Design space exploration for PIM architectures in 3D-stacked memories

João Paulo C. de Lima
Fed. University of Rio Grande do Sul
Porto Alegre, RS, Brazil

Paulo Cesar Santos
Fed. University of Rio Grande do Sul
Porto Alegre, RS, Brazil

Marco A. Z. Alves
Fed. University of Paraná
Curitiba, PR, Brazil

Antonio C. S. Beck
Fed. University of Rio Grande do Sul
Porto Alegre, RS, Brazil

Luigi Carro
Fed. University of Rio Grande do Sul
Porto Alegre, RS, Brazil

ABSTRACT

Scaling existing architectures to large-scale data-intensive applications is limited by energy and performance losses caused by off-chip memory communication and data movements in the cache hierarchy. Processing-in-Memory (PIM) has been recently revisited to address the issues of memory and power wall, mainly due to the maturity of 3D-stacking manufacturing technology and the increasing demand for bandwidth and parallel access in emerging data-centric applications. Recent studies have shown a wide variety of processing mechanisms to be placed in the logic layer of 3D-stacked memories, not to mention the already available 3D-stacked DRAMs, such as Micron's Hybrid Memory Cube (HMC). Nevertheless, a few studies compare PIM accelerators to each other and have made efforts to indicate the trade-offs between power, area, and performance. In this paper, we review different state-of-the-art 3D-stacked in-memory accelerators, and we analyze them considering important constraints regarding area and power due to critical embedded nature of PIM. Aiming to point in the direction of massive parallel PIM designs, we take the simplest design found in this survey, and we explore the architectural design space to meet the constraints imposed by HMC. Our results show that the most straightforward approach can provide the highest performance while consuming the lowest amount of area and power, which makes it the most suitable design found in this survey for an energy-efficient in-memory accelerator, whether it goes in High-Performance Computing or Embedded Systems. For instance, the outstanding point in the design space indicates that a performance density of 320 GBps/mm^2 and a performance efficiency of 0.6 GBps/mW can be achieved in the best scenario, that is, when a massive parallel application reaches the peak bandwidth.

CCS CONCEPTS

• **Hardware → Emerging architectures**;

KEYWORDS

Processing in Memory, Near Data Processing, Hybrid Memory Cube, 3D-stacked memories, vector processing, energy efficiency

ACM Reference Format:
João Paulo C. de Lima, Paulo Cesar Santos, Marco A. Z. Alves, Antonio C. S. Beck, and Luigi Carro. 2018. Design space exploration for PIM architectures in 3D-stacked memories. In *CF '18: CF '18: Computing Frontiers Conference, May 8–10, 2018, Ischia, Italy.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3203217.3203280

CF '18, May 8–10, 2018, Ischia, Italy

ACM ISBN 978-1-4503-5761-6/18/05…$15.00
https://doi.org/10.1145/3203217.3203280

1 INTRODUCTION

In the last decades, memory architectures have continually undergone several improvements in speed bandwidth, power efficiency, and latency. Nonetheless, improving process technology, increasing clock frequency, decreasing voltage has not been enough for breaking the walls (memory, bandwidth and power wall) which are the main bottleneck of conventional memory systems. As the gap between the technological basis of processors and memories increasingly widen, even more pressure on the memory system can be seen. Processor with a novel technology, for instance, can issue loads faster and more data and instructions are required at a higher bandwidth. This situation will cause more cache-lines to be moved through the cache hierarchy, potentially being replaced and underutilized, leading to a significant data-transfer overhead and demand for higher bandwidth [33].

Due to constraints compelled by the end of Dennard scaling [8] and the increasing demand for performance and exascale systems, computer architects are required to come up with new designs to extract performance where it provides more speed-up and consumes less energy. A viable approach to achieve such computing capacity consists of avoiding data movement by performing computation where the data resides, which is the main purpose of Processing-in-Memory (PIM) and Near-Data Processing (NDP) concept and specialization of processing units [33]. Moreover, 3D-stacking technologies open up opportunities for new designs to tackle memory wall problems. However, 3D-stacking PIM accelerators present a huge change to the current hardware architecture design and, consequently, reveal new challenges in how massive acceleration can be extracted.

The main problem addressed on this paper is related to an absence of discussion of costs in PIM logic designs. Recent studies have explored different organizations of processing elements, from programmable cores to custom logic [2, 12, 29, 36]. However, they can barely meet 3D-Dynamic Random Access Memory (DRAM)

constraints and match the high bandwidth provided by these memories. Given these issues, how do we find the best trade-off between area, power density, and processing capability for in-memory accelerators and which organization best suits PIM?

Observation 1) Power dissipation is still a limiting factor for stacking memory directly on top of high-performance processors since heat generated by CPU cores reduces retention time of data in DRAM dies [10, 34]. Although regular cores may not be beneficial for PIM architectures, many current studies utilize ARM cores [2, 5, 32] or lightweight alternatives [20] to build a massive parallel system.

Observation 2) Operations involving a large amount of data, such as gather/scatter, sorting, traversal, and data reorganization, may have significant improvement in terms of area, power and performance using simpler fixed-function PIM implementations in comparison with a fully-programmable PIM design [3, 32].

Observation 3) An in-memory vector unit has the potential to exploit the maximum internal memory bandwidth. Even for smaller vector operations, we can still take advantage of memory parallelism on distinct vector operations as independent Single Instruction Multiple Data (SIMD) issues [31].

Our goal is to evaluate the efficiency of general-purpose in-memory accelerators by reviewing state-of-the-art PIM designs and doing a theoretical comparison based on results extracted from them. In addition, this survey presents a design space exploration in a simple PIM device found in the related works to highlight some of the potential trade-offs of the different approaches for PIM accelerators. **Our Contribution** highlights advantages and drawbacks of state-of-the-art PIM designs, mainly establishing the relationship between bandwidth, area, and power, while comparing these designs to a SIMD/vector processor based on the work of [31].

