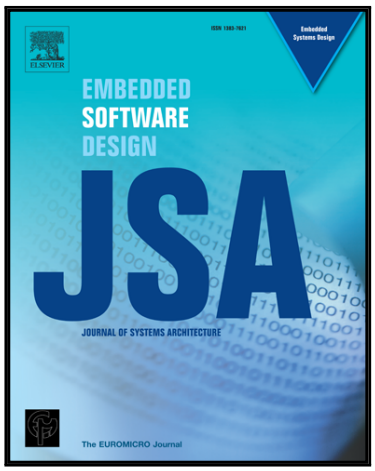


Exploring the Processing-in-Memory Design Space

Marko Scrbak , Mahzabeen Islam , Krishna M. Kavi ,
Mike Ignatowski , Nuwan Jayasena

PII: S1383-7621(16)30086-8
DOI: 10.1016/j.sysarc.2016.08.001
Reference: SYSARC 1378

To appear in: *Journal of Systems Architecture*

Received date: 3 May 2016
Accepted date: 6 August 2016

Please cite this article as: Marko Scrbak , Mahzabeen Islam , Krishna M. Kavi , Mike Ignatowski , Nuwan Jayasena , Exploring the Processing-in-Memory Design Space, *Journal of Systems Architecture* (2016), doi: 10.1016/j.sysarc.2016.08.001

Exploring the Processing-in-Memory 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

1 ABSTRACT

With the emergence of 3D-DRAM, Processing-in-Memory has once more become of great interest to the research community and industry. Here 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. We include a detailed power consumption breakdown for an ARM-like core as a PIM core. We show the maximum possible number of PIM cores we can place in the logic layer with respect to a predefined power budget. Additionally, we catalog additional sources of power consumption in a system with PIM such as 3D-DRAM link power and discuss the possible power reduction techniques. We describe the shortcomings of using ARM-like cores for PIM and discuss other alternatives for the PIM cores. Finally, we explore the optimal design choices for the number of cores as a function of performance, utilization, and energy efficiency.

Keywords: Processing in memory; Heterogeneous Computing; High Performance Computing; 3D-DRAM; Energy Efficient Computing

2 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. Therefore using hardware accelerators to aid the Big Data processing is becoming more and more prominent. 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. Here 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 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 as well as scientific OpenMP benchmarks. 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 with respect to a power budget
- Possible design choices for number of cores as a function of frequency, utilization, and energy efficiency
- Bandwidth consumption of the benchmarks and the impact on the decision about the number and type of PIM cores
- Discussion on alternative choices for the PIM cores

3 Background and Related Study

3.1 3D-DRAM

3D-DRAM memory provides high memory bandwidth, which reduces average memory-access latency, and lower power consumption than traditional DRAM. 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 4-8GB and can provide maximum memory bandwidth of 480GB/s [10]. 3D-DRAM memory is typically consists 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 processing or controller logic [14, 19]. The proposed TDP budget of the logic layer is conservatively set at 10W [19] while recent studies show that it can be increased by using more effective cooling solution [29].

3.2 An Overview on PIM

Processing-in-Memory (PIM) is the concept of placing 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 also provided useful insights on research directions for PIM-augmented 3D-DRAM systems.

4 PIM Integrated 3D-DRAM

In present 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, known commonly as the famous memory-wall. 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 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 480GB/s off-chip memory bandwidth. SerDes links are used to support this high memory bandwidth. Each SerDes link can support 120GB/s while consuming high power, and in order to provide 480GB/s, 4 such links are required. This bandwidth 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 SerDes 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 SerDes 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).

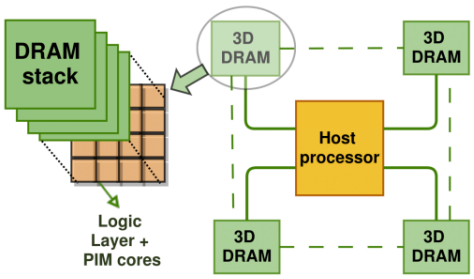


Figure 1 – A system with Processing-in-Memory capabilities consisting of a host processor and multiple 3D-DRAM stacks, each stack with a certain number of processing cores in the logic layer. Inter-stack links are optional but should be considered allowing for off-stack data accesses for PIM.

