Realizing the Full Potential of Heterogeneity through Processing in Memory

Nuwan Jayasena, Dong Ping Zhang, Amin Farmahini-Farahani, and Mike Ignatowski

AMD Research
Advanced Micro Devices, Inc.
{nuwan.jayasena, dongping.zhang, amin.farmahinifarahani, mike.ignatowski}@amd.com

Abstract

While many processing in memory (PIM) research studies demonstrate significant improvements in memory system energy efficiency, relatively little attention has been paid to the sources of overall energy efficiency of PIM systems. In this paper, we quantify the sources of energy efficiency of a GPU-based PIM design and show that selecting low-power operating points for the in-memory processors is an important aspect, accounting for a 1.9x improvement in energy efficiency compared to a mainstream implementation of the evaluated GPU design. Memory interface efficiency of PIM provides an additional 3.8x improvement over that. These results also demonstrate that, due to memory system inefficiencies, implementing high-performance and low-power heterogeneous cores on the same die attached to a conventional memory system can only realize a fraction of the overall improvement realized by PIM (52% in our study). While these results in general confirm conventional wisdom, we quantify the relative importance of these processor and memory efficiency factors across a wide range of benchmarks and encourage further research to enable and leverage the symbiosis between PIM and heterogeneous computing to further improve energy efficiency.

1. Introduction

Processing in memory (PIM) has been researched over a long period of time as a technique to address memory system bottlenecks (e.g., [12, 13, 17, 20, 21]). Many of these approaches required the integration of processing elements and memory (DRAM) on the same silicon die, hampering adoption in mainstream implementations. Recent PIM studies have used 3D die-stacking to incorporate processing and memory within a single die stack (e.g., [5, 11, 19, 23, 27]). This approach allows logic and memory dies to be fabricated in their respective process technologies before being incorporated into a 3D stack. Figure 1 shows an example organization incorporating such a 3D-stacked PIM implementation. The *host* is a traditional processor attached to a number of memory modules (four in this example). Each memory module consists of DRAM, which forms the main memory of the host, and a logic die under the memory dies, which contains the PIM devices.

Improvements in processor energy efficiency and emerging data-intensive applications cause an increasing fraction of system energy to be spent on moving data to and from main memory. PIM is an effective method to reduce this data movement energy. Consequently, energy efficiency evaluations of many PIM studies focus on the memory system. However, a variety of factors also drive PIM implementations to select energy-efficient processors and less aggressive operating points. For example, the computations that benefit the most from PIM are memory-intensive ones. Therefore, PIM cores need not be designed nor operated in a manner that maximizes arithmetic throughput. Further, thermal challenges are exacerbated for PIM as DRAM data retention times are reduced at higher temperatures, which also motivates lower PIM power consumption. The use of such energy-efficient processor cores also contribute significantly to the overall energy savings of PIM solutions relative to mainstream systems.

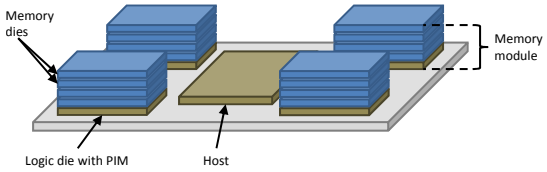

Figure 1: Example system organization with PIM.

Another important approach to improve energy efficiency is processor heterogeneity. Research and industry efforts on heterogeneity fall into two categories. In one, heterogeneity is based purely on performance and arises from high-performance and low-power variants of the same execution architecture. For example, "big" cores and "little" cores may be incorporated into the same die such that non-compute-intensive tasks may be executed on the power-efficient cores [18]. In the other, heterogeneity arises from the incorporation of accelerators along with general-purpose processor cores. Examples include systems with programmable accelerators specialized for certain classes of applications (e.g., GPGPU for data-parallel workloads), configurable accelerators (e.g., FPGAs), and domain-specific or fixed-function accelerators (e.g., image and video codecs). In both categories, the common characteristic is improved performance or energy efficiency relative to homogeneous processors through heterogeneity. However, as energy efficiency of the execution units improves, the energy consumed by conventional memory systems becomes more significant, limiting the overall benefit.

