Modeling and Benchmarking Computing-in-Memory for Design Space Exploration

Dayane Reis
University of Notre Dame
Notre Dame, IN, USA
dreis@nd.edu

Di Gao
Zhejiang University
Hangzhou, China
digao@zju.edu.cn

Shaahin Angizi
Arizona State University
Tempe, AZ, USA
sangizi@asu.edu

Xunzhao Yin
Zhejiang University
Hangzhou, China
xzyin1@zju.edu.cn

Deliang Fan
Arizona State University
Tempe, AZ, USA
dfan@asu.edu

Michael Niemier
University of Notre Dame
Notre Dame, IN, USA
mniemier@nd.edu

Cheng Zhuo
Zhejiang University
Hangzhou, China
czhuo@zju.edu.cn

X. Sharon Hu
University of Notre Dame
Notre Dame, IN, USA
shu@nd.edu

ABSTRACT

The bottleneck between the limited memory bandwidth and high speed processing demands is the main cause of problems associated with high volume of data transfers in data-intensive applications. As a possible remedy to these issues, computing-in-memory (CiM) enables a subset of logic and arithmetic operations to be performed where the data resides, i.e., inside the memory. Various CiM designs have been proposed to date, based on different technologies. Given the variety of options available, picking the right design option for a system/application can be a complex task. When choosing a CiM design, it is important to establish evaluation conditions that are as uniform as possible to make a fair choice between available design options. In this paper, we describe a methodology for an *uniform* benchmarking of CiM designs. Our approach evaluates devices/circuits, arrays and the overall impact of CiM to a system with a framework based on Eva-CiM. As a case study, we analyze the array-level performance of 7 recent CiM designs implemented with SRAM, DRAM, FeFET-RAM, STT-MRAM, SOT-MRAM, and RRAM. After we identify that the FeFET-RAM-based design shows promising energy and delay savings at the array level, we carry out a system level evaluation showing that FeFET-RAM-based CiM outperforms a CMOS SRAM CiM baseline by an average of 60% across a set of 17 benchmarks (with respect to energy savings). Regarding speedups, both technologies offer virtually the same benefit of about ~1.5× when compared to a situation where processing does not happen in memory.

CCS CONCEPTS

• **Hardware → Emerging architectures**; **Emerging tools and methodologies**;

KEYWORDS

Computing-in-memory, benchmarking, emerging technologies

ACM Reference Format:
Dayane Reis, Di Gao, Shaahin Angizi, Xunzhao Yin, Deliang Fan, Michael Niemier, Cheng Zhuo, and X. Sharon Hu. 2020. Modeling and Benchmarking Computing-in-Memory for Design Space Exploration. In *Proceedings of the Great Lakes Symposium on VLSI 2020 (GLSVLSI '20), September 7–9, 2020, Virtual Event, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3386263.3407580

1 INTRODUCTION

The rise of Big Data has brought data-intensive applications to prominence — these applications demand an ever high volume of information storage and processing. Data-intensive applications are challenging to address with von Neumann architectures [1]. More precisely, data movement between memory and processing units may lead computing systems to hit a "memory wall" where the overall system performance is limited by the memory bandwidth rather than the central processing unit (CPU) speed [2]. Moreover, larger latencies associated with high volumes of data transfers result in higher energy consumption overhead. Computing-in-memory (CiM) is a potential solution to the problems associated with the "memory wall" in von Neumann architectures. CiM enables a subset of logic and arithmetic operations to be performed where the data resides, i.e., inside the memory. Various CiM designs for on-chip memories (i.e., CPU caches) have been proposed to date, e.g., [3–6], which can provide support to different data-intensive applications.

To this end, CPU caches are often based on complementary metal–oxide–semiconductor (CMOS) technology. CMOS caches usually employ 6T- static random-access memory (SRAM) cells, which have notably low density and high leakage power, further aggravating the problem of high energy consumption in modern computer systems. Memories based on beyond-CMOS, emerging technologies are alternatives to building denser and more energy efficient memories. For instance, spin-transfer torque magnetoresistive random access memory (STT-MRAM), spin-orbit torque magnetoresistive random access memory (SOT-MRAM), resistive random-access memory (RRAM), and ferroelectric field effect transistor-based random-access memory (FeFET-RAM) offer high integration density. Since these memories also offer the benefits of non-volatility, no power supply is needed to preserve information stored in their bit cells, resulting in much lower leakage when compared to SRAM. Various CiM designs based on emerging technologies have been proposed to-date, e.g., [7–15].

