the combination of a traditional cache, Energy-Saver Buffers, and the PADRAM memory (denoted by (2) in the table) performs the best in terms of energy-delay product, on average bearing a savings of 80% when compared to traditional memory hierarchy. This is at the cost of a three-fold increase in memory delay, which might drastically slow down the whole system. A better tradeoff between the memory delay and energy consumption is achieved with the introduction of dynamic cache. In Table 3, it can be seen that a more complex power-aware memory hierarchy combination (shown as (1) in the table) provides a substantial improvement in the delay, while still maintaining low energy consumption ratio. Even though the energy-delay product has increased, there is a substantial drop in the delay. This shows that the combination of dynamic cache, ESB, and PADRAM performs better than the combination of traditional cache, ESB, and PADRAM by providing an optimal tradeoff between energy and delay.

The following inferences can be derived from our experimental results:

- When two power-aware schemes are merged, the resulting scheme need not necessarily be power-aware. In our case, it is quite evident from Table 3 that on combining (TC,-, PM) and (DC,-, TM), the resulting scheme (DC,-, PM) does not perform well.
- While designing a power-aware scheme for any given layer in the hierarchy, the power schemes of both the immediate layers (upper and lower) should also be considered. For instance, cache schemes should be aware of the power schemes of the memory. Any buffer in between the cache and the memory should be aware of the power schemes of both the cache and the memory.
- Customization is necessary. For example, Victim Buffers, proposed in [16] and elsewhere, are known to improve performance. In our case, adding a Victim Buffer in-between dynamic cache and PADRAM (aiming to improve performance) ultimately results in an undesirable behavior (as seen in the [*, VB, *] combinations in Table 3). Hence a customization, similar to ESB, is necessary.
- Achieving a tradeoff between performance and power is indispensable in efficient memory hierarchy design. As we have already shown, certain memory schemes that are extremely power-aware can lead to considerably worse performance due to increased component delays. Achieving a superior solution often requires a complete enumeration of valid configurations.

In conclusion, an integrated approach to reducing memory hierarchy's power consumption leads to significant savings in terms of energy-delay product. Due in part to the efficiency of our proposed Energy-Saver Buffers, our power-aware memory hierarchy designed through such an effort reduces the energy delay product by 50% when compared to a traditional non power-aware memory hierarchy.

8. ACKNOWLEDGMENTS

This work was supported by DARPA under the contract F33615-01-C-1631. We thank Todd Austin (University of Michigan) for providing the SimpleScalar/Arm toolkit to Northwestern University. Our thanks to the other members of the Center for Parallel and Distributed Computing at Northwestern University for their invaluable help in refining this paper.

9. REFERENCES

[1] D.H. Albonesi. Selective Cache Ways: On-Demand Cache Resource Allocation. Journal of Instruction-Level Parallelism, Vol. 2, 2000.

[2] T. M. Austin and D. Burger. Micro-30 SimpleScalar Tutorial. Available HTTP: http://www.cs.wisc.edu/~mscalar/simplescalar.html

[3] R. I. Bahar, G. Albera, and S. Manne. Power and Performance Tradeoffs Using Various Caching Strategies, in Proceedings of the International Symposium on Low Power Electronics and Design (Monterey CA, 1998), ACM Press, pp. 64-69.

[4] R. Balasubramonian, D. Albonesi, A. Buyuktosunoglu, and S. Dwarkadas. Memory Hierarchy Reconfiguration for Energy and Performance in General Purpose Processor Architectures, in Proceedings of the International Symposium on Microarchitecture (Monterey CA, 2000), ACM Press, pp. 245-257.

[5] N. Bellas, I. Haij, and C. Polychronopoulos. Using dynamic cache management techniques to reduce energy in a high-performance processor, in Proceedings of the International Symposium on Low Power Electronics and Design (San Diego CA, 1999), ACM Press, pp. 64-69.

[6] L. Benini and G. De Micheli. System-Level Power Optimization: Techniques and Tools. ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 5, Issue 2 (April 2000).

[7] S. Borkar. Design challenges of technology scaling. IEEE Micro, Vol. 19, No. 4 (July/August 1999).

[8] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architecture-Level Power Analysis and Optimizations, in Proceedings of the International Symposium on Computer Architecture (Vancouver, Canada, 2000), ACM Press, pp. 83-94.

[9] D. Burger, T. M. Austin, and S. Bennett. Evaluating future microprocessors: the SimpleScalar tool set. Technical Report TR-1308, Univ. of Wisconsin-Madison Computer Sciences Dept., July 1996.

[10] R. Chen, M. Irwin, and R. Bajwa. Architectural Level Power Estimation and Design Experiments. ACM Transactions on Design Automation of Electronic Systems (TODAES), Volume 6, Issue 1 (January 2001).

Table 3. Summary of all viable combinations considered for the experiment

Legend

TC – Traditional Cache, DC – Dynamic Cache, EB – Energy-Saver Buffer, TM - Traditional Memory, PM – PADRAM, VB – Victim Buffer. DC, EB, PM means all of DC, EB and PM were considered for the case. '-' means that the particular component in the hierarchy wasn't considered for the test. For example, DC, -, PM means there was no EB or VB. ✓ means that the component has been considered for that particular case. Blank means the component wasn't considered.

