# Energy/Performance Design of Memory Hierarchies for Processor-in-Memory Chips

**Michael Huang**[†], **Jose Renau**[†], **Seung-Moon Yoo** [‡], and **Josep Torrellas**[†]
Department of Computer Science[†]
Department of Electrical and Computer Engineering[‡]
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

## 1 Introduction

Merging processors and memory into a single chip has the well-known benefits of allowing high-bandwidth and low-latency communication between processor and memory, and reducing energy consumption. As a result, many different systems based on what has been called Processor In Memory (PIM) architectures have been proposed [14, 2, 6, 7, 9, 11, 12, 13, 15, 16, 18].

Recent advances in technology [3, 4] appear to make it possible to integrate logic that cycles nearly as fast as in a logic-only chip. As a result, processors are likely to put much pressure on the relatively slow on-chip DRAM. To handle the speed mismatch between processors and DRAM, these chips are likely to include non-trivial memory hierarchies in each DRAM bank.

With many on-chip high-frequency processors, all of them potentially accessing the memory system concurrently, these chips will consume much energy. In addition, these chips are likely to be used in non-traditional places like the memory of a server [2, 6, 11] or the I/O subsystem [14], which may not have heavy-duty cooling support. Consequently, it is important to design the chips for energy efficiency.

In this abstract, we examine, from a performance and energy-efficiency point of view, the design of the memory hierarchy in a multi-banked PIM chip with many simple, fast processors. Our results suggest the use of per-processor memory hierarchies that include modest-sized caches, simple DRAM bank organizations that support segmentation, and no prefetching.

## 2 Memory Hierarchies for PIM Chips

Our focus architecture is a PIM chip that includes tens of relatively simple, high-frequency processors, each of which is associated with a bank of DRAM. Such a design has been suggested for systems like Active Pages [11, 12], FlexRAM [6], and DIVA [2] among others. The chip can be modeled as in Figure 1-(a), where the organization of the processors, memory, and network may vary. We feel, however, that currently-proposed designs are relatively conservative in logic speed. Recent advances in technology appear to allow logic to cycle nearly as fast as in a logic-only chip [3, 4]. This means that these chips may soon include processors cycling at about 800-1000 MHz. Such processors are likely to put much pressure on the slower DRAM.

To handle the speed mismatch between processors and DRAM, these chips are likely to associate a non-trivial memory hierarchy to each DRAM bank. In this paper, we assume a per-bank baseline memory hierarchy as in Figure 1-(b). In the figure, the instruction memory hierarchy includes a fast SRAM memory. The data memory hierarchy includes a cache with hardware sequential prefetch of 1 line. The DRAM bank itself is sub-banked and has row and data buffers. For example, Figure 1-(c) shows the DRAM organized into 8 sub-banks, with 10 row buffers, and 2 256-bit data buffers.

Unlike in memory-only chips, where the DRAM organization is often limited to standard designs, embedded systems allow many different organizations for the DRAM array. For example, designers can change the width and length of a DRAM sub-bank, and the number of sub-banks. These changes can affect the performance delivered and the energy consumed by DRAM accesses, and the area utilized.

In a traditional DRAM array organization, when a bank is accessed, every other sub-bank is activated. Consecutive sub-banks are not activated because they share a row buffer. Figure 2-(a) shows a 4 sub-bank organization. We now consider three improvements: segmentation, interleaving, and pipelining.

With segmentation (Figure 2-(b)), only one sub-bank is activated at a time. The resulting row buffer decoupling changes the hit rate of the row buffers. In addition, DRAM accesses consume less energy: because only half of the bit lines are activated, about 50% of the energy is saved.

With interleaving, each sub-bank is vertically sliced and a data bus is assigned to each of the resulting slices. Figure 2-(c) shows a 2-way interleaved system. The performance is higher because both data busses work in parallel (Figure 2-(d) shows a timing diagram with the maximum overlap, assuming a single address bus). As for energy, although row buffer hits now cost a bit more, DRAM accesses again save about 50% of the energy because only half of the cells are activated. The area used increases.

Finally, one problem shown in Figure 2-(d) is that reads from different sub-banks that share a data bus are serialized by long sub-bank occupancy times. With pipelining, these sub-banks can overlap their occupancy times (Figure 2-(e)). The only serialization happens in the shared address bus and data bus. The result is higher performance. As for energy, pipelining has only a small impact.

## 3 Evaluation Environment

We evaluate the PIM chip of Section 2 using a MINT-based simulation system [8]. The architecture modeled is a single chip with 64 processors connected in a ring. Each processor is associated with a 1-Mbyte DRAM bank like in Figure 1-(b). The baseline parameters of each processor-bank pair are shown in Table 1. The target technology is IBM's 0.18 $\mu$m Blue Logic SA-27E ASIC [3] with the default voltage of 1.8 V.

