Processing-in-Memory: Exploring the Design Space

Marko Scrbak[1], Mahzabeen Islam[1], Krishna M. Kavi[1], Mike Ignatowski[2], and Nuwan Jayasena[2]

[1] University of North Texas, USA
{markoscrbak,mahzabeenislam}@my.unt.edu,krishna.kavi@unt.edu
[2] AMD Research - Advanced Micro Devices, Inc., USA
{mike.ignatowski,nuwan.jayasena}@amd.com

Abstract. With the emergence of 3D-DRAM, Processing-in-Memory has once more become of great interest to the research community and industry. In this paper, we present our observations on a subset of the PIM design space. We show how the architectural choices for PIM core frequency and cache sizes will affect the overall power consumption and energy efficiency. Our findings include detailed power consumption modeling for an ARM-like core as a PIM core. We show the maximum number of PIM cores we can place in the logic layer with respect to a power budget. In addition, we explore the optimal design choices for the number of cores as a function of frequency, utilization, and energy efficiency.

Keywords: Processing-in-Memory, 3D-DRAM, Big Data, MapReduce

1 Introduction

Over the last decade, we have witnessed the Big Data processing evolution. Existing commodity systems, which are widely used in the Big Data processing community, are becoming less energy efficient and fail to scale in terms of power consumption and area. [21] clearly shows that this is also true for any Scale-Out workloads in general. With the evolution of new emerging DRAM technologies, in particular 3D-DRAM, Processing-in-Memory (PIM) has again become of great interest to the research community as well as the industry [15, 16]. When it comes to Big Data processing, systems with 3D-DRAM including PIM could prove to be more energy efficient and powerful than traditional commodity systems. Recent studies [14, 19, 8] have shown the potential use of PIM in 3D-DRAM chips. However, in order to prove the efficiency and usability of PIM, a much larger design space needs to be explored. This includes both software and hardware related design choices as well as tackling the challenges which arise from such a complex heterogeneous system. From a software perspective, challenges such as programmability, scalability, programming interfaces, and usability need to be explored. Major hardware challenges include PIM core micro-architecture, interconnection networks, and interfaces. In this paper, we present our observations for a subset of architectural choices for the PIM cores, e.g. core architecture, frequency, and cache sizes to maximize energy efficiency. Our goal is to explore a part of the large design space and investigate the trade-offs between certain design choices. We believe that our observations can be useful for narrowing down some architectural choices. We focus on an ARM-like energy-efficient core as a PIM core and evaluate design choices for caches, core frequency, and number of cores for a set of Big Data analyses benchmarks based on MapReduce. Our findings and observation include:

- How cache size and core frequency affect the performance of a single PIM core and total power consumption
- How these parameters and metrics translate to overall energy efficiency
- Power decomposition for different system components
- Potential number of cores we can place in the logic layer within a power budget
- Possible design choices for number of cores as a function of frequency, utilization, and energy efficiency

The rest of the paper is organized as follows. Section 2 covers the background and related studies, and Section 3 describes benefits and challenges of PIM in 3D-DRAM. Section 4 shows our contribution to the design space exploration. In Section 5 we describe the methodology and in Section 6 we present our results followed by a discussion. We conclude with Section 7 and discuss the future work.

2 Background and Related Work

2.1 3D-DRAM Memory

3D-DRAM memory provides memory access with lower latency, higher bandwidth and lower power consumption. A prototype of such 3D-DRAM is already available from Micron [22]. A group of different vendors, Hybrid Memory Cube Consortium (HMCC) [10], are working on expanding 3D-DRAM capabilities. Current prototype 3D-DRAM, known as Hybrid Memory Cube (HMC) has a capacity of 4GB and can provide maximum memory bandwidth of 320GB/s [10]. 3D-DRAM memory is typically going to consist of several layers of DRAM (nMOS) dies stacked on top of each other with a logic layer (CMOS) sitting on the bottom of the stack. Communication between different layers is done through high speed TSVs (Through Silicon Vias) [10, 9]. The logic die contains necessary interfacing circuits for the DRAM dies, and it still has enough area to accommodate additional logic [14, 19]. The proposed TDP budget of the logic layer is conservatively set at 10W [19].