5 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. 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 as well as scientific OpenMP benchmarks. We have used gem5 [20] to capture the performance statistics of the core and McPAT [26] and CACTI-3DD [27] for the power analyses.

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 and large structured datasets for OpenMP 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.

6 Methodology

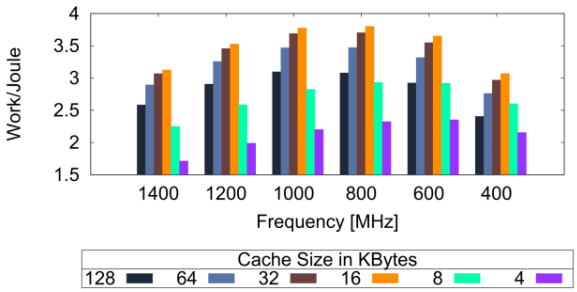
We used the gem5 simulator [20] to capture the performance statistics needed for our power and energy efficiency evaluation. We used the "minor" CPU, an in-order, single-issue CPU model with support for ARM ISA. We are aware that this model is not as detailed, but it is the only available in-order model with ARM ISA support. We used a simple DRAM model with a fixed latency of 40ns [14] to match the core latency of the 3D-DRAM. We ran four different microbenchmarks, written in the C programming language, which capture the map() function behavior of common MapReduce applications and four OpenMP benchmarks from the Rodinia benchmark suite [30]. We execute only map() phases of MR workloads. Because of their high degree of parallelism, map tasks seem to be most suitable for PIM-like architectures. OpenMP benchmarks are also highly parallel and could run efficiently on PIMs. We perform the simulations for four MapReduce micro-benchmarks, wordcount, histogram, linear regression, and string match and four OpenMP benchmarks from the Rodinia suite, backprop, bfs, nn and euler3d. 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.

7 PIM cores frequency and cache sizes

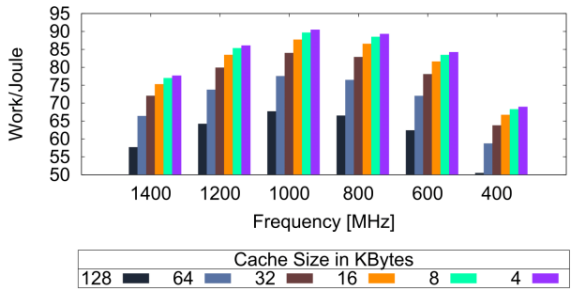
We use the collected statistics from gem5 to evaluate what would be good architectural choices for cache sizes and core frequencies. In order to do that, we look at the overall energy efficiency for different cache size-frequency pairs. The goal is to find an optimal point where we get the most out of the PIM cores with 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, Eeff as Eeff = 1/Energy where

$$Energy = (CPU\ Power * Exec.Time) + (no.of\ mem.accesses * DRAM\ access\ energy)$$

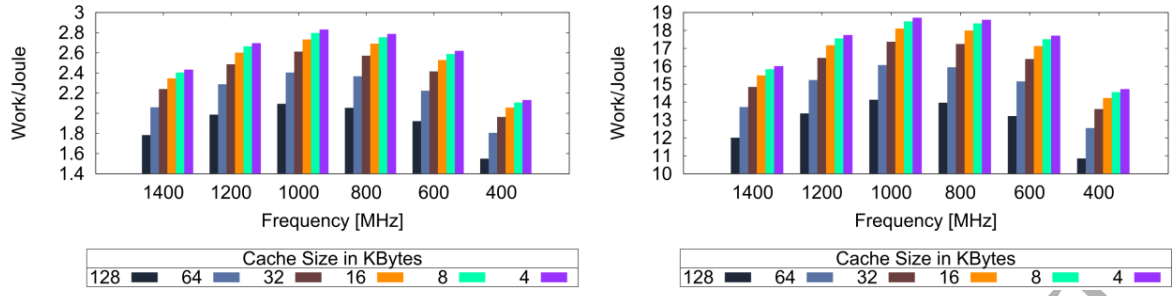
Figure 2 (a through d) shows the overall PIM core energy efficiency in Work/Joule for the MapReduce workloads. The data shows that, for applications like wordcount, a PIM core with 16KB L1 cache running at 800MHz frequency is the most energy efficient choice. For applications similar to histogram, linear regression and string match, a 4KB L1 cache and a frequency of 800MHz results in the most energy efficient setup. Even though a lower frequency seem to be less energy efficient it could prove useful to ramp down the processing frequency in order to save power when operating under stricter power budgets. This is especially important when multiple PIM modules are in place since the aggregate power will increase significantly because of the large number of cores in the system. A reduction in dynamic power directly translates in reduction in power dissipation which is of great importance in large scale systems. Similarly, on the performance end, we can ramp up the frequency to get higher performance if power is not critical. Looking at the data we conclude 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 will range between 8KB-32KB. From our results, we observe that the MapReduce workloads do not benefit from larger caches and therefore a second level of cache would just introduce more power overhead and not provide performance gains.