The names for the DRAM bank organizations that we evaluate are *Trad*, *S*, *SP*, *IS*, and *ISP*, which refer to traditional,

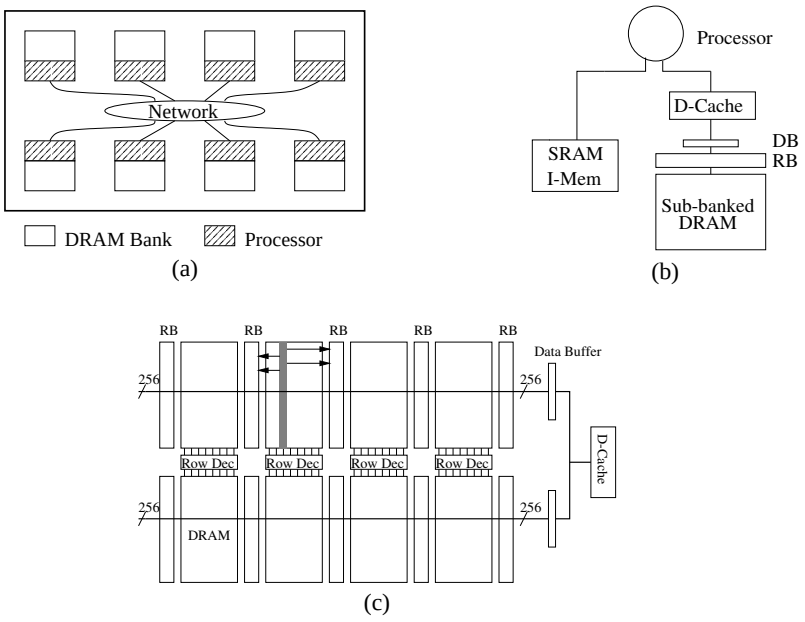


Figure 1: Example of chip architecture considered. *RB*, *DB*, and *Row Dec* stand for row buffer, data buffer and row decoder, respectively.

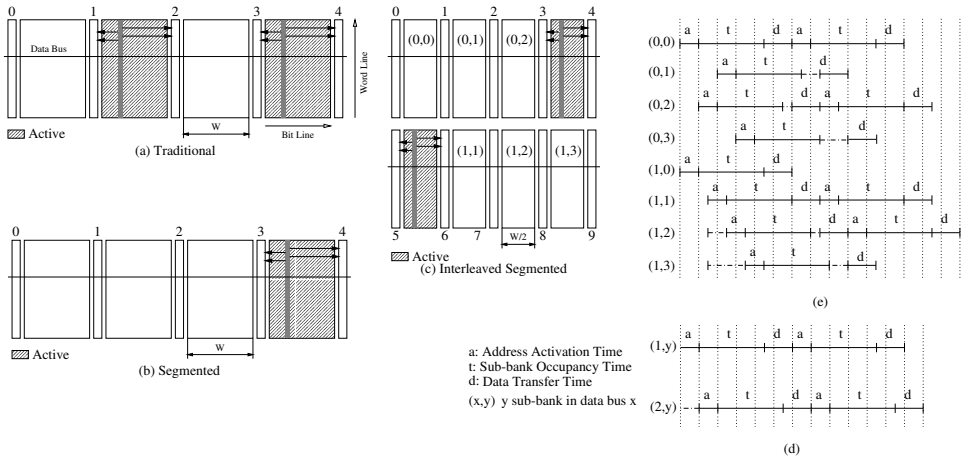


Figure 2: Different DRAM bank organizations and timings.

segmented, segmented pipelined, interleaved segmented, and interleaved segmented pipelined, respectively. In each case, we add $(i, j)$ to refer to $i$-ways interleaved with $j$ sub-banks per way.

To estimate the energy consumed in the chip, we have applied scaling-down theory to data on existing devices reported in the literature, as well as used several techniques and formulas reported in the literature [5, 17, 19, 20]. We add the contributions of the processors, clock, memory hierarchies, and other modules. A detailed discussion of the methods that we have followed can be found in [21]. In [21], we have additionally validated our estimates with CACTI [19] and with published results on the ARM processor [10].

For the experiments, we use 6 applications that are suitable to the integer-based PIM chip considered: they access a large memory size, are very parallel, and are integer based. They come from several industrial sources. We have parallelized each application into 64 threads by hand.