In the following section, we present an overview of 3D high-densities memories and PIM architectures. Then, an evaluation of different PIM approaches are made in Section 3 and we discuss the main features of each design and their sources of performance and power efficiency improvement. From the related works, we choose the simplest design that presents the highest processing bandwidth and we explored the design space of in-memory vector unit by adjusting the Functional Unit (FU) design in Section 4. In section 5, we compare the efficiency metrics of each related work (area, power consumption, and bandwidth capabilities) using our fixed-function PIM from Section 4 as baseline.

2 BACKGROUND

In this section, we provide an overview of fundamental aspects of Commercial off-the-shelf (COTS) 3D-stacked memories and a description of Processing-in-Memory (PIM) architectures.

2.1 3D high-density memories

High-density 3D memories rely on multiple stacked DRAM dies to provide high bandwidth and capacity to meet the demand of today's system workloads. Most of the today's major memory manufacturers develop 3D-DRAMs, such as Samsung's DDR4, Tezzaron's DiRAM4, AMD and Hynix's High Bandwidth Memory (HBM) and Micron's Hybrid Memory Cube (HMC).

As described in the last specification [17], the HMC is a package containing either four or eight DRAM die and one logic die stacked together and connected by Through-Silicon Via (TSV), as shown in Figure 1. Within each cube, the memory is organized vertically into *vaults*, which consist of a group of corresponding memory portions from different DRAM dies combined with a *vault* controller within the logic die. Each HMC contains 32 *vaults* and each *vault* controller is functionally independent to operate upon 16 memory banks in the eight DRAM layers configuration. The available bandwidth from all *vaults* is up to 320 GBps and is accessible through multiple serial links, each with a default of 16 input lanes and 16 output lanes for full duplex operation. Moreover, the HMC specifies atomic requests commands which enable the logic layer to perform read-update-write operations atomically on data using 16-byte operands.

Due to independent access and simple processing capabilities in the logic layer, HMC is broadly chosen as target device in PIM architecture researches [2, 4, 5, 32].

2.2 PIM architectures

The main purpose of performing computation near the data resides is to take advantage of minimum latency and higher bandwidth available internally in the memory device. Although PIM in 3D-DRAM is the most popular form of integration, processing elements can also be placed in different levels of the memory hierarchy, such as cache [1] and storage memory [14], and combined with different memory technologies, e.g. phase-change memory and resistive RAM.

Several attempts of integrating processing logic at different locations within planar and stacked DRAM memory chips were studied (e.g. between the sense amplifiers and the column decoder, distributed in each *vault* of a 3D-stack chip, embedded logic die between DRAM dies) [33]. However, only with the emergence of 3D-stacking technology that architectural studies about 3D-stacked PIM have regained great interest. Most of the studies rely on general-purpose architectures, but a significant part of them propose application-specific and custom logic. In fact, [23] has categorized PIM architecture in three classes:

- **Non-compute Logic-in-Memory** - This class includes logic in memory which is responsible for software-transparent features, such as in-stack integration of memory controller and built-in self-test capabilities present in Micron's HMC [17].

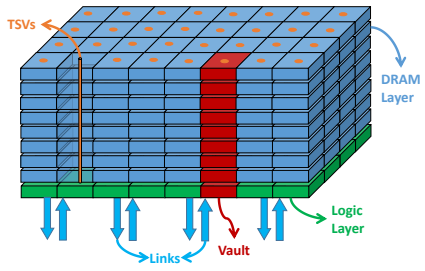

Figure 1: HMC layout comprises eight DRAM layers and a base logic layer connected by TSVs and vertically organized in *vaults*.

- **PIM with Fixed-function Operations** - This class provides pre-defined or fixed operations based on existing memory access instructions. Fixed-function PIM operations can be divided into Bounded-operand PIM operation (BPO), which comprises a single operation or a limited set of operations on single or multiple data, and Compound PIM operation (CPO), which includes a dynamic number of operations and arbitrary number memory location. [15, 21]
- **Fully programmable PIM** - The last class comprises full or simplified processors which fetch, decode and execute instructions from the code offloaded to the PIM accelerator.

3 ADDRESSING ON THE EFFICIENCY OF STATE-OF-THE-ART PIM ARCHITECTURES

There is an enormous space for design exploration in PIM architectures and several attempts have been made in the past years. Since 90's, projects such as [11, 19, 27] introduced PIM architectures that exploit lower access latency and higher bandwidth by moving processing logic closer to the memory. For many years, PIM architectures faced the lack of a viable manufacturing technology for processor logic and DRAM integration into a single chip as its main obstacle, and most of the projects made in that time were only for academic purpose. In the years of 2D-integrated research, most of the efforts showed that specialized architecture can reveal massive speed-ups. Computational RAM [11], Intelligent RAM [27] and [19] integrated processing elements to the memory die to perform Single Instruction Multiple Data (SIMD) streams, query processing, graph processing, and other irregular computations.

Since 2013, 3D-stacked PIM has regained attention and PIM projects have revisited different processing elements and organizations, from simple cores to Application Specific Integrated Circuits (ASICs) and reconfigurable blocks, to provide a balance of energy-efficiency, area-efficiency and flexibility for PIM accelerators. We limited our analysis to those papers that propose flexible PIM architecture integrated in logic layer of 3D-stacked memories. We considered several studies from 2014 to 2017 and we categorize them according to the main processing element which the logic layer is built upon: ARM cores [2, 5, 9, 29, 32], alternative cores [20, 25], SIMD units [26, 31], Graphics Processing Unit (GPU) [35] and Coarse-Grain Reconfigurable Array (CGRA) [12].