a) wordcount



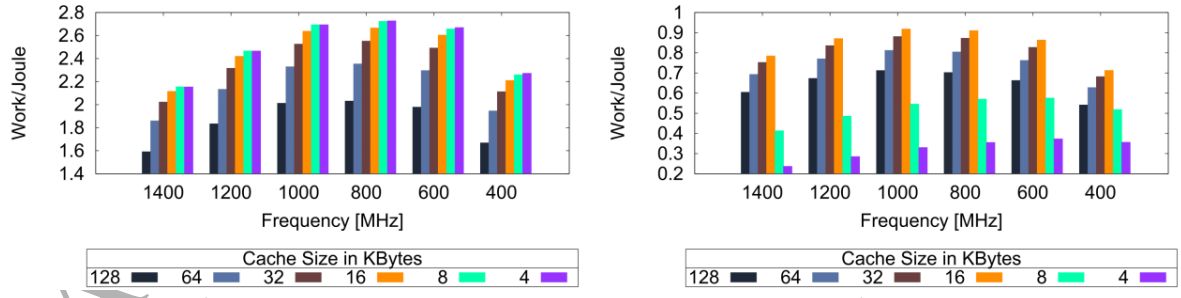
b) histogram



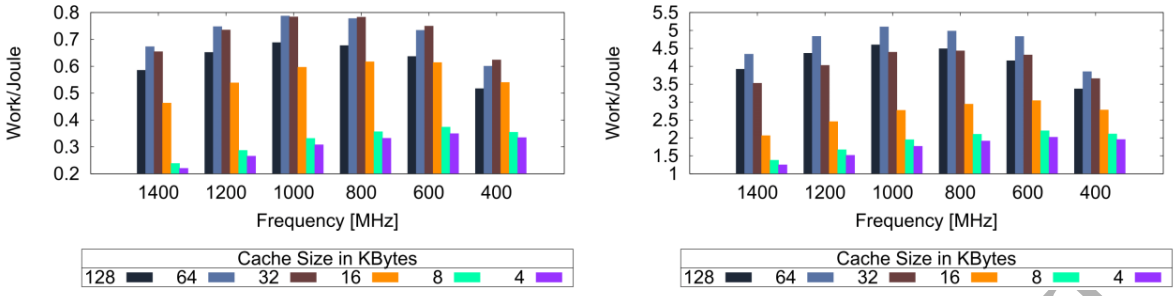
c) linear regression d) string match

Figure 2 (a-d) - Energy efficiency of MapReduce workloads running map() function on an ARM core. Work equals to the input processed by one map() task. PIM cores running on a frequency of 800MHz show highest 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, while a frequency of 600MHz can save power at cost of performance. Smaller cache sizes are more energy efficient due to poor data locality in MR tasks.