To understand the contributions of processor heterogeneity and memory interface to the energy efficiency of PIM, we compare three designs: a *host* configuration that models a

hypothetical, near-future, high-performance accelerated processing unit (APU) that combines CPU and GPU on the same die with a memory system based on high-speed serial links[1] similar to Hybrid Memory Cube (HMC) [22]; a *small (APU) core* configuration that implements less aggressive APU cores on the host die and utilizes the host's memory system; and a *PIM* configuration that implements the same compute units as the small cores in the memory modules. Comparing host and small core configurations, we show that heterogeneity in the execution units is an important source of energy efficiency for PIM. By comparing to the PIM configuration, we also show that heterogeneous cores with a conventional memory system result in limited benefits due to memory system energy consumption and bottlenecks, and that moving the energy-efficient computing units closer to memory enables much greater overall energy savings. Therefore, we encourage further research to better exploit the complementary nature of PIM and heterogeneous computing. We also identify a few such directions and highlight technology evolutions necessary to support them.

2. Sources of PIM Energy Efficiency

In order to quantify the energy efficiency contributions of processor heterogeneity and the memory system, we analyze the system and benchmarks evaluated by Zhang et al. [27] and extend the energy estimates of that work in two ways. First, we break down the energy consumption across key components of the system. Second, we introduce the small core configuration to better understand the benefits of processor and memory interface optimizations. The system organization is similar to Figure 1, consisting of a host and four memory modules, and the key parameters are summarized in Table 1. Host, small core, and PIM devices are APUs and the evaluation focuses on their GPU execution engines. The GPUs are based on AMD's Graphics Core Next architecture [6], the host design point was chosen by extrapolating GPU market trends, and the PIM design point was chosen based on area and thermal constraints [9, 27]. The PIM devices use a less aggressive operating point enabling the use of a low-power process and/or low-leakage devices. Note that the aggregate PIM compute throughput is 50% of that of the host. The benchmarks evaluated consist of kernels from subsets of the Mantevo [15] and Rodinia [8] suites and custom-developed graph processing algorithms. For PIM execution, data sets are partitioned and shared data structures are replicated to minimize communication among PIM devices as is likely to be the case in realistic, efficient PIM-based implementations.

Our evaluation uses the simulation methodology described by Zhang et al. [27], which gathers hardware performance counters during native execution on an existing GPU and uses machine-learning models to estimate performance and core

	Host	Small Core	PIM
Number of instances	1	1	4
GPU compute units per instance	64	48	12
GPU clock frequency (MHz)	1000	650	650
Compute throughput per instance (TFLOP/s)	4	2	0.5
Peak DRAM bandwidth (GB/s)	4×160	4×160	640
Technology node (nm)	16	16	16LP
TDP per instance (W)	170	40	10
Leakage (% of TDP)	22.5	5	5

Table 1: System configuration and parameters.

dynamic energy consumption of other design points. This approach has been shown to have accuracy comparable to cycle-level simulators [26, 27]. Cache energy consumption is included in core dynamic energy and is assumed to scale with the number of compute units. Processor static power is estimated assuming an aggressive implementation in a performance-optimized process for the host, low-leakage devices in the same process for small cores, and a power-optimized process and/or low-leakage devices for PIM. We conservatively assume the small core static power can be controlled to the same degree as the PIM devices, which in reality may not be possible as the small cores are implemented on the same die as the host, which may limit the range of process parameters available to reduce leakage (such as threshold voltages and oxide thicknesses) compared to a dedicated low-power process that can be used for PIM devices. Memory interface power is estimated for the SerDes and interface support logic within each memory module for communication with the host die. We assume this interface is similar to HMC and consumes 4.5pJ/b [16, 24] at 160GB/s per memory module. We also assume interface power consumption is constant regardless of utilization as idle symbols need to be transmitted when the interface would otherwise be idle to keep the links aligned and the recovered clocks locked [4, 7]. Power consumed by the memory interface on the processor die (in the case of host and small core) is included in the processor dynamic power estimate. We model DRAM dynamic power encompassing DRAM core access (4pJ/b [25]), expected TSV traversal (0.4pJ/b [1]), and expected wire traversal within the memory module[2] (0.8pJ/b [2]). DRAM static power corresponds to background and refresh power and is estimated at 10% of the maximum active power of the DRAM dies [23]. We assume unused compute units are completely power-gated (e.g., PIM and small cores are power-gated during host execution).