In past studies, [35] integrated Compute Units (CU) based on AMD's Graphics Core Next (GCN) architecture, and sized the number of CUs by establishing the power and area budget according to HMC constraints. The authors of [29] analyzed the impact of ARM cores in the logic layer for embarrassing parallel and largely localized memory access workloads. The Active Memory Cube, a vector architecture capable of performing floating-point, predicated operations and gather/scatter access across HMC, revealed huge performance for matrix multiplication and DAXPY kernels [25]. Tesseract is an accelerator for large-scale graph processing which places ARM core in each HMC *vault* [2].

A heterogeneous reconfigurable array for PIM systems, which combines both coarse-grain and fine-grain logic blocks, is presented by [12] and outperforms FPGA and CGRA-based PIM system on kernels of graph processing, MapReduce, and Deep Neural Networks. The authors of [26, 30, 31] proposed a fixed-function accelerator based on Intel's Advanced Vector Extensions (AVX) to perform data transfer, data reordering, logical, floating-point and integer operations, and evaluate the mechanism on processing query over column database, spiking neural networks and operations on linked data structures. The study proposed by [5] presented the Smart Cube Memory and a full system analysis, and evaluated gains of latency reduction on graph kernels.

The authors of [9] revealed huge performance speed-up on basic operators of data analytic processing by placing SIMD-enabled ARM cores on each HMC *vault*. The Harmonica core is a light-weight Single Instruction Multiple Thread (SIMT) core designed specifically for 3D-stacked DRAMs. [20] presented a description of Harmonica's micro-architecture, Instruction Set Architecture (ISA), and interface for thread creation, and evaluated the mechanism on several kernel applications as well. A design space exploration on ARM cores in the logic layer is presented by [32], where they established a power budget and varied frequency, cache size, number of cores and, finally, estimated its impact on MapReduce and OpenMP benchmarks. In the following sections, the approaches and results from previous state-of-art works are compared to each other in order to guide future designs.

3.1 Application and performance considerations

CPUs and GPUs have been continually evolving towards higher frequencies and number of Functional Units (FUs) because the source of performance improvement in traditional architectures is still based on processing bandwidth. Although recent memories cannot deliver data at the same rate CPUs and GPUs can consume, compute-bounded applications are mostly impacted by processing power and memory latency, rather than by memory bandwidth. On the other hand, the performance of memory-bounded applications depends more on memory bandwidth.

To demonstrate how processing bandwidth can determine the compute-bounded application performance, we take mean GFlops rate of two simple benchmarks running on GPUs. The first one is an SGEMM microbenchmark running on a GeForce GTX280. Increasing the order of the matrix to 4096 elements, we are able to saturate the processing bandwidth to 370 GFlop/s. Although GTX280 can theoretically provide 933 GFlops, the saturation bandwidth represents more than 10 times the memory bandwidth available on this device (141.7 GBps). The second example is an N-body benchmark, a simulation of a dynamical system of particles, from CUDA samples source code in a Titan Xp GPU. The N-body simulation can achieve up to 7904 GFlop/s, which represents 65% of the maximum processing bandwidth and more than 14 times the memory bandwidth available on Titan Xp.

In Section 2.1 we explained how 3D-DRAMs can provide a massive memory bandwidth. Assuming that the HMC architecture can deliver up to 10 GBps per vault, each Processing Unit (PU) needs to deliver more than 10 GBps to take full advantage of internal memory bandwidth. Moreover, considering that most of the data-intensive applications also present a little but representative amount of reuse, the processing power of each PU ideally must be greater than memory bandwidth as well.

Table 1

Reference	Technological process	Number of *vaults* per device	Number of cores	Logic description	Area (mm^2/*vault*)	Maximum bandwidth per *vault* (GBps)	Power (mW/vault)
A [35]	16nm	16	12	AMD's Compute Unit (CU) @650MHz	2.5	120*	625
B [29]	32nm	16	16	ARM Cortex-A5 @1GHz	0.51	8*	80
C [25]	14nm	32	32	4-lane VLIW-based @1.25 GHz	0.8#	40	312
D [2]	28nm	32	32	ARM Cortex-A5 + FPU @2GHz	0.68	16*	320
E [12]	45nm	8	n/a	HRL @200MHz	6	51.2	625
F [26, 30, 31]	32nm	32	32	Reconfigurable Vector Unit (RVU) @1GHz	n/a	320	206
G [5]	28nm	16	1	OoO ARM Cortex-A15 @1GHz	4.13	1	37.5
H [9]	28nm	32	32	ARM Cortex-A35 SIMD 1024-bit @1GHz	1.15	128	180
I [20]	15nm	32	32	Harmonica (SIMT) @650MHz	1.5	12	300
J [32]	40nm	16	16	ARM Cortex-A5 @1.4GHz	0.51	11.2*	625

* Estimated values based on ARM Cortex-A5 and -A15 data-sheets. # Estimated area.

A typical low-power Out-of-Order (OoO) core can achieve a peak bandwidth much lower than each HMC *vault* is capable of providing, which end up being a limiting factor to extract memory parallelism in PIM architectures. A common ARM Cortex-A57 @2GHz, for instance, can read up to 5.3 GBps in an ideal case [9]. In addition, the results of [29] cope with such statement: data-intensive applications, such as MapReduce kernel, are limited to achieve a peak read bandwidth of approximately 3.4 TBps because the processing power is limited to 4 TBps (512 PIM cores), despite the environment were capable of providing 8 TBps (32 HMC). MapReduce kernels are a great example of how applications embarrassing parallel and mostly memory-bounded can be harmed by CPU designs that cannot match 3D-DRAMs bandwidth.

In terms of bandwidth and performance, the designs proposed by [9, 12, 25, 31, 35] present the highest streaming bandwidth among the studies listed in Table 1. The seventh column in Table 1 represents the maximum theoretical bandwidth that can consume from each *vault*. Coupled with SIMD-enabled operations, high streaming bandwidth is widely known as the main source of performance improvement for data-intensive applications. Although each design in Table 1 uses different benchmarks, they provide an evidence that processor bandwidth is a limiting factor to speed-up most of the data-intensive applications in PIM architectures.