Figure 3 (a-d) shows the PIM energy efficiency for the four OpenMP workloads from the Rodinia benchmark suite. Applications like backprop and bfs which do not benefit from the larger caches show higher energy efficiency for lower cache size (4KB – 16KB). This is not true for the other two benchmarks (euler3d and nn) which show higher energy efficiency for cache sizes ranging from 32KB-128KB. Note that in the case of the nn benchmark a PIM core running on 1000MHz and 64KB of cache is twice as efficient as the PIM core with only 16KB at the same frequency. The euler3d benchmark benefits from a larger cache size at higher frequencies (64 KB cache size is more efficient) while at lower frequencies a smaller cache size seems more energy efficient (e.g. 32KB at 600MHz). This is because euler3d is latency sensitive and has a larger working set size than other benchmarks. A higher frequency will cause the PIM core to stall on memory accesses and thus a bigger cache will amortize this latency, while on lower frequencies the PIM core will spend less cycles sitting idle, because the memory latency is fixed in terms of time, and thus a smaller cache suffices. It is important to note that larger caches will increase the static power consumed by the cores and thereby reduce the total number of cores we can place in the logic layer under a power budget.



a) backprop b) bfs

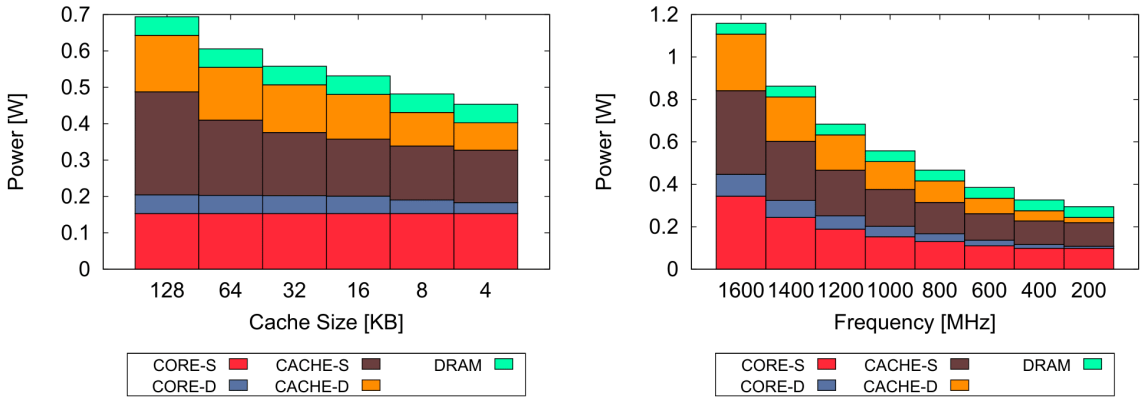


c) euler3d **d) nn**

Figure 3(a-d)- Energy efficenncy of OpenMP Rodinia benchmarks. Higher data locality for euler3d and nn makes good use of larger caches. However, large caches will reduce the number of cores we can place in the logic layer under a power budget.

8 Power Breakdown

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



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

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

Figure 4a, 4b - Power breakdown of an ARM PIM core when running wordcount. Other workloads exhibit similar behavior where the static power is the major power component and where caches consume most of the static power. Static power doesn't depend on the workload while the change in dynamic power consumption is +/- 100mW for the chosen benchmarks.

voltage and, therefore, will scale exponentially. Figure 4a and 4b show 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. Furthermore, the size of the cache as well as the core frequency will be the major factor limiting the number of cores we can place in the logic layer. We include the dynamic power of the DRAM to capture the effects of cache sizes.

9 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 5 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, while at 1000MHz and 64 KB we can put a maximum of 18 cores. This suggests that 16-24 cores would seem the most reasonable choice when it comes to the number of cores.