2.2 An Overview on PIM

Processing-in-Memory (PIM) is the concept of putting computation as close as possible to memory to get faster access to memory and achieve higher bandwidth. Processing logic can be integrated in different levels of the storage hierarchy, e.g.

cache, memory (DRAM), permanent storage (Solid State Drive-SSD). In this study, we focus only on processing in DRAM memory.

Research in the area of PIM can be categorized into two eras from the implementation point of view. In the first era, researchers relied on a processing technology that tried to combine both logic and DRAM cells on a single die. However the incompatibilities in the manufacturing process of these different types of devices made it difficult to integrate DRAM with logic [15, 23]. The invention of 3D-die stacking technology breathed a new life for PIM research. 3D-Die stacking technology enables two disparate technologies to be integrated in the same die. It provides a very useful way of constructing a single die that can offer both dense memory and fast logic. Also, some other common challenges anticipated by the researchers of the past PIM studies seem to be easily solved with 3D-DRAM technology.

PIM, Previous Studies. From the 1990s to 2005, a number of studies proposed appropriate architectures employing PIM to achieve lower memory latency, higher memory bandwidth and high throughput. Some interesting studies from that era include EXECUBE [24], IRAM [13], FlexRAM [15], Smart Memories [25], DIVA [11], and Intelligent Memory Manager [4]. In most of the work, the researchers advocated architectures with vector [13] or SIMD type [24, 15, 11, 4] processing units sitting close to the memory arrays.

PIM, Related Studies. Recently proposed Near Data Computing (NDC) architecture [14] and PIM for MapReduce applications [8] propose to integrate simple ARM cores as PIM cores in 3D-DRAM memory and have shown performance and energy gains. In our study, we closely resemble the architecture but the goal of our study differs. In this paper, we explore the design space of PIM cores utilizing MapReduce applications as a use case. In TOP-PIM [19] the researchers presented a 3D-DRAM PIM model with GPUs as PIM cores. For different process technologies, they have shown significant energy efficiency with little or no performance degradation for different HPC and graph applications. Other studies [15, 2, 3] have provided useful insights on research directions for PIM-augmented 3D-DRAM systems.

3 PIM Integrated 3D-DRAM: Looking into the Future

In data center systems we need to process large amounts of data as fast as possible. The main bottleneck in achieving higher speed processing is the gap between processor and memory speed. Here we discuss the two most important issues which create this problem, namely latency and bandwidth. Energy efficiency is another crucial requirement for today's data centers. 3D-DRAM memory cubes provide memory accesses with lower latency, higher bandwidth and lower power consumption. PIM cores integrated in the logic layer of 3D-DRAM are expected to capitalize these benefits.

Latency. Memory access latency for a commodity processor can be divided in two parts [13]. The first part is the time to send the address bits to the DRAM. This includes lookups in the cache hierarchy, memory controller overhead, multiplexing the address over the system memory bus, and reaching the DRAM pins, etc. The second part is the core DRAM access latency, which may include row precharge time (tRP), row address to column address delay time (tRCD) and column access delay time (tCAS). DRAM core latency is approximately 40-50ns [14, 5]. PIM core's DRAM access latency will be reduced by the lookup time for L2 and L3 caches as it only has L1 caches. In addition, the off-chip memory bus delay can be avoided as the PIM cores reside in the same stack as the DRAM dies and are connected with high speed TSVs. The reduction in DRAM access latency is expected to be at least 30% [2].