3.2 Area and power considerations

One of the main problems of manufacturing 3D-stacked PIM accelerators integrated to is to meet 3D-DRAM constraints. 3D-stacked PIM must be: *area-efficient*, i.e. it must provide compute throughput to match high bandwidth available in HMC, and *power-efficient*, i.e. it must reduce total energy consumption and avoid causing thermal issues in DRAM stacks [12].

The first generation of HMC devices, which comprises 1Gb distributed in 4 DRAM layers (50nm) and a logic layer (90nm), consumes 11W [28]. On the other hand, [10] discuss the thermal feasibility of 3D-stacked memories logic die and find a maximum of 8.5W of power consumption in the logic layer considering a low-cost cooling mechanism. Based on the power budget of 11W presented by [28], several designs [2, 12, 32, 35] define the maximum number of cores by this power budget (respectively 625mW and 312mW per *vault* for the first and second generation of HMC), although 11W can represent an increment of 100% in original HMC power. Area constraints are discussed in Section 4 since we estimated the area available on HMC by downscaling to more recent technological process and also by scaling to higher capacity memories.

Table 1 also summarizes the area and power consumption of PU normalized to each *vault*. From this Table, one can observe that designs with simple FUs capable of performing large vector operations can provide a better compromise between area, power and bandwidth per *vault*. The data presented in Table 1 will be used later in Section 5 to analyze the efficiency of each design and compare them with the results obtained in the design space exploration in Section 4.

4 DESIGN SPACE EXPLORATION FOR IN-MEMORY VECTOR UNIT

In Section 3 we conclude that fixed PIM approach is the most efficient. In this way, among several approaches and to better illustrate our conclusion we targeted on redesigning the mechanism presented by [31] without harming its performance.

First, we chose to use a traditional set of adders, multipliers, and multiplexers to implement the FUs that allows the SIMD operations proposed by [26, 31]. Furthermore, we take into account the different operations and instructions needed to execute the applications that authors considered among their work presented on [26, 30, 31]. Moreover, authors claimed that their approach is capable of parallel compute up to 64x 32bits or 32x 64bits operations per HMC *vault*. Also, considering that a FU of 64bits is able to compute 32bits operations, another 32 FUs of 32bits need to be available in order to make possible the 64x 32bits parallel operations.

Therefore, we implemented a vector unit based on traditional units containing 32x32bits and 32x64bits, integer and float-pointing FUs (adders and multipliers), an 8x32x64bits register file, and a Finit State Machine (FSM) able to represent a single RVU instance. Supported by Cadence RTL Compiler tool, we extracted area, power consumption, and operating frequency for our implementation in 32nm process technology. Table 2 summarizes the synthesis results of traditional FUs for different frequencies. From this table it is possible to notice that the power consumption reaches 2.3W for a single *vault*, which is considerably high since our case study [31] claims 206mW per *vault*. It means that a different organization of FUs is required.

Although the characteristics of HMC make its logic layer suitable for PIM mechanisms, the design of this layer must meet the constraints given by DRAM layers. According to [18], a design comprising of 1Gb DRAM cells per layer distributed along 16 *vaults* takes 68mm^2 using 50nm of technological process. Hence, theoretically up to 68mm^2 of circuit logic can be supported by this layer. Moreover, other studies also found area results ranging from 3.5

Table 2: Per *vault* grouped set of traditional FUs 32x64bits + 32x32bits adders and multipliers

Frequency(MHz)	Bandwidth (GB/s)	Area(mm^2)	Power(mW)
250	64	0.703	350
500	128	0.916	983
750	192	1.123	1770
1000	256	1.210	1969
1250	320	1.351	2360

to 4.4mm^2 [9, 13] per *vault* for 1Gb DRAM cells and 16 *vaults*. Although we chose an 8GB, 8 layers, 32 *vaults* HMC configuration for our evaluation which would increase the total area, we also chose to downscale the technological processing to 32nm. We use CACTI-3D [7] for estimating the area of 8GB HMC, which results in 144.13mm^2. The configuration has 32 *vaults* which provides an area of 4.5mm^2 per *vault* where the FUs must fit in the available logical layer.

One can observe from Table 2 that for an operating frequency of 1GHz, 1.2mm^2 is occupied by the 64 FUs while achieving near 2W of power consumption. Despite the area can fit on the available logical space of 4.5mm^2, the power consumption for each *vault* can increase 6 times the total power of HMC [18], while resulting in more than 64W for all 32 *vaults*. In this way, although the fixed PIM using huge vector approach has proved to be more efficient in theory, a new approach in terms of efficient FUs design urges. Therefore, we focus on reducing the power consumption of the required set of FUs.

4.1 Architecture Model Definition

Concerned about the power required by the simple FUs approach, this section presents a design able to maintain the same bandwidth available on [31], as presented on Table 2, while it reduces the power consumption with no performance harm.

Traditionally, FUs of 64bits are able to operate over operands of 32bits, as aforementioned. In our design, we go further allowing each 64bits unit capable of operating on 1 operand of 64bits, 2 operands of 32bits, or 4 operands of 16 bits in parallel. Furthermore, by adopting this design we were able to reduce the needed amount of units, which means that instead of using a total of 64 units per *vault* (32x 64bits + 32x 32bits) we are able to use 32 units of 64bits

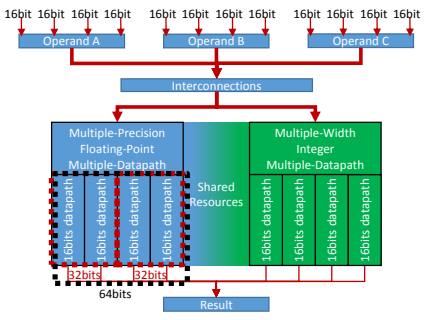

Figure 2: Multiple-Width Multiple-Datapaths Fused Integer/Floating Point Functional Unit.

capable of computing operations over 32x 64bits, 64x 32bits or 128x 16bits.