Figure 2 shows the performance of each of the evaluated kernels on PIM normalized to the execution time on the host. The execution times for the small cores match that of the host for memory-limited kernels and PIM for compute-limited kernels. In general, memory-intensive kernels show performance improvements on PIM (e.g., *RW_mem_init*) while compute-bound kernels (e.g., *CoMD_EAM3*) show performance degradations on PIM due to its reduced compute capabilities. Fur-

[1] While discrete GPUs are now available with in-package stacked memory, we do not anticipate the inclusion of all of system memory within the processor package for high-performance systems in the foreseeable future.

[2] We assume an organization similar to High Bandwidth Memory [3] within the memory module, where all memory channel TSVs are near the center of the 3D stack.

ther discussion of performance results from this study was provided by Zhang et al. [27].

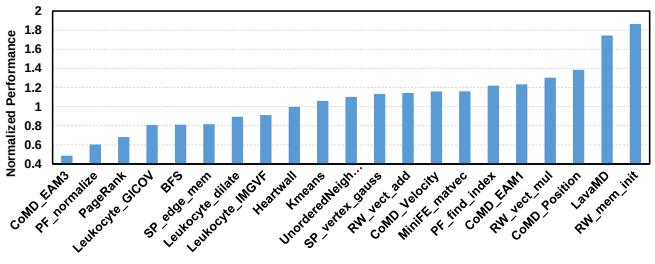

Figure 2: Normalized performance of PIM relative to host. *PF*, *SP*, and *RW* refer to *ParticleFilter*, *ShortestPath*, and *RandomWalk* benchmarks. Multiple kernels of a benchmark are labeled as *benchmark_kernel*.

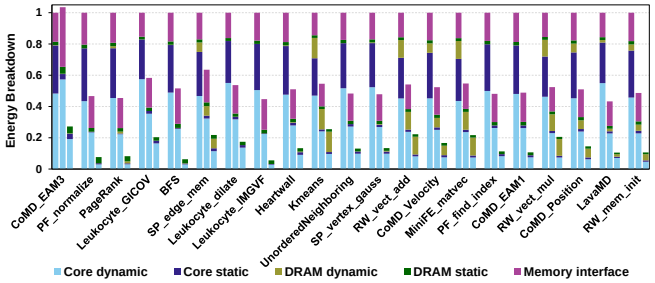

Figure 3: Normalized energy consumption of host (left bar of each group), small core (middle), and PIM (right). Energy consumed by the memory interface on the processor die (for host and small core) is included in core dynamic energy.

Figure 3 shows the energy consumed by the host, small core, and PIM configurations, normalized to the host. Comparing these three configurations enables us to evaluate the memory access efficiency of PIM independently of the impact of processor core heterogeneity. We assume the small cores are implemented close to the host die's memory interfaces and do not incur wire traversal overheads on the host die. Note, however, that these small cores must access memory via the host's conventional memory interface.

On average, the small cores reduce energy consumption by 47.5% (1.9× improvement) compared to host execution for the kernels evaluated when both processor and memory energy are considered, which highlights the contribution from processor heterogeneity. PIM further reduces energy by 73.7% (3.8× improvement) compared to the small cores, which largely corresponds to eliminating off-chip memory interface power on both memory and processor sides[3]. PIM also benefits from cases where kernel execution is memory-bound and the additional memory bandwidth available to PIM (4× the host bandwidth) results in faster execution times, reducing energy due to static and memory interface power. Conversely, compute-bound kernels take longer to execute on PIM and small cores due to their reduced arithmetic capabilities relative to the host. In these cases, the small cores are more severely affected due to their higher memory interface power consumption. *CoMD_EAM3* is an extreme case where the increased execution time results in small cores consuming more energy than host execution.