Bandwidth. Today's processors, which typically have superscalar pipelines,support Out-Of-Order execution, and support speculation need an excessive amount of data per second. A good part of data can be supplied by large caches. However, present data intensive applications, e.g. Scale-Out applications [21], do not benefit from deep cache hierarchies and demand more memory accesses resulting in a high bandwidth requirement. Additionally, non-blocking and prefetch-enabled caches increase this requirement. The invention of 3D-DRAM memory can provide a viable solution to the high bandwidth requirement. Current prototypes [10] offer as much as 320GB/s off-chip memory bandwidth. Ser-Des links are used to support this high memory bandwidth. Each Ser-Des link can support 40GB/s while consuming high power, and in order to provide 320GB/s, 8 such links are required. This bandwidth (320GB/s) is also available to the logic die sitting at the bottom of the stacked DRAM dies through TSV buses. If we integrate PIM cores into the logic layer they will be able to utilize the high bandwidth without requiring Ser-Des links.

Power. The memory subsystem (memory chip, I/O interface and link) is power hungry, and in modern Petascale systems, it consumes approximately 35% of the total system power budget and is anticipated to consume more than 60% in future Exascale systems [6]. 3D-DRAM will be able to provide 72% less energy per bit as compared to current DDR4 DRAM systems [18]. Nonetheless accessing off-chip memory has high overhead in terms of energy. Studies have shown that around 50%-70% of the DRAM access energy is consumed by the interfaces [6, 7]. Other studies show that approximately 20-30 pJ/b are spent when transferring data over DRAM buses [7], 5-10 pJ/b for Ser-Des links, and it is expected to be only 30-110 fJ/b when traversed along 3D TSV [19]. Thus, PIM integrated systems would be more energy efficient when running data-intensive workloads.

Challenges. There exist a number of issues which need to be solved for PIMs to be effective. The crucial challenge is designing an appropriate system architecture. This involves many design parameters, such as, the host processor, PIM processors, the memory hierarchy, communication channels, interfaces, etc. Also a number of changes must be made to the operating system (e.g. memory management), programming framework (e.g. libraries), and programming models (e.g. synchronization, coherence, data layout).

4 PIM Design Space Exploration

A general model of a PIM augmented architecture using 3D-DRAM has been proposed by Zhang et al. [16] and a similar model has been used in recent studies [14, 8] as well. We use the same model for our studies. The model consists of a host processor connected to one or many 3D-DRAM modules where each 3D-DRAM module has several PIM cores residing in the logic layer. The host processor views all the 3D-DRAM modules as one physical address space shared between the host processor and the PIM cores.

Previous studies have shown high performance gains and energy reductions for PIM-augmented architectures running MapReduce workloads [14, 8]. However, the power analyses performed in these studies, for ARM-like PIM cores, are not accurate. The overall power consumption of the PIM core is underestimated, and not all power components are considered, e.g. cache power. Furthermore, the studies are limited for a fixed cache size and core frequency.

In this paper, our goal is to explore the design space of the PIM cores in terms of cache sizes, operating frequency, the number of cores for a specific microarchitecture, and perform more realistic power estimations. We take an in-order, single issue, ARM-like core and perform simulations for different MapReduce workloads. Our focus is on the map() phase of MapReduce workloads because they are data intensive and highly parallelizable. We have used gem5 [20] to capture the performance statistics of the core and McPAT [26] and CACTI-3DD [27] for the power analyses.

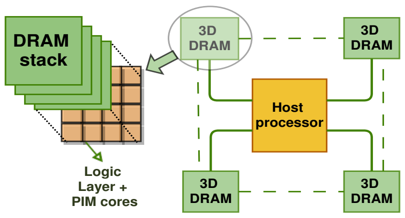


Fig. 1: A PIM augmented system comprising of 4 3D-DRAM cubes with several PIM cores embedded in the logic layer.