In order to implement the proposed design, we based our work on [6, 16, 24]. In [16] is shown a multi-precision floating-point multiply-add fused unit able to share resources between single and double precision floating-point operands, while the work presented in [24] is able to share resources for single, double and quad precision floating-points. The work presented in [6] shows a way to share resources between integer and floating-point operands. By sharing hardware, these works are able to reduce area and power consumption. Our work focus on supporting half, single, and double precision floating point numbers, and also integer operands of 16bits, 32bits, and 64bits. In this way, our contribution mixes previously explained ideas while splitting the adders and multipliers to allow its parallel utilization for different operand sizes. Figure 2 illustrates the proposed hardware organization.

We chose to implement a 16bits datapath for integer and floating-point operations. Initially, these datapaths share adders and multipliers in a similar way as presented in [6]. After that we integrate 2 and 4 datapaths, which allows computing 32 and 64 bits respectively, inspired by [16, 24]. In this way, to reproduce the work of reference presented in [31], 32 FUs are grouped as illustrated on Figure 3. The design presented shows 32 register-files and 32 FUs connected by an interconnection network. Each register-file has 512bits of storage capacity, and it can be accessed in portions of 16, 32, or 64bits, as shown in Figure 3. Also, the design shows that each register file and its respective FU can be accessed independently, which allows up to 32 scalar operations. Triggering various FUs it is possible to exploit SIMD operations. According to [31] its instructions is based on AVX-512 instruction set, which requires the support of complex data movement SIMD instructions, such as broadcast and permutation.

We synthesize the set of FUs using a 32nm technological node and the same operating frequencies presented in Table 2. The synthesis results for our design is presented in Table 3. It is important to notice that although the frequencies of our designs are increased linearly (250MHz), area and power does not follow the same pattern because there are different options and pre-defined cells for different frequencies in the 32nm library.

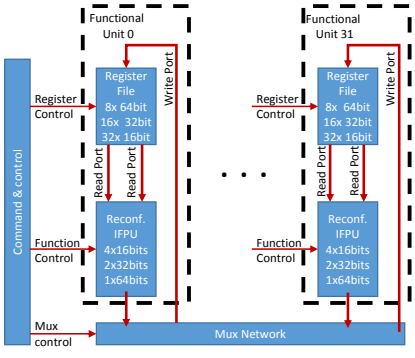

Figure 3: Grouped Multiple Functional Units Multiple-width Register Files - Multiple-Precision Integer/FP Units

Table 3: Per *vault* grouped set of optimized FUs Multiple-Precision FP/Integer 32x64bits

Frequency(MHz)	Bandwidth (GB/s)	Area(mm^2)	Power(mW)
250	64	0.502	119.44
500	128	0.621	258.30
750	192	1.198	400.57
1000	256	1.269	585.50
1250	320	1.302	703.42

The main gain from our design is related to power consumption, for example, it is possible to notice that using traditional FUs 1.9mW is achieved for 1GHz of frequency while using multiple-precision approach 585mW is possible at 1GHz. However, despite our efforts we are not able to achieve the power consumption of 206mW per *vault* as claimed in [31]. Therefore, the next section will use the results from Table 3 in order to compare several design approaches.

5 COMPARISON BETWEEN LOGIC ALTERNATIVES

The following subsections depict performance, area and power comparisons with the results presented in Section 3. As listed in Table 1, the designs were implemented using different technological processes and it requires a technology scaling for a proper comparison. Thus, we scaled all designs to 28nm process, as most of the related works were implemented in this technological node. We also approximated area and power results using similar works, data-sheets, or by using McPat [22] for those related works that did not present such values.

As the configuration of each related work varies in the number of *vaults* and cores, we normalized all designs to 32 *vaults* to provide results that can be directly compared to each other. The main idea is to seek the highest performance while comparing area and power. We consider the maximum theoretical bandwidth of the target memory as 320 GBps (10 GBps/*vault*). In the following subsections, we also use the term *bandwidth* to refer to the processing capability of a Processing-in-Memory (PIM) processing unit, and not only to its capacity to access the memory. In this section we referred to designs presented in Table 1 by their letter indexes. The results for implementation ***F*** were obtained from Table 3, which was discussed in Section 4.

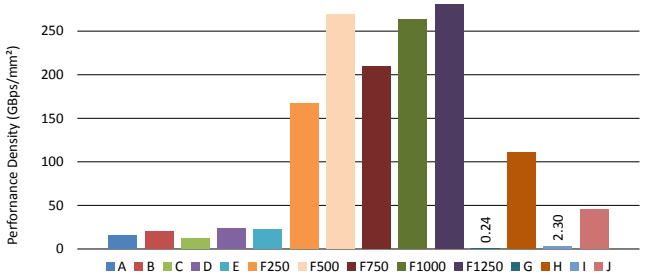

Figure 4: Performance Density - GBps/mm^2 All designs scaled to 28nm

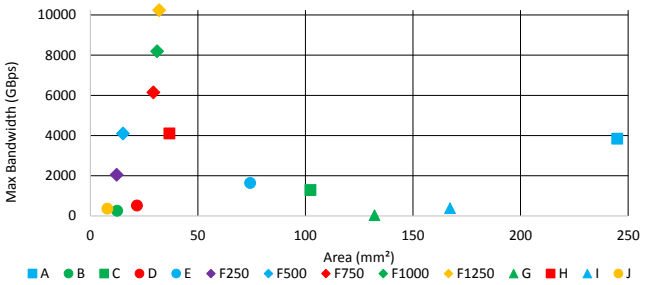

Figure 5: Maximum Bandwidth vs Required Area All designs Normalized to 32 *vaults*

5.1 Area breakdown

The area is critical for PIM architectures, as it is expected that these logical designs fit within the logical layer available in 3D-stacked memories. Figure 4 shows the performance density for all designs presented in Table 1. Performance density was computed as bandwidth per area (used by processing logic), or GBps/mm^2. According to Table 1, the designs ***F*** and ***H*** present the highest theoretical processing bandwidth per *vault*, respectively, 320 GBps and 128 GBps.