Table 2 lists the applications and their characteristics. They include the domains of data mining, neural networks, protein matching, multimedia, and image compression. Each application runs for several billions of instructions.

# 4 Evaluation

labelevaluation

The best memory hierarchy organization depends on the metric being optimized. We consider two metrics: performance and energy-delay product. In our evaluation, we start with the baseline architecture of Section 3 and then vary it. As a reference, we use an ideal architecture (*Perf*): loads and stores are satisfied with zero latency and consume no energy in the memory system.

### Maximizing Performance

To compare performance, we measure the average IPC delivered by the combined 64 processors for the duration of the application. We first evaluate the effect of the memory bank organization. Figure 3 shows the IPC of the applications running on the baseline architecture for different memory bank

| Processor | D-Cache | I-Memory | Data Buffer | Row Buffer | Sub-Bank |
|---|---|---|---|---|---|
| 2-issue in-order 800MHz | Sz: 8KB, WB | Size: 4 Kinst. | Number: 1 | Number: 5 | Number: 4 |
| BR Penalty: 2 cycles | Assoc: 2 | Line: 4 inst. | Size: 256 b | Size: 1 KB | Cols: 4096 |
| Int,Ld/St,FP Units: 2,1,0 | Line: 32 B | RTrip:1.25ns | Bus: 256 b | Bus: 256 b | Rows: 512 |
| Pending Ld,St: 2,2 | RTrip:1.25ns | | RTrip:3.75ns | RTrip:7.5ns | RTrip:15 ns |

Table 1: Parameters for a single memory bank and processor pair. In the table, *BR* and *RTrip* stand for branch and contention-free round-trip latency from the processor, respectively.

| Appl. | What It Does | Problem Size | D-Cache Hit Rate | Average Power(W) |
|---|---|---|---|---|
| *GTree* | Data mining: tree generation | 5 MB database, 77.9 K records, 29 attributes/record | 50.7% | 10.2 |
| *DTree* | Data mining: tree deployment | 1.5 MB database, 17.4 K records, 29 attributes/record | 98.6% | 10.8 |
| *BSOM* | BSOM neural network | 2 K entries, 104 dims, 2 iters, 16-node network, 832 KB db | 94.7% | 15.5 |
| *BLAST* | BLAST protein matching | 12.3 K sequences, 4.1 MB total, 1 query of 317 bytes | 96.9% | 8.7 |
| *Mpeg* | MPEG-2 motion estimation | 1 1024x256 frame plus a reference frame. Total 512 KB | 99.9% | 11.3 |
| *FIC* | Fractal image compressor | 1 512x512 image, 4 512x512 internal structure. Total 2 MB | 97.8% | 6.1 |

Table 2: Applications executed.

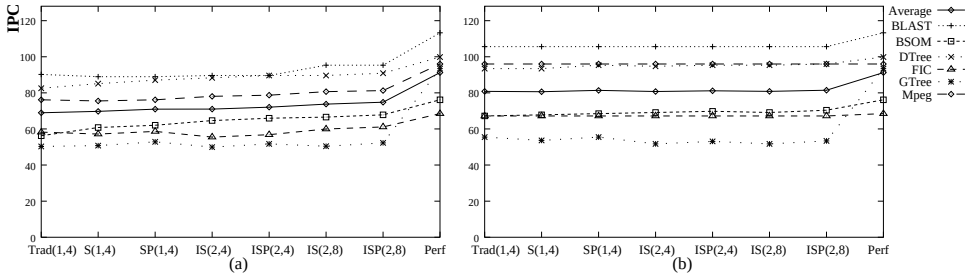


Figure 3: Effect of the DRAM bank organization on the IPC in systems with 1-Kbyte (a) and 8-Kbyte (b) data caches.

organizations. Charts (a) and (b) correspond to systems with 1- and 8-Kbyte D-caches, respectively. The memory organizations are ordered from the simpler ones on the left side to the more sophisticated ones on the right side. Each chart has an *Average* line that tracks the average of all applications.

Figure 3-(a) shows that performance improves slightly as we move to the more sophisticated designs. Going from *Trad(1,4)* to *ISP(2,8)* increases the IPC by an average of 8%. However, for 8-Kbyte caches (Figure 3-(b)), the changes are very small. This is because, with large caches, there are relatively few cache misses and, as a result, the type of DRAM bank organization matters less.

Comparing the IPC in *Perf* and *ISP(2,8)*, we see the IPC lost in the most advanced memory system. This fraction is on average 18% and 11% in Figures 3-(a) and (b).