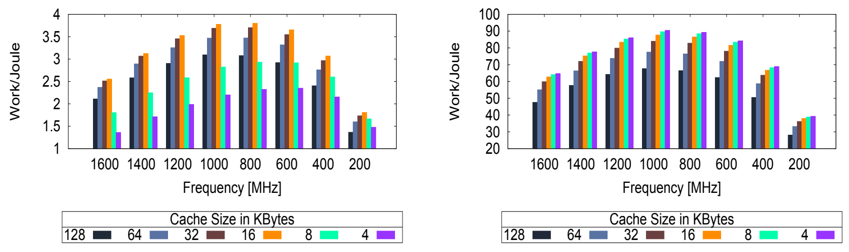
The architectural choices for cache size and frequency for the PIM cores will depend on two metrics, i.e. power consumption and energy efficiency. Total power consumption of a PIM core is an important factor because it limits the number of cores we can place in the logic layer within a power budget of 10W. We define the energy efficiency as useful work done per unit of energy [work/Joule]. We do not focus solely on total execution time, because it would imply the largest cache size and highest frequency as optimal choices. This is not a good approach because we want to minimize the power consumption while maximizing the performance. We performed experiments with varying L1 cache sizes with and without enabled prefetching. We have observed a moderate cache size with prefetch offers the best energy efficiency. The reason behind this is the low temporal locality and streaming-like behavior of map() phases in MapReduce workloads. Note that including another level of cache would consume a significant amount of power without providing a significant performance improvement. We also vary the PIM core frequency and adjust the supply voltage accordingly [28] to ensure a minimal supply voltage. There will be an optimal frequency for which we get the best energy efficiency. Because the power increases exponentially and execution time reduces linearly, higher frequencies than optimal will result in low energy efficiency due to high power. Lower frequencies will result in lower energy efficiency due to high execution times.

We also calculate the maximum number of cores we can place in the logic layer within the power budget of 10W. Note that the maximum number of cores may not be the optimal choice since the utilization of the cores will depend on the application which will run on the PIM cores. We therefore evaluate the optimal number of cores we want to place in the logic layer with respect to minimal execution time and minimal energy spent. We calculate the execution times using Amdahl's law for different possibilities of serial fractions. We reason that, although the computation done on PIM cores is typically going to be parallel, there may be some overhead due to communication, synchronization, or load imbalance. We observe that the more overhead we have, the fewer cores we want in the logic layer. We do not get significant performance gains with increasing the number of cores but add unnecessary power consumption. If we do not place the maximum number of cores, we hardly utilize the available bandwidth within a 3D stack. This leads to a conclusion that a SIMD/VLIW/vector processor architecture, which can consume much more bandwidth, should be considered as a PIM core. We plan to investigate such designs in the future.

5 Methodology

We used the gem5 simulator [20] to capture the performance statistics needed for our power and energy efficiency evaluation. We used the "minor" CPU, an in-order, single-issue CPU model with support for ARM ISA. We are aware that this model is not as detailed, but it is the only available in-order model with ARM ISA support. We used a simple DRAM model with a fixed latency of 40ns [14] to match the latency of the 3D-DRAM. We ran four different micro-benchmarks, written in the C programming language, which capture the map() function behavior of common MapReduce applications. After input reading, we take a snapshot of the execution and simulate the run only for the map() function. We perform the simulations for four micro-benchmarks, wordcount, histogram, linear regression, and string match. We vary the L1 cache sizes and core frequencies. L1 cache means split instruction and data caches of the same size, e.g. 16KB L1 cache means a 16KB L1 instruction and 16KB L1 data cache. We use a 64B block size for cache.

For the power consumption modeling we used McPAT [26], a power modeling tool with support for power, area and timing optimization. The tool uses a CPU model description and the corresponding performance statistics for an application run. We take the needed input parameters from gem5 statistics outputs and feed them into McPAT. We do so for each benchmark we run with different cache sizes and frequencies. We adjust the supply voltage for each frequency accordingly. This also allows us to capture the correct increase in power while varying the frequency. The chosen voltage-frequency pairs mimic those in [28]. To keep the static power consumption low, we allow power-gating. All the power estimations were conducted with respect to the 40nm process, and technology parameters follow the ITRS roadmap. We have modeled a 3D-DRAM with respect to JEDEC-HBM [12] standard using CACTI-3DD [27]. We obtained the 3D-DRAM access energy of 3.98pJ/bit which is close to 3.7pJ/bit as presented in [14]. The next section describes the experiments and results in more detail followed by a discussion.