** L2 cache size : 2-way set-associative 1M cache*

Case Considered (in order of memory hierarchy)	Dynamic L1 Cache		Victim Buffer (8K)	Energy Saver Buffer		PADRAM (FF1000, 512K)	Memory Delay Ratio	Energy Ratio	Energy-Delay product ratio	
	128K, 32 Sets	256K, 64 Sets		4K	8K					
TC, - , TM							1	1	1	
TC, - , PM						✓	4.06	0.15	0.55	
TC, EB , PM				✓		✓	3.64	0.06	0.20	
					✓	✓	**3.62**	**0.05**	**0.20**	2
DC, - , TM	✓						10.10	0.23	2.02	
		✓					10.59	0.17	1.54	
DC, - , PM	✓					✓	2.13	0.38	0.79	
		✓				✓	2.10	0.32	0.66	
DC, EB , PM	✓			✓		✓	2.11	0.30	0.62	
		✓		✓		✓	2.08	0.24	0.49	
	✓				✓	✓	1.85	0.30	0.56	
		✓			✓	✓	**1.83**	**0.24**	**0.44**	1
TC, VB , TM			✓				1.00	1.00	1.01	
DC, VB , TM	✓		✓				8.95	0.24	1.82	
		✓	✓				8.88	0.25	1.34	
DC, VB , PM	✓		✓			✓	2.21	0.39	0.84	
		✓	✓			✓	2.19	0.32	0.69	
DC, VB , EB, PM	✓		✓	✓		✓	2.19	0.31	0.67	
		✓	✓	✓		✓	2.17	0.32	0.64	
	✓		✓		✓	✓	1.93	0.31	0.60	
		✓	✓		✓	✓	1.91	0.33	0.58	

[11] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam. M. J. Irwin. DRAM Energy Management Using Software and Hardware Directed Power Mode Control, in IEEE Proceedings of the International Symposium on High Performance Computer Architecture (Nuevo Leone, Mexico, 2001), pp. 159-169.

[12] 128/144-Mbit Direct RDRAM Data Sheet, Rambus Inc., 1999.

[13] X. Fan, C. S. Ellis, and A. R. Lebeck. Memory Controller Policies for DRAM Power Management, in Proceedings of the International Symposium on Low Power Electronics and Design (Huntington Beach CA, 2001), ACM Press, pp. 129-134.

[14] K. Ghose and M. B. Kamble. Reducing Power in Super Scalar Processor Caches using Sub-banking, Multiple Line Buffers and Bit Line Segmentation, in Proceedings of the International Symposium on Low Power Electronics and Design (San Diego CA, 1999), ACM Press, pp. 70-75.

[15] J.L. Hennessey and D.A. Patterson. Computer Architecture – A Quantitative Approach. Morgan Kaufmann, 2000.

[16] N.P. Jouppi. Improving direct-mapped cache performance by addition of a small fully associative cache and pre-fetch buffers, in Proceedings of the International Symposium on Computer Architecture (Seattle WA, 1990), ACM Press, pp. 364-373.

[17] M. B. Kamble and K. Ghose. Analytical Energy Dissipation Models for Low Power Caches, in Proceedings of the International Symposium on Low-Power Electronics and Design (Monterey CA, 1997), ACM Press, pp. 143-148.

[18] M. Kandemir, U. Sezer, and V. Delaluz. Improving Memory Energy Using Access Pattern Classification, in Proceedings of the International Conference on Computer Aided Design (San Jose CA, 2001), ACM Press, pp. 201-206.

[19] S. Kaxiras and Z. Hu. Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power, in Proceedings of the International Symposium on Computer Architecture (Göteborg, Sweden, 2001), ACM Press, pp. 240 - 251.

[20] S. Kim, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. Energy-efficient instruction cache using page-based placement, in Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (Atlanta GA, 2001), ACM Press, pp. 229 - 237.

[21] A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis. Power Aware Page Allocation, in Proceedings of the International Conference on Architectural Support for Programming Language and Operation Systems (Cambridge MA, 2000), ACM Press, pp. 105 - 116.

[22] K. Li, R. Kumpf, P. Horton, and T. Anderson. A Quantitive Analysis of Disk Drive Power Management in Portable Computers. Winter Usenix, 1994, pp. 279-291.

[23] Pentium III Processor Mobile Module MMC-2, Datasheet 243356-001, Intel Corporation, 2001.

[24] RDRAM® Technology, Rambus Inc., 1999. Available HTTP: http://www.rdram.com/

[25] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, and R. Patel. Reducing Power in high-performance microprocessors, in Proceedings of the Design Automation Conference (San Francisco CA, 1998), ACM Press, pp. 732 - 737.

[26] C-L. Su and Despain. Cache Design Tradeoffs for Power and Performance Optimization: A Case Study, in Proceedings of the International Symposium on Low-Power Electronics and Design (Dana Point CA, 1995), ACM Press, pp. 63-68.

[27] T. Sherwood, B. Calder, and J. Emer. Reducing Cache Misses Using Hardware and Software Page Placement, in Proceedings of the International Conference on Supercomputing (Rhodes, Greece, 1999), ACM Press, pp. 155 - 164.

[28] Se-Hyun 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 IEEE Proceedings of the International Symposium on High-Performance Computer Architecture (2001), pp. 147-158.

[29] H. Zhou, M. C. Toburen, E. Rotenberg, and T. M. Conte. Adaptive Mode Control: A Static-Power-Efficient Cache Design, in IEEE Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (Barcelona, Spain, 2001), pp. 61 -70.