Given the variety of technologies and CiM designs available, picking the option that provides maximum performance and energy savings can be a complex task. Furthermore, as the memory hierarchy consists of different levels, i.e., L1, L2, L3 caches, aside from the main memory, different data-intensive applications are expected to benefit differently from CiM due to their distinct memory access patterns and the size of their workloads. When choosing a CiM design, it is important that we consider all the options available under conditions that are as uniform as possible. Namely, to ensure a uniform benchmarking for CiM designs, the technology node, memory size, CiM location in the memory hierarchy, etc. should be identical. In this regard, benchmarking methodologies, e.g., [10, 16, 17], are helpful to evaluate different CiM architectures considering figures-of-merit (FoMs) such as energy consumption, latency and density, as they allow for CiM evaluation when the designs are operating under similar conditions.

In this paper, we describe a strategy for *uniform* benchmarking of CiM designs based on different memory technologies. We follow a bottom-up methodology, i.e., from device to system, as illustrated in Fig. 1. We divide the benchmarking stages into phases 1 through 3 for ease of explanation (in Fig. 1 and also in Sec. 3). In essence, phase 1 consists of simulating a CiM design at the memory cell level, which takes technology models and the netlists for CiM circuits as the inputs, and leverages SPICE simulations to generate energy and delay results for the evaluated designs. Phase 2, in turn, takes into account the memory structure and organization, beside technology to generate energy and delay results at the array/memory bank level (with tools like NVSIM [18] or DESTINY [19]). The results from phase 2 are used in phase 3, which produces an estimate of energy and delay at the application level, taking into account array-level data. For phase 3, a comprehensive evaluation framework, Eva-CiM [17], is employed to analyze the CiM-capable instructions by capturing memory accesses and dependency-aware ISA traces.

We further present benchmarking results obtained with our uniform benchmarking approach for seven different memory technologies. Specifically, we comprehensively analyze the array-level performance of seven recent CiM designs implemented with volatile memories including SRAM [20] and DRAM [21] and non-volatile memories such as FeFET-RAM [12], STT-MRAM [9, 10] (with 2 different implementations), SOT-MRAM [9], and RRAM [22]. After we identify that the FeFET-RAM-based design [12] shows promising energy and delay savings at the array level, we perform a system level evaluation using the Eva-CiM [17] framework for this design and a SRAM-CiM baseline.

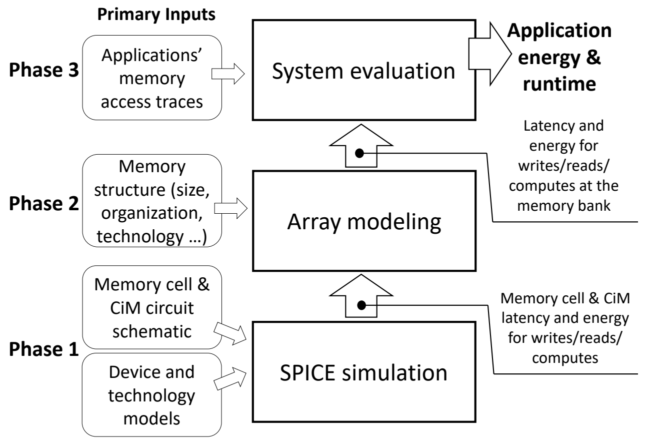

Primary Inputs
Phase 3
Applications' memory access traces
System evaluation
Application energy & runtime
Latency and energy for writes/reads/ computes at the memory bank
Phase 2
Memory structure (size, organization, technology ...)
Array modeling
Memory cell & CiM circuit schematic
Phase 1
Device and technology models
SPICE simulation
Memory cell & CiM latency and energy for writes/reads/ computes

Figure 1: The high level view of our bottom-up approach for uniform benchmarking of CiM designs.