(a) wordcount, work=2MB.

(b) histogram, work=lena.bmp.

Fig. 2: Energy efficiency for two MapReduce workloads. (a) A configuration with 16KB L1 cache and a frequency of 800MHz results in the best energy efficiency. A frequency of 1GHz provides almost the same energy efficiency and represents a better alternative in terms of performance at the cost of higher power consumption. (b) A configuration with 4KB cache and 1GHz frequency results in the best energy efficiency.

6 Results and Discussion

6.1 PIM Core Frequency and Cache Size

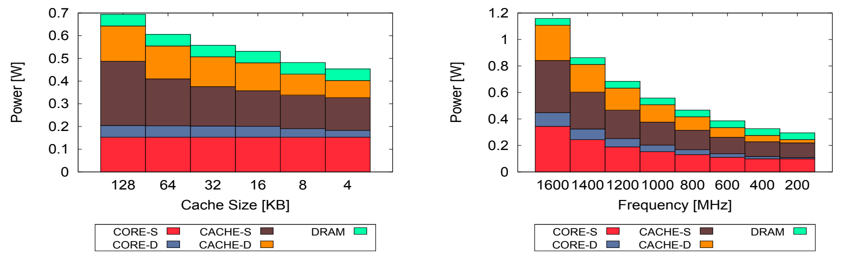
We use the collected statistics from gem5 to evaluate what would be good architectural choices for cache sizes and core frequencies. In order to do that, we look at the overall energy efficiency for different cache size-frequency pairs. The goal is to find an optimal point where we get the most out of the PIM core for the lowest possible power consumption. For that, we take the total execution time obtained from gem5 and the power consumption of the core obtained from McPAT [26]. We include both static and dynamic power consumption, for the core and caches, as well as the dynamic 3D-DRAM power obtained from CACTI-3DD [27]. It is important to include the dynamic DRAM power consumption because smaller cache sizes can create more accesses to the DRAM and result in increased overall power consumption. We calculate the energy efficiency, $\mathrm{E_{eff}}$ as $\mathrm{E_{eff}} = 1/\mathrm{Energy}$ where,

$$\begin{aligned}\mathrm{Energy} = &(\mathrm{CPU\_Power} \times \mathrm{Total\_Execution\_Time}) \\ &+(\mathrm{Number\_of\_Memory\_Access} \times \mathrm{DRAM\_Access\_Energy})\ . \end{aligned} \quad (1)$$

Figure 2 shows the overall PIM core energy efficiency in Work/Joule for two distinct workloads: wordcount and histogram. We do not present the data for the other two workloads since they exhibited similar behavior as that of histogram. The data shows that, for applications like wordcount, a PIM core with 16KB L1 cache running at 800MHz frequency is the most energy efficient choice. For applications similar to histogram a 4KB L1 cache and a frequency of 800MHz results in the most energy efficient setup. We acknowledge that the actual values for cache sizes and operating frequencies may be benchmark dependent, and we plan to conduct further experiments with several other benchmarks. Our goal here is to present an approach for exploring these design choices. We do, however, believe that, if we are using ARM like cores, for most MapReduce applications, where map functions will be executed by PIM cores, the best operating frequencies will range between 600MHz-1000MHz and the optimal cache sizes range between 8KB-32KB. From our results, we observe that the applications don't benefit from larger caches and therefore a second level of cache would just introduce more power overhead and not provide performance gains. Thus, we do not evaluate the use of L2 caches.