Figure 5-(a) shows the effect of the cache size and prefetching support. We consider the baseline architecture with three different DRAM bank organizations: conservative (*Trad(1,4)*), aggressive (*ISP(2,8)*), and in-between (*IS(2,4)*). The figure shows the IPC averaged over all applications. We analyze caches of 256 bytes, 1 Kbyte, 8 Kbytes, and 16 Kbytes, all with and without prefetching. For each memory organization, there are 8 bars, labeled with the cache size in bytes followed by *P* or *NP* for prefetching or not prefetching, respectively.

The best performance is achieved with the largest cache size (16 Kbytes). However, large caches deliver diminishing returns. The figure also shows that adding the simple prefetching support considered here makes little difference to performance.

**Minimizing the Energy-Delay Product**

In embedded systems, a common figure of merit is the energy-delay product [1]. A low product implies that the system is both fast and energy-efficient. Consequently, in this section, we compare the energy-delay product of the chips with different memory hierarchy designs. To compute the energy consumed, we add up the contributions of all the subsystems in the chip.

Figures 4-(a) and 4-(b) show the energy-delay product of the chip under the baseline architecture for different DRAM bank organizations. Charts (a) and (b) correspond to systems with 1- and 8-Kbyte D-caches respectively, and are organized as in Figures 3-(a) and 3-(b). For each application, the charts are normalized to *Perf*.

In systems with 1-Kbyte caches (Figure 4-(a)), the average energy-delay product decreases for the more advanced memory organizations. For example, the product in *ISP(2,8)* is only 60% of that in *Trad(1,4)*. The reason is that advanced DRAM bank organizations deliver slightly higher IPCs and consume much less energy in the process. However, as caches increase to 8 Kbytes (Figure 4-(b)), the changes are smaller. Overall, for 8-Kbyte cache systems, only segmentation (going from *Trad(1,4)* to *S(1,4)*) makes a significant difference. Supporting interleaving and increasing the number of sub-banks from (2,4) to (2,8) has only a small effect.

Figure 5-(b) measures the energy-delay product for the average of all applications for different cache sizes and prefetching support. The bars are normalized to *Perf*. From the figure, we see that designs with larger caches tend to have lower energy-delay products. For example, in *Trad(1,4)*, the product with 16-Kbyte caches is about 30% of the product with 256-byte caches. The reason is that caches have a double effect: they speed up the program and, in addition, eliminate energy-consuming memory accesses. We observe, however, that for the more advanced memory organizations and

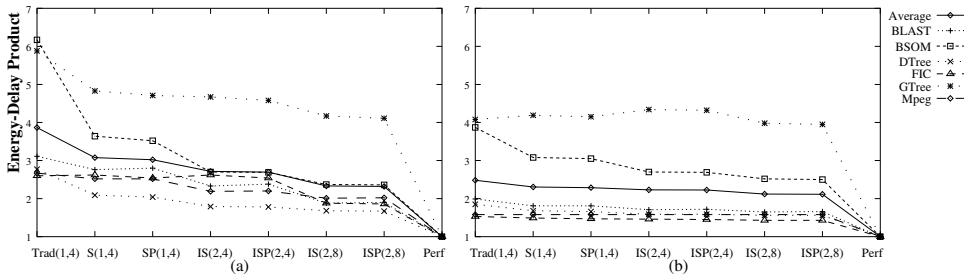


Figure 4: Effect of the DRAM bank organization on the energy-delay product in systems with 1-Kbyte (a) and 8-Kbyte (b) data caches.

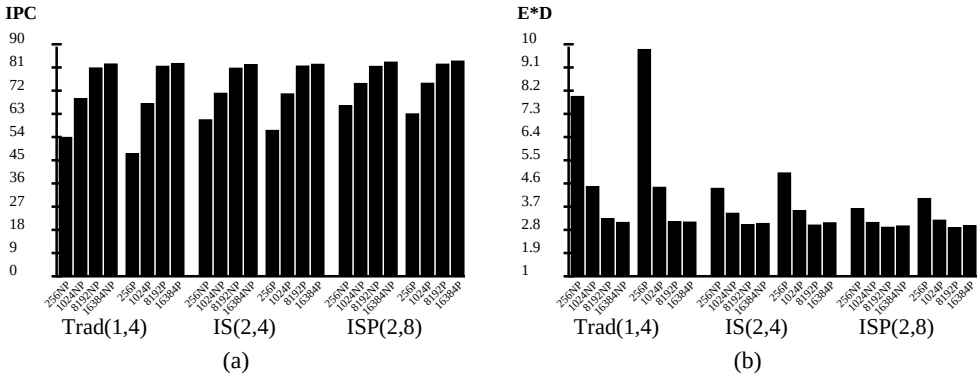


Figure 5: Effect of the cache size and prefetching support on IPC (a) and energy-delay product (b).

large caches, the trend reverses: 16-Kbyte caches are slightly worse than 8-Kbyte caches. The reason is that the diminishing returns in lower miss rates delivered by larger caches do not compensate for the higher energy consumption that larger caches require. We also see that simple prefetching does not help.