In Figure 5, the designs were normalized to 32 *vaults* in order to illustrate the highest performance and the required area for each implementation while keeping an equivalent system organization. From Figure 5, one can also restate that in-order ARM cores have reduced cost in area, but lack of capabilities to provide high processing bandwidth, except for ***H*** which incorporates large Single Instruction Multiple Data (SIMD) in its design. These results show the efficiency of ***F*** approach in terms of performance density.

5.2 Power breakdown

Power consumption is another important challenge to accommodate logical hardware inside 3D-stacked memories. In Figure 6 we show the performance efficiency for each design, which is calculated by relating the bandwidth (GBps) and power consumption (mW). Despite design ***H*** is able to achieve the highest performance efficiency, ***H*** can only compute fixed point operations, while the others are able to operate on floating-point representations. As pointed in [9], ***H*** efficiency could be reduced by approximately 8× if floating-point were supported. Nevertheless, the designs ***F***

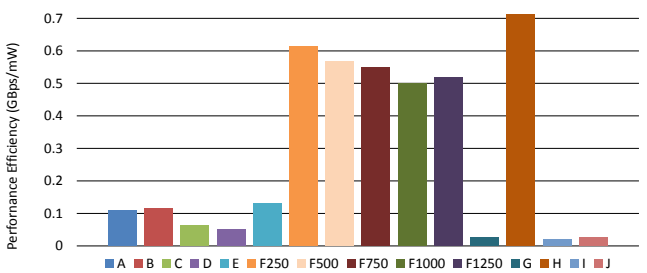

Figure 6: Performance Efficiency - GBps/mW All designs scaled to 28nm

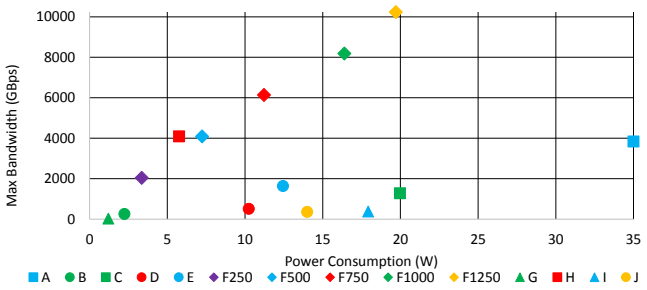

Figure 7: Maximum Bandwidth vs Required Power
All designs normalized to 32 *vaults*

are able to operate on multi-precision floating-point, achieving a performance efficiency between 0.5 and 0.6 GBps/mW for their variations.

Similar to what happens in Section 5.1, fully programmable PIM cores also consume extra power to keep their complex mechanism mostly when out-of-order execution model and larger caches exist. For instance, the lowest performance efficiency values are found in designs ***G***, ***I*** and ***J***, which respectively correspond to an Out-of-Order (OoO) core, Single Instruction Multiple Thread (SIMT) core, and an in-order core with large caches and higher clock frequency. Therefore, these cores cannot achieve the required bandwidth without spending a higher amount of power. Figure 7 illustrates the required power for each design to achieve the theoretical maximum bandwidth for 32 *vaults*. One can observe the similarities between ***H*** and ***F500***: both achieve the same bandwidth with practically the same total power consumption, as both lies on large SIMD Functional Units (FUs).

5.3 Matching the Performance and Area limits

In order to complete our analysis, in this section, we aim at equating the performance for all designs presented in Table 1. For this, we extrapolate the respective number of cores of each design in order to achieve the same performance of ***F1250*** (320 GBps per *vault*). Table 4 presents absolute estimated values of area, power, and number of cores required to achieve the target performance. From Table 4, it is important to notice that most of the designs that use typical ARM cores cannot be placed in the area and power budget, respectively discussed in Section 4 and 3.2 available on Hybrid Memory Cube (HMC) devices, requiring from 80 (***H***) to 1280 (***B***) cores to achieve a maximum bandwidth of 320 GBps. Despite designs ***F*** use less area, our estimation indicates that design ***H*** is able to consume slightly less power than designs ***F*** in this extrapolation. Considering the absence of floating-point units in design ***H***, one can foresee that ***F*** consume the lowest amount of power given the features provided.

5.4 Limits of parallelism and execution model

High-performance computing architectures have historically relied on two approaches to extract parallelism in a single thread: Instruction Level Parallelism (ILP) and Data Level Parallelism (DLP) paradigms. Both paradigms have limits dictated by data dependency, which is particular to each application. Related work based on ARM core [2, 5, 9, 29, 32] are generally based on ILP, except for the [9] that can also exploit DLP. ARM cores can exploit parallelism in multiple threads without changing the current software infrastructure and rely on a mature programming model, such as OpenMP.

The designs presented by [20] and [35] combine Thread Level Parallelism (TLP) and are supported by mature software infrastructures such as CUDA and OpenCL. Such parallelism is found in most of today's throughput-hungry computing applications. However, those architectures are not suitable for single-thread workloads.

The work of [31] combines both ILP and DLP paradigms using reconfigurable vector units and multiple operand sizes. In the mechanism proposed by [31], each vault has its own instruction queue and its own SIMD FU, which allows the exploitation of massive parallel ILP and DLP, respectively. The programming model is based on compiler decision. However, compiler and the cycle-accurate simulator are nearing completion, and the architecture has open issues on simultaneous multi-thread execution.