6.2 Power Consumption

We obtain the total PIM core power consumption from McPAT [26]. We scale the supply voltage to support various frequencies by using the voltage-frequency pairs as in [28]. We separate the power consumption into four different components: static core power, dynamic core power, static cache power, and dynamic cache power. The power consumption will depend on both frequency and supply voltage and, therefore, will scale exponentially. Figure 3 shows the breakdown of different power components within a PIM core. For a cache size of 32KB and core frequency of 1GHz, the total PIM power consumption (including cache power) is around 500mW. The core dynamic power is roughly 50mW which supports the published data for an energy-efficient in-order ARM core [17]. Previous studies [14, 8] used the power specifications for the same ARM core and took into consideration only the core dynamic power consumption. However, we notice that the core static power and the cache power are the most significant components and should be taken into account. Even after allowing for power-gating, static power consumption is high. This implies that the PIM cores should be turned off whenever they are not performing computation. We include the dynamic power of the DRAM to capture the effects of cache sizes.



(a) The effect of cache size on the power consumption for a fixed frequency 1GHz.

(b) The effect of frequency on total power consumption for a fixed L1 cache size of 32KB.

Fig. 3: Power decomposition for PIM core components. A significant portion of the power comes from the static power for core and cache. A configuration with a cache size of 32KB, and core frequency of 1GHz, consumes 153mW of static and 50mW of dynamic core power, and 173mW of static and 131mW of dynamic cache power. The DRAM dynamic power consumption is 51mW. The power consumption was modeled using McPAT with enabled power-gating.

6.3 Number of PIM Cores

The maximum number of PIM cores that can be placed in the logic layer of a 3D-DRAM will depend on the individual PIM core power consumption as well as the power limit of the logic layer. Researchers have proposed a conservative power budget of 10W for the logic layer [19].

Figure 4 shows the maximum number of cores, within that power budget, for different setups. For 800MHz and 16KB L1 cache, we can maximally put 26 cores in the logic layer. Due to various parallel overheads, the parallel code which will run on the PIM cores may result in lower utilization of the PIM cores. Therefore, we reason about a good number of PIM cores with respect to Amdahl's law; so, we maintain good performance while minimizing the energy. The rate at which the power increases with the number of cores will be higher than the obtained speedup. We are trying to find the trade-off between energy consumption and execution time. We do that by calculating the execution times for different numbers of cores using Amdahl's law for different parallel overheads (serial fractions). For specific core parameters (cache, frequency), we vary the number of cores and obtain different execution times by using Amdahl's law. The obtained execution time for n cores, Total_Execution_Time(n), is then used to calculate the energy consumed by n cores, E(n).

$$E(n) = n \times \text{CPU\_Power} \times \text{Total\_Execution\_Time}(n) \ . \tag{2}$$

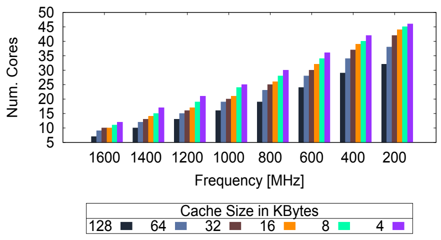
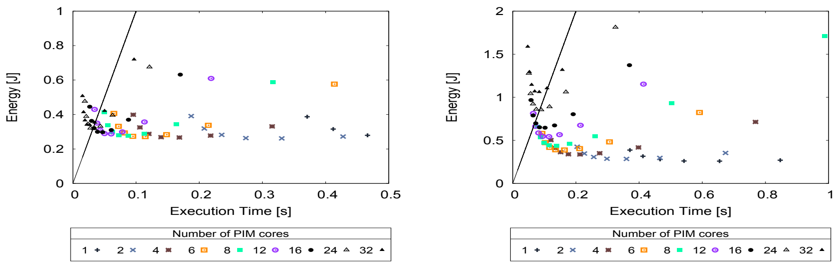
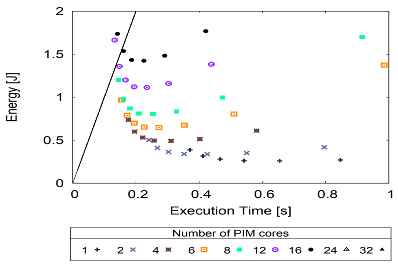


Fig. 4: Maximum number of PIM cores we can place in the logic layer within a power budget of 10W. We can place as many as 26 ARM-like PIM cores, with a 16KB L1 cache running on 800MHz, and not exceed the 10W power budget.