[3]Our assumptions throughout the evaluation bias in favor of the small cores over PIM, from assuming leakage identical to PIM to ignoring wire traversal power on the host die. Therefore, in reality, the benefits of memory interface savings of PIM may be slightly greater than reflected here.

3. PIM and Heterogeneity

The evaluation above demonstrates the complementary nature of the energy savings afforded by PIM and heterogeneous computing. While this study evaluated PIM based on a more power-efficient implementation of the same architecture as the host, these results encourage the exploration of other forms of heterogeneity in conjunction with PIM as well. One such option is to move all accelerators that are geared towards streaming or high memory bandwidth utilization (e.g., GPGPU) to PIM and remove them entirely from the host. Such a design can not only reap the benefits of PIM but, by moving memory-intensive computations to PIM, may also be able to reduce host energy consumption by using more energy-efficient memory interfaces that can still meet the reduced host requirements. Accelerators that are geared largely for energy efficiency may also be moved entirely to PIM.

Another form of heterogeneity that can further enhance the benefits of PIM is to adapt the processor microarchitecture for in-memory computation. While the above study used an off-the-shelf APU design for PIM, optimizing the compute units (e.g., increased load/store issue rates), cache hierarchy (e.g., fewer levels and reduced sizes), on-chip interconnect (e.g., tailor to distribution of memory channel TSVs), and other components to better match the characteristics of memory-bound kernels and to better utilize the stacked memory can yield additional benefits. Further, the tighter coupling between compute and memory in PIM could be exploited to implement novel memory access scheduling and page management policies, improved memory-side prefetching, finer-grain memory power management, in-memory atomics and synchronization primitives, and other such enhancements. While some PIM proposals (e.g., [5, 19]) incorporate a subset of these elements, they present specific designs with little exploration of the design space. Therefore, much remains to be done in quantifying the impact of microarchitectural decisions in the context of PIM, especially when applied to architectures such as GPUs that can leverage their existing software ecosystems to ease PIM adoption. Another promising direction is to implement non-traditional techniques such as approximate computing [10, 14] and reduced-precision arithmetic as PIM for even greater energy efficiency gains.

Fully enabling techniques such as those identified above require evolutions in the system-level and software ecosystems. Examples of necessary evolutions include memory interfaces that fully support PIM (including cache coherence and efficient synchronization), programming abstractions that ease data partitioning across multiple memory modules with PIM, system

software that can intelligently manage PIM accelerators, task schedulers that not only exploit processor heterogeneity but also data-compute affinity, virtual memory management techniques that enable a shared virtual address space among host and PIM devices, and power management solutions that efficiently allocate power to host or PIM computations. While many of these are not exclusive to PIM (i.e., accelerators in general and high-performance memory systems require many of the same evolutions), significant research opportunities remain to be addressed in these areas. However, we believe the potential energy savings demonstrated in this paper and the opportunities for even greater improvements motivate and justify research into realizing the full potential of effectively combining PIM and heterogeneous computing.

4. Acknowledgments

We would like to thank Joe Greathouse, Alexander Lyashevsky, and Mitesh Meswani for their contributions to the simulation methodology used in this work. AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

References

[1] "ITRS interconnect working group, 2012 update," www.itrs.net/links/2012Summer/Interconnect.pptx.

[2] *International Technology Roadmap for Semiconductors, 2011 Edition*, 2012 update.

[3] *High Bandwidth Memory (HBM) DRAM*, 2013.

[4] *Hybrid Memory Cube Specification 1.0*, 2013.

[5] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in *42nd Annual International Symposium on Computer Architecture*, 2015.

[6] AMD, "White paper: AMD graphics cores next (GCN) architecture," Jun 2012.

[7] A. Athavale and C. Christensen, *High-Speed Serial I/O Made Simple*, 1st ed., 2005.

[8] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: a benchmark suite for heterogeneous computing," in *International Symposium on Workload Characterization*, 2009.