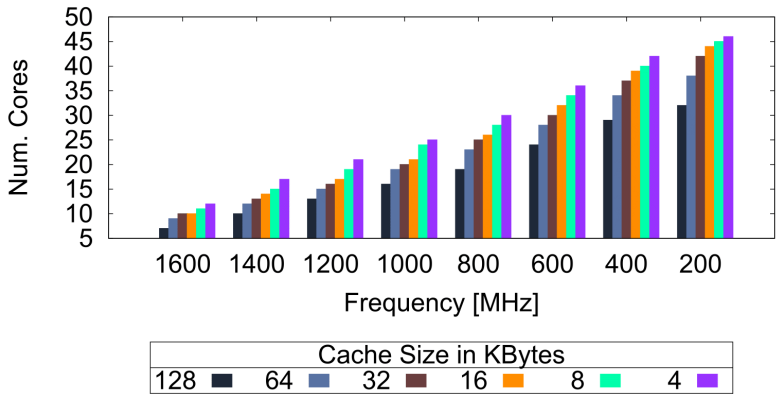
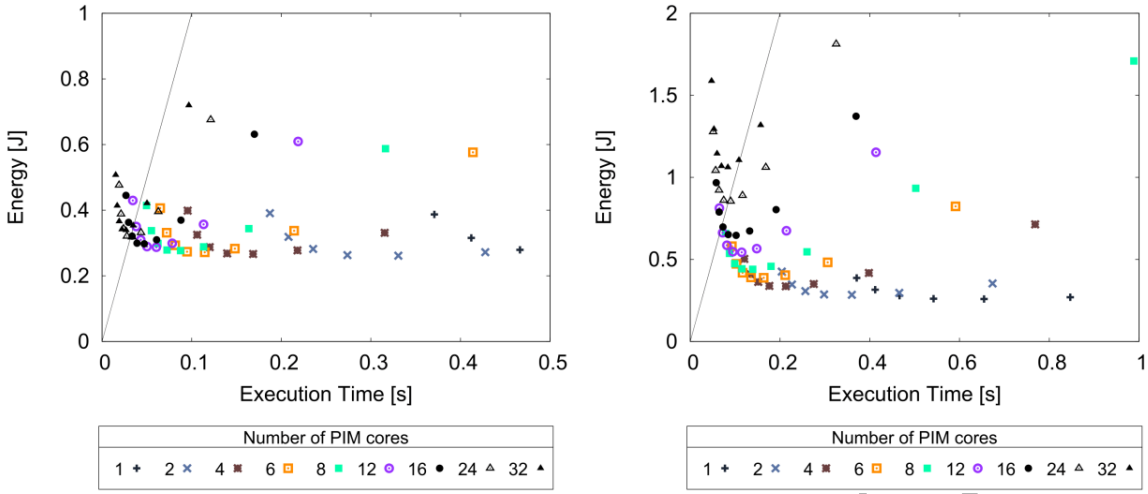


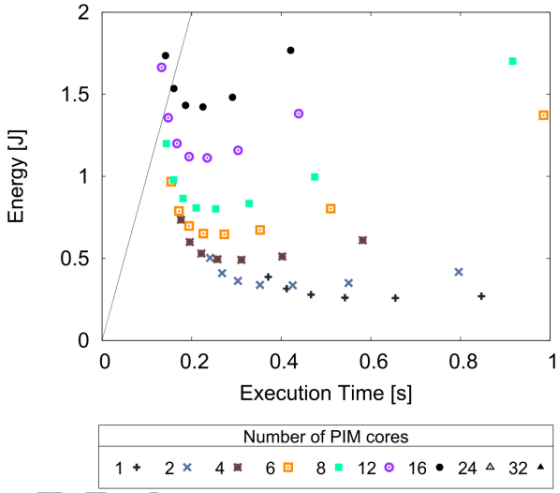
Figure 5- Maximum number of PIM cores we can place in the logic layer within a power budget of 10W. With a frequency of 800MHz and cache size 16KBwe can place up to 26 cores in the logic layer. Increasing the frequency to 1000MHZ and cache sizes to 64KB will limit us to 18 cores. This suggest a range of maximum 16-24 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.