The VLIW approach [25] presents different ways of extracting parallelism, such as ILP, multithreading, vector and spatial SIMD. Moreover, software infrastructure was designed, and programming model is based on OpenMP 4.0. On the other hand, the parallelism in the CGRA approach [12] is inherited from CGRA/FPGA automation flow. The software stack divides an application into coarse-grained phases called kernels, which are translated using high-level synthesis into kernel circuits that exploit DLP.

6 CONCLUSIONS AND FUTURE WORK

In this survey, we evaluate different state-of-the-art 3D-stacked in-memory accelerators, and we compare them against the simplest design found in the literature.

We show that many PIM architectures reuse existing low-power and high-performance processors used in traditional architectures. We revealed that several studies size the number of cores according to a power budget that is difficult to achieve using simple cooling mechanisms in a HMC device. In general, placing full processors into a single logic layer introduces a significant area and power overhead, and more importantly, it cannot fully utilize the bandwidth provided by high-density memories in realistic designs.

We also show how the issuing rate of load/store unit of such processors can be a bottleneck for applications considered compute-bounded and mostly memory-bounded, such as MapReduce, in specific environments. This situation happens when the memory bandwidth is higher than the processing power, which is the case of many architectures that use low-power processors and HMC. Placing full processors into the logic layer of HMC in a traditional fashion can take advantage of existing compilers, programming models, and offloading mechanisms. Also, sharing the same Instruction Set Architecture (ISA) enables the PIM accelerator to execute unmodified host code. However, to fit some full processors into a single logic layer, area and power are increased, and it leads to performance overhead in comparison to fixed-function PIM accelerators.

The design presented in the Section 4, which is a fixed-function PIM based on [31], provides the highest processing bandwidth, although it depends on a host processor to issue SIMD operations.

Table 4: Total Area, Power and Number of Cores needed to achieve 10TBps of processing power (320GBps per *vault*)

	A	B	C	D	E	F250	F500	F750	F1000	F1250	G	H	I	J
Total Area (mm^2)	653.3	499.8	819.2	435.2	464.5	61.4	38.1	48.9	38.8	31.9	42291.2	92	4460.0	228.48
Total Power (W)	93.2	89.6	159.7	204.8	77.7	16.7	18.1	18.7	20.4	19.6	384.1	14.4	477.8	400
Number of Cores	64	1280	256	640	n/a	160	80	53.33	40	32	640	80	853.3	914.3

As 3D-stacked PIM accelerators need to meet power and area constraints, we explored the design space of in-memory vector unit by adjusting the FU design.

In future works, we intend to evaluate applications from different domains and estimate the energy behavior of each type of PIM accelerator. Also, we plan to analyze multithread behavior on different PIM approaches.

REFERENCES

[1] Shaizeen Aga, Supreet Jeloka, Arun Subramaniyan, Satish Narayanasamy, David Blaauw, and Reetuparna Das. 2017. Compute Caches. In *Int. Symp. on High Performance Computer Architecture (HPCA)*. IEEE.

[2] Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi. 2015. A scalable processing-in-memory accelerator for parallel graph processing. In *Int. Symp. on Computer Architecture (ISCA)*. IEEE.

[3] Berkin Akin, Franz Franchetti, and James C Hoe. 2015. Data reorganization in memory using 3D-stacked DRAM. In *Computer Architecture (ISCA), 2015 ACM/IEEE 42nd Annual International Symposium on*. IEEE, 131–143.

[4] Marco AZ Alves, Matthias Diener, Paulo C Santos, and Luigi Carro. 2016. Large vector extensions inside the HMC. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE.

[5] Erfan Azarkhish, Davide Rossi, Igor Loi, and Luca Benini. 2016. Design and evaluation of a processing-in-memory architecture for the smart memory cube. In *Int. Conf. on Architecture of Computing Systems (ARCS)*. Springer.

[6] Tom M. Bruintjes, Karel H. G. Walters, Sabih H. Gerez, Bert Molenkamp, and Gerard J. M. Smit. 2012. Sabrewing: A Lightweight Architecture for Combined Floating-point and Integer Arithmetic. *ACM Trans. Architecture and Code Optimization* 8, 4 (Jan. 2012).

[7] Ke Chen, Sheng Li, Naveen Muralimanohar, Jung Ho Ahn, Jay B. Brockman, and Norman P. Jouppi. 2012. CACTI-3DD: Architecture-level Modeling for 3D Die-stacked DRAM Main Memory. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*.

[8] Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. 1974. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits* 9, 5 (1974).

[9] Mario Drumond, Alexandros Daglis, Nooshin Mirzadeh, Dmitrii Ustiugov, Javier Picorel, Babak Falsafi, Boris Grot, and Dionisios Pnevmatikatos. 2017. The mondrian data engine. In *Int. Symp. on Computer Architecture*. ACM.

[10] Yasuko Eckert, Nuwan Jayasena, and Gabriel H. Loh. 2014. Thermal Feasibility of Die-Stacked Processing in Memory. In *2nd Workshop on Near-Data Processing (WoNDP)*.

[11] Duncan G Elliott, Michael Stumm, W Martin Snelgrove, Christian Cojocaru, and Robert McKenzie. 1999. Computational RAM: Implementing processors in memory. *IEEE Design & Test of Computers* 16 (1999).

[12] Mingyu Gao and Christos Kozyrakis. 2016. HRL: efficient and flexible reconfigurable logic for near-data processing. In *Int. Symp. High Performance Computer Architecture (HPCA)*.

[13] Mingyu Gao, Jing Pu, Xuan Yang, Mark Horowitz, and Christos Kozyrakis. 2017. Tetris: Scalable and efficient neural network acceleration with 3d memory. In *Int. Conf. on Architectural Support for Programming Languages and Operating Systems*.

[14] Boncheol Gu, Andre S Yoon, Duck-Ho Bae, Insoon Jo, Jinyoung Lee, Jonghyun Yoon, Jeong-Uk Kang, Moonsang Kwon, Chanho Yoon, Sangyeun Cho, et al. 2016. Biscuit: A framework for near-data processing of big data workloads. In *Int. Symp. on Computer Architecture (ISCA)*. IEEE.