(a) Parallel overhead = 1%

(b) Parallel overhead = 10%



(c) Parallel overhead = 30%

Fig. 5: Time-Energy pairs for 3 different parallel overheads. The desirable number of PIM cores are those closer to the (0,0) coordinate. As the parallel overhead increases, the configurations with more cores "drift away" because more cores do not provide additional performance and increase the power consumption. The black line represents the 10W power budget. All the configurations which are on the left-hand side of the slope are not possible, since they exceed the power limit. For a parallel overhead of 1% we want as many as 16-24 cores, for 10% overhead 8-12 and for 30% 4-6 cores. For each number of cores, we plot the points for different frequencies starting with the largest frequency (1600MHz) on the left most side and ending with the lowest frequency (200MHz) on the right most side. Note that the 800MHz frequency still gives the best results in terms of energy and time.

We compute E(n) for different frequencies so we can observe different design alternatives. We plot the time-energy pairs in a 2D plane. The points closest to the optimum point (0,0) will be the configurations which are optimized for both performance and energy. Figure 5 shows how the desired number of cores changes because of Amdahl's law. The general observation is the more overhead we have, the fewer cores we want in the logic layer. For a parallel overhead of 1% we want as many as 16-24 cores, for 10% overhead 8-12 and for 30% 4-6 cores.

The desired number of cores depends on the parallel overhead and is subject to Amdahl's law. Therefore, it would be wise to choose highly parallelizable applications with no parallel overhead to run on PIM. If we assume that more general applications are going to run on PIM we might consider putting less cores and not waste additional energy.

7 Conclusion and Future Work

In this paper, we presented our observations on a subset of architectural choices for PIM cores. As a use case, we have used map() phases of several MapReduce workloads. Our study shows that a PIM core running at 800MHz clock frequency, with 16KB instruction cache and 16KB data cache, provides the best energy efficiency. In addition, we have shown the power consumption components and calculated the maximum number of cores we can place in the logic layer. Also, we have shown how the parallel overhead of a program can limit the advantage of having a larger number of cores in the logic layer.

In the future we want to explore other possible micro-architectures for PIM cores such as simple RISC cores, VLIW processors, vector processors and Dataflow. Also, we would like to characterize which applications benefit from a PIM architecture and how to exploit the possible benefits.

Acknowledgments. This work is conducted in part with support from the NSF Net-centric IUCRC and AMD. We acknowledge Jim Buchanan's help in making this paper more readable.

References

1. Kogge, P. M., Jay, B. B., Sterling, t., Guang, G.:Processing In Memory: Chips to Petaflops. In: Workshop on Mixing Logic and DRAM: Chips that Compute and Remember at ISCA, vol. 97. (1997)
2. Zhang, D. P., Jayasena, N., Lyashevsky, A., et al.: A new perspective on processing-in-memory architecture design. In: Proceedings of the ACM SIGPLAN Workshop on Memory Systems Performance and Correctness, p. 7. ACM, (2013)
3. Loh, G., Jayasena, N., Oskin, M., et al.: A Processing in Memory Taxonomy and a Case for Studying Fixed-Function PIM. In: WoNDP: 1st Workshop on Near-Data Processing. (2013)
4. Rezaei, M., Kavi, K. M.: Intelligent memory manager: Reducing cache pollution due to memory management functions. In: Journal of Systems Architecture, 52(1), 41-55. (2006)