a) Parallel overhead=1%

b) Parallel overhead=10%



c) Parallel overhead=30%

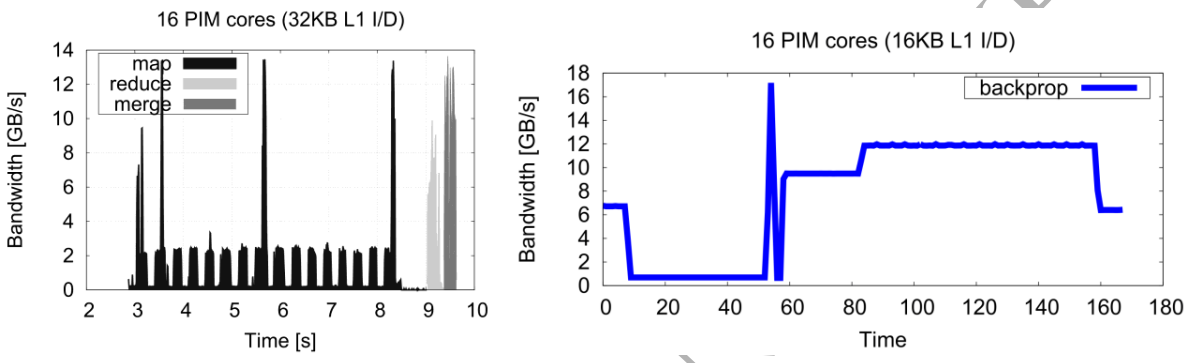
Figure 6 (a-c) - Time-Energy pairs for 3 different parallel overheads. The desirable number of PIM cores are those closer to the (0,0) coordinate. As the parallel overhead increases, the configurations with more cores "drift away" because more cores do not provide additional performance and increase 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

The obtained execution time for n cores, Total Execution Time(n), is then used to calculate the energy consumed by n cores, E(n).

$$E(n) = n * CPU\ Power * Exec.Time(n)$$

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 6 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. MapReduce applications are one set of applications which would benefit from larger number of PIM cores. If we assume that more general applications are going to run on PIM we might consider putting less cores and not waste additional energy.



a) wordcount bandwidth consumption **b) backprop bandwidth consumption**

Figure 7 - bandwidth consumption over time for wordcount(a) and backprop(b). Both workloads were running at f=1000MHz. Even when running on 16 cores the bandwidth utilization remains low. This raises concerns whether using ARM cores is the best option for PIMs.

10 Bandwidth and link power

Figure 7 shows the actual bandwidth consumption of wordcount (7a) and backprop (7b) when running on 16 PIM cores. None of the benchmarks exceed 20GB/s bandwidth consumption. This is also true for other workloads as well. Four PIM stacks, each with 16 PIM cores, would consume not more than 80GB/s which is relatively low to the total available bandwidth. However, the PIM stacks would still be more energy efficient then running the code on the host processor. This is due to the fact that the host processor would have to transfer the data over high speed SerDes links which are power hungry. HMCs provide memory bandwidth of up to 480GB/s [10]. The bandwidth is available to the host processor via 4 high speed SerDes links [4], with average energy consumption of 5-10pJ/bit. However, we can have low bandwidth links between the host processor and the PIMs and thereby reduce power consumption because we would offload the memory intensive work to the PIMs. The bandwidth utilized within each stack can be further increased by increasing the number of PIM cores per stack, however at the expense of higher power consumption and possibly lower utilization. To fully utilize the 480GB/s bandwidth, more than 250 PIM cores are needed, but then the power consumption will exceed the constraint of 10W TDP for the logic layer. This raises concerns about using ARM cores as PIM cores which we discuss in the next section.

11 PIM core alternative

One of the design points for a PIM system is deciding on the type of the processing core for the PIM. Loh et al. [3] presented couple of possibilities for the PIM cores including ASICs and fixed function PIMs. Our study, as well as some of the previous studies, explored the case where we would employ energy efficient low-power ARM cores as PIM cores. An advantage of having ARM cores as PIMs is the high energy efficiency. Designed for low power, ARM cores can still provide a significant performance for single threaded applications. An increased number of such cores can also provide a higher degree of parallelism and deliver a decent throughput as well. However, the amount of bandwidth consumed by such cores is very limited. A PIM stack with 16 ARM PIM cores does not consume more than 15GB/s when running MapReduce workloads, and not more than 20GB/s when running OpenMP benchmarks such as backprop. A system with 4 such PIM stacks would allow for an aggregate bandwidth of 4 x 480GB/s but only 80GB/s (aggregate) would be used on average. This raises concerns whether such cores are suitable for workloads which could potentially utilize the high in-stack available bandwidth. Thus, cores with high energy efficiency and high throughput, such as GPGPUs, would seem to benefit more from the available bandwidth. However, GPGPUs are suitable only in cases where applications have a high degree of parallelism. On the other hand ARM cores are more efficient in running single threaded code with more complex memory access patterns. Therefore it is still justifiable to explore ARM cores as an alternative to GPGPUs, specifically because we could offload memory intensive functions which are not easily translatable to GPGPU applications.