[9] Y. Eckert, N. Jayasena, and G. Loh, "Thermal feasibility of die-stacked processing in memory," in *2nd Workshop on Near-Data Processing*, 2014.

[10] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2012.

[11] A. Farmahini-Farahani, K. Morrow, J. Ahn, and N. Kim., "NDA: near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules," in *Int. Symp. on High-Performance Computer Architecture*, 2015.

[12] M. Gokhale, B. Holmes, and K. Iobst, "Processing in memory: the Terasys massively parallel PIM array," *Computer*, vol. 28, no. 4, 1995.

[13] M. Hall, P. Kogge, J. Koller, P. Diniz, J. Chame, J. Draper, J. LaCoss, J. Granacki, J. Brockman, A. Srivastava, W. Athas, V. Freeh, J. Shin, and J. Park, "Mapping irregular applications to DIVA, a PIM-based data-intensive architecture," in *ACM/IEEE Supercomputing Conference*, 1999.

[14] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," *IEEE European Test Symposium*, 2013.

[15] M. Heroux, D. Doerfler, P. Crozier, J. Willenbring, H. Edwards, A. Williams, M. Rajan, E. Keiter, H. Thornquist, and R. Numrich, "Improving performance via mini-applications," SAND2009-5574, Tech. Rep., 2009.

[16] J. Jeddeloh and B. Keeth, "Hybrid memory cube new dram architecture increases density and performance," in *VLSI Technology (VLSIT), 2012 Symposium on*, 2012.

[17] Y. Kang, W. Huang, S.-M. 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*, 1999.

[18] R. Kumar, K. Farkas, N. Jouppi, P. Ranganathan, and D. Tullsen, "Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction," in *36th annual IEEE/ACM International Symposium on Microarchitecture*, 2003.

[19] R. Nair, S. Antao, C. Bertolli, P. Bose, J. Brunheroto, T. Chen, C. Cher, C. Costa, J. Doi, C. Evangelinos, B. Fleischer, T. Fox, D. Gallo, L. Grinberg, J. Gunnels, A. Jacob, P. Jacob, H. Jacobson, T. Karkhanis, C. Kim, J. Moreno, J. O'Brien, M. Ohmacht, Y. Park, D. Prener, B. Rosenburg, K. Ryu, O. Sallenave, M. Serrano, P. Siegl, K. Sugavanam, and Z. Sura, "Active memory cube: a processing-in-memory architecture for exascale systems," *IBM Journal of Research and Development*, vol. 59, no. 2/3, pp. 17:1–17:14, March 2015.

[20] M. Oskin, F. Chong, and T. Sherwood, "Active pages: a computation model for intelligent memory," in *25th Annual International Symposium on Computer Architecture*, 1998.

[21] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick, "Intelligent RAM (IRAM): chips that remember and compute," in *IEEE International Solid-State Circuits Conference*, 1997.

[22] J. T. Pawlowski, "Hybrid memory cube (HMC)," in *Hot Chips 23*, 2011.

[23] S. Pugsley, J. Jestes, H. Zhang, R. Balasubramonian, V. Srinivasan, A. Buyuktosunoglu, A. Davis, and F. Li, "NDC: analyzing the impact of 3D-stacked memory+logic devices on MapReduce workloads," in *International Symposium on Performance Analysis of Systems and Software*, 2014.

[24] G. Sandhu, "Dram scaling and bandwidth challenges," in *NSF Workshop on Emerging Technologies for Interconnects (WETI)*, 2012.

[25] T. Vogelsang, "Understanding the energy consumption of dynamic random access memories," in *43rd International Symposium on Microarchitecture*, 2010.

[26] G. Wu, J. Greathouse, A. Lyashevsky, N. Jayasena, and D. Chiou, "GPGPU performance and power estimation using machine learning," in *21st IEEE Symp. on High Performance Computer Architecture*, 2015.

[27] D. Zhang, N. Jayasena, A. Lyashevsky, J. Greathouse, L. Xu, and M. Ignatowski, "TOP-PIM: throughput-oriented programmable processing in memory," in *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, 2014.