5. Chang, D. W., Byun, G., Kim, H., et al.: Reevaluating the latency claims of 3D stacked memories." In: Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific, pp. 657-662. IEEE, (2013)
6. Gara, A.: Energy efficiency challenges for exascale computing. In: ACM/IEEE Conference on Supercomputing: Workshop on Power Efficiency and the Path to Exascale Computing. (2008)
7. Keckler, S. W., Dally, W. J., Khailany, B.: GPUs and the future of parallel computing. In: IEEE Micro 31, no. 5 (2011): 7-17. (2011)
8. Islam, M., Scrbak, M., Kavi, K. M., et al.: Improving Node-level MapReduce Performance using Processing-in-Memory Technologies. In: to be appear in Workshop on UnConventional High Performance Computing 2014.
9. Black, B., Annavaram, M., Brekelbaum, N., DeVale, et al.: Die stacking (3D) microarchitecture. In: Micro, pp. 469-479. IEEE, (2006)
10. Hybrid Memory Cube Consortium, `http://hybridmemorycube.org/`
11. Draper, J., Chame, J., Hall, M., et al.: The architecture of the DIVA processing-in-memory chip. In: Proceedings of the Supercomputing, pp. 14-25. ACM, (2002)
12. JEDEC, `http://www.jedec.org/category/technology-focus-area/3d-ics-0`
13. Patterson, D., Anderson, T., Cardwell, N., et al.: A case for intelligent RAM. In: Micro, 17(2), 34-44. IEEE, (1997)
14. Pugsley, S. H., Jestes, J., Zhang, H.: NDC: Analyzing the Impact of 3D-Stacked Memory+Logic Devices on MapReduce Workloads. In: International Symposium on Performance Analysis of Systems and Software. (2014)
15. Torrellas, J.: FlexRAM: Toward an advanced Intelligent Memory system: A retrospective paper. In: Intl. Conference on Computer Design, pp. 3-4. IEEE, (2012)
16. Zhang, D. P., Jayasena, N., Lyashevsky, A., et al.: A new perspective on processing-in-memory architecture design. In: Proceedings of the ACM SIGPLAN Workshop on Memory Systems Performance and Correctness, p. 7. ACM, (2013)
17. ARM, `http://www.arm.com/products/processors/cortex-a/cortex-a5.php`
18. Graham, S.: HMC Overview. In: memcon Proceedings. (2012)
19. Zhang, D., Jayasena, N., Lyashevsky, A., et al.: TOP-PIM: throughput-oriented programmable processing in memory. In: Proceedings of international symposium on High-performance parallel and distributed computing, pp. 85-98. ACM, (2014)
20. gem5 Simulator System, `http://www.m5sim.org`
21. Ferdman, M., Adileh, A., Kocberber, O., et al.: A Case for Specialized Processors for Scale-Out Workloads. In: Micro, pp. 31-42. IEEE, (2014)
22. Hybrid Memory Cube, Micron, `http://www.micron.com/products/hybrid-memory-cube`
23. Brockman, J. B., Kogge, P. M.: The Case for Processing-in-Memory. In: Reports in University of Notre Dame. (1997)
24. Kogge, P. M.: EXECUBE-A new architecture for scaleable MPPs. In: International Conference on Parallel Processing, vol. 1, pp. 77-84. IEEE, (1994)
25. Mai, K., Paaske, T., Jayasena, N., et al.: Smart memories: A modular reconfigurable architecture. In: Vol. 28, no. 2. ACM, (2000)
26. McPAT, HP Labs, `http://www.hpl.hp.com/research/mcpat/`
27. Chen, K., Li, S., Muralimanohar, N., et al.: CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 33-38. EDA Consortium. (2012)
28. Spiliopoulos, V., Bagdia, A., Hansson, A., et al.: Introducing DVFS-management in a full-system simulator. In: Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 535-545. IEEE, (2013)