12 Conclusion

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 and OpenMP scientific benchmarks. Our study shows that a PIM core running at 800MHz clock frequency has the best energy efficiency. Smaller caches are more energy efficient for MapReduce workloads while more intensive scientific workloads benefit from larger caches. We have shown the power consumption components and calculated the maximum number of cores we can place in the logic layer. We observed that static power consumption, specifically from caches, dominates the power consumption. 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. We conclude that 16-24 ARM cores are a reasonable choice for PIMs utilizing ARM cores. One drawback of having ARM cores as PIMs is the low bandwidth utilization. Therefore high throughput processing cores, such as GPGPUs, would be better of utilizing the in-stack bandwidth and potentially prove even more energy efficient.

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)

29. Eckert, Y., Jayasena N., Loh, G.H.: Thermal Feasibility of Die-Stacked Processing in Memory, In the 2nd Workshop on Near-Data Processing (WoNDP), Cambridge, UK, December 2014 (Held in conjunction with MICRO'14)

30. Shuai Che; Boyer, M.; Jiayuan Meng; Tarjan, D.; Sheaffer, J.W.; Sang-Ha Lee; Skadron, K., "Rodinia: A benchmark suite for heterogeneous computing," in Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on , vol., no., pp.44-54, 4-6 Oct. 2009

BIOGRAPHIES

Marko Scrbak is a Ph. D. student and research assistant at University of North Texas. His research interests revolve around computer architecture, specifically Processing-in-Memory (PIM) systems and memory system design and optimizations. He joined the Computer Systems Research Laboratory (CSRL) at UNT in 2012, after receiving his Bachelor's degree from University of Zagreb, Croatia. He received his MS degree in Computer Science from University of North Texas in 2015.

Mahzabeen Islam is a Ph.D. student and research assistant at University of North Texas. Her research interest focuses on different aspects of computer memory systems, ranging from memory systems optimization and processing-in-memory to emerging memory technologies. She received her MS degree in Computer Science and Engineering from University of North Texas in 2015.

Dr. Krishna Kavi is currently a Professor of Computer Science and Engineering and the Director of the NSF Industry/University Cooperative Research Center for Net-Centric Software and Systems at the University of North Texas. During 2001-2009, he served as the Chair of the department. He also held an Endowed Chair Professorship in Computer Engineering at the University of Alabama in Huntsville, and served on the faculty of the University Texas at Arlington. He was a Scientific Program Manager at US National Science Foundation during 1993- 1995. He served on several editorial boards and program committees. His research is primarily on Computer Systems Architecture including multi-threaded and multicore processors, cache memories and hardware assisted memory managers. He also conducted research in the area of formal methods, parallel processing, and real-time systems. He published nearly 200 technical papers in these areas. He received more than US $6 M in research grants. He graduated 15 PhDs and more than 35 MS students. He received his PhD from Southern Methodist University in Dallas Texas and a BS in EE from the Indian Institute of Science in Bangalore, India.

Michael Ignatowski is an AMD Fellow since 2010. Prior to joining AMD, Mike was a Research Senior Technical Staff member at the IBM Watson Research Center. His worked on IBM Power and blade systems, systems and data center power efficiency management. At AMD he is focusing his research on advanced memory technologies and high performance computing. Mike received his BS in Physics from Michigan State University and a MS in in Computer Engineering the University of Michigan.

Dr. Nuwan Jayasena is a Principal Member of the Technical Staff at AMD Research. Previously he was with NVIDIA and Stream Processors Inc. His current research focus is on advanced memory systems including processing in memory and multi-level memories. He also contributed to AMD Fusion Systems Architecture and Heterogeneous Systems Architecture. He received his MS and PhD in Electrical Engineering from Stanford University and a BS in Computer Engineering from the University of Southern California.