[15] Byungchul Hong, Gwangsun Kim, Jung Ho Ahn, Yongkee Kwon, Hongsik Kim, and John Kim. 2016. Accelerating linked-list traversal through near-data processing. In *Int. Conf. on Parallel Architecture and Compilation Techniques (PACT)*.

[16] L. Huang, L. Shen, K. Dai, and Z. Wang. 2007. A New Architecture For Multiple-Precision Floating-Point Multiply-Add Fused Unit Design. In *Symposium on Computer Arithmetic (ARITH), 2007*.

[17] Hybrid Memory Cube Consortium. 2013. Hybrid Memory Cube Specification Rev. 2.0. (2013). http://www.hybridmemorycube.org/.

[18] Joe Jeddeloh and Brent Keeth. 2012. Hybrid memory cube new DRAM architecture increases density and performance. In *Int. Symp. on VLSI Technology (VLSIT)*.

[19] Yi Kang, Wei Huang, Seung-Moon Yoo, D. Keen, Zhenzhou Ge, V. Lam, P. Pattnaik, and J. Torrellas. 1999. FlexRAM: toward an advanced intelligent memory system. In *Int. Conf. on Computer Design: VLSI in Computers and Processors*.

[20] Chad D Kersey, Hyesoon Kim, and Sudhakar Yalamanchili. 2017. Lightweight SIMT core designs for intelligent 3D stacked DRAM. In *Int. Symp. on Memory Systems (MEMSYS)*. ACM.

[21] Duckhwan Kim, Jaeha Kung, Sek Chai, Sudhakar Yalamanchili, and Saibal Mukhopadhyay. 2016. Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory. In *Int. Symp. on Computer Architecture (ISCA)*. IEEE.

[22] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. 2009. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Int. Symp. on Microarchitecture (MICRO-42)*. IEEE/ACM.

[23] Gabriel H Loh, Nuwan Jayasena, M Oskin, Mark Nutter, David Roberts, Mitesh Meswani, D Ping Zhang, and Mike Ignatowski. 2013. A processing in memory taxonomy and a case for studying fixed-function pim. In *Workshop on Near-Data Processing*.

[24] K. Manolopoulos, D. Reisis, and V.A. Chouliaras. 2016. An Efficient Multiple Precision Floating-point Multiply-Add Fused Unit. *Journal of Microelectronic* 49 (2016).

[25] Ravi Nair, Samuel F Antao, Carlo Bertolli, Pradip Bose, Jose R Brunheroto, Tong Chen, C-Y Cher, Carlos HA Costa, Jun Doi, Constantinos Evangelinos, et al. 2015. Active memory cube: A processing-in-memory architecture for exascale systems. *IBM Journal of Research and Development* 59 (2015).

[26] Geraldo F. Oliveira, Paulo C. Santos, Marco A.Z. Alves, and Luigi Carro. 2017. NIM: An HMC-Based Machine for Neuron Computation. In *Int. Symp. on Applied Reconfigurable Computing (ARC)*. Springer.

[27] David Patterson, Thomas Anderson, Neal Cardwell, Richard Fromm, Kimberly Keeton, Christoforos Kozyrakis, Randi Thomas, and Katherine Yelick. 1997. A case for intelligent RAM. *IEEE micro* 17 (1997).

[28] J Thomas Pawlowski. 2011. Hybrid memory cube (HMC). In *Hot Chips 23 Symposium (HCS)*. IEEE.

[29] Seth H Pugsley, Jeffrey Jestes, Huihui Zhang, Rajeev Balasubramonian, Vijayalakshmi Srinivasan, Alper Buyuktosunoglu, Al Davis, and Feifei Li. 2014. NDC: Analyzing the impact of 3D-stacked memory+ logic devices on MapReduce workloads. In *Int. Symp. on Performance Analysis of Systems and Software (ISPASS)*.

[30] Paulo C. Santos, Geraldo F. Oliveira, Joao P. Lima, Marco A.Z. Alves, Luigi Carro, and Antonio C.S. Beck. 2018. Processing in 3D memories to speed up operations on complex data structures. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE.

[31] Paulo C. Santos, Geraldo F. Oliveira, Diego G. Tomé, Marco A.Z. Alves, Eduardo C. Almeida, and Luigi Carro. 2017. Operand size reconfiguration for big data processing in memory. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE.

[32] Marko Scrbak, Mahzabeen Islam, Krishna M Kavi, Mike Ignatowski, and Nuwan Jayasena. 2017. Exploring the Processing-in-Memory design space. *Journal of Systems Architecture* 75 (2017).

[33] Patrick Siegl, Rainer Buchty, and Mladen Berekovic. 2016. Data-Centric Computing Frontiers: A Survey On Processing-In-Memory. In *Int. Symp. on Memory Systems (MEMSYS)*. ACM.

[34] Christian Weis, Matthias Jung, Peter Ehses, Cristiano Santos, Pascal Vivet, Sven Goossens, Martijn Koedam, and Norbert Wehn. 2015. Retention time measurements and modelling of bit error rates of WIDE I/O DRAM in MPSoCs. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE.

[35] Dongping Zhang, Nuwan Jayasena, Alexander Lyashevsky, Joseph L Greathouse, Lifan Xu, and Michael Ignatowski. 2014. TOP-PIM: throughput-oriented programmable processing in memory. In *Int. Symp. on High-performance Parallel and Distributed Computing*. ACM.

[36] Qiuling Zhu, Berkin Akin, H Ekin Sumbul, Fazle Sadi, James C Hoe, Larry Pileggi, and Franz Franchetti. 2013. A 3D-stacked logic-in-memory accelerator for application-specific data intensive computing. In *Int. 3D Systems Integration Conference (3DIC)*. IEEE.