## 5 Discussion

In a PIM chip like the one analyzed here, minimizing the energy-delay product is likely to be the top priority. Our results suggest to use modest-sized D-caches (8 Kbytes), a simple DRAM bank organization that supports only segmentation, and no prefetching. Modest-sized caches are effective: they speed-up the application, are energy-efficient, consume modest area, and render fancy DRAM bank organizations largely unnecessary. If area is not an issue, the energy-delay product can be improved slightly by supporting interleaving in the DRAM bank and increasing the number of sub-banks.

## REFERENCES


[1] R. Gonzalez and M. Horowitz. Energy Dissipation In General Purpose Microprocessors. *IEEE Journal on Solid-State Circuits*, 31(4):1277–1284, September 1996.

[2] M. Hall et al. Mapping Irregular Aplications to DIVA, a PIM-based Data-Intensive Architecture. In *Supercomputing*, November 1999.

[3] IBM Microelectronics. Blue Logic SA-27E ASIC. http://www.chips.ibm.com/news/1999/sa27e, February 1999.

[4] S. Iyer and H. Kalter. Embedded DRAM technology: opportunities and challenges. *IEEE Spectrum*, April 1999.

[5] M. Kamble and K. Ghose. Analytical Energy Dissipation Models for Low Power Caches. In *International Symposium on Low Power Electronics and Design*, pages 143–148, August 1997.

[6] Y. Kang, W. Huang, S. Yoo, D. Keen, Z. Ge, V. Lam, P. Pattnaik, and J. Torrellas. FlexRAM: Toward an Advanced Intelligent Memory System. In *International Conference on Computer Design*, pages 192–201, October 1999.

[7] P. Kogge, S. Bass, J. Brockman, D. Chen, and E. Sha. Pursuing a Petaflop: Point Designs for 100 TF Computers Using PIM Technologies. In *Frontiers of Massively Parallel Computation Symposium*, 1996.

[8] V. Krishnan and J. Torrellas. An Execution-Driven Framework for Fast and Accurate Simulation of Superscalar Processors. In *International Conference on Parallel Architectures and Compilation Techniques*, pages 286–293, October 1998.

[9] K. Mai et al. Smart Memories: A Modular Reconfigurable Architecture. In *International Symposium on Computer Architecture*, 2000.

[10] J. Montanaro et al. A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor. *IEEE Journal Solid State Circuits*, 31(11):1703–1714, November 1996.

[11] M. Oskin, F. Chong, and T. Sherwood. Active Pages: A Computation Model for Intelligent Memory. In *International Symposium on Computer Architecture*, pages 192–203, June 1998.

[12] M. Oskin et al. Exploiting ILP in Page-Based Intelligent Memory. In *International Symposium on Microarchitecture*, 1999.

[13] D. Patterson et al. A Case for Intelligent DRAM. *IEEE Micro*, pages 33–44, March/April 1997.

[14] D. Patterson et al. ISTORE: Intelligent Store. http://iram.cs.berkeley.edu/ istore/index.html, 1998.

[15] D. Patterson and M. Smith. Workshop on Mixing Logic and DRAM: Chips that Compute and Remember. 1997.

[16] S. Rixner et al. A Bandwidth-Efficient Architecture for Media Processing. In *International Symposium on Microarchitecture*, November 1998.

[17] C-L. Su and A. Despain. Cache Design Trade-offs for Power and Performance Optimization: A Case Study. In *International Symposium on Low Power Electronics and Design*, pages 63–68, April 1995.

[18] E. Waingold et al. Baring It All to Software: Raw Machines. *IEEE Computer*, pages 86–93, September 1997.

[19] S. Wilton and N. Jouppi. CACTI: An Enhanced Cache Access and Cycle Time Model. *IEEE Journal on Solid-State Circuits*, 31(5):677–688, May 1996.

[20] N. Yeung. et al. The Design of a 55SPECint92 RISC Processor under 2W. *ISSCC Digest of Technical Papers*, pages 206–207, February 1994.

[21] S-M. Yoo, J. Renau, M. Huang, and J. Torrellas. FlexRAM Architecture Design Parameters. Technical Report CSRD-1584, Department of Computer Science, University of Illinois at Urbana-Champaign, October 2000. http://iacoma.cs.uiuc.edu/flexram/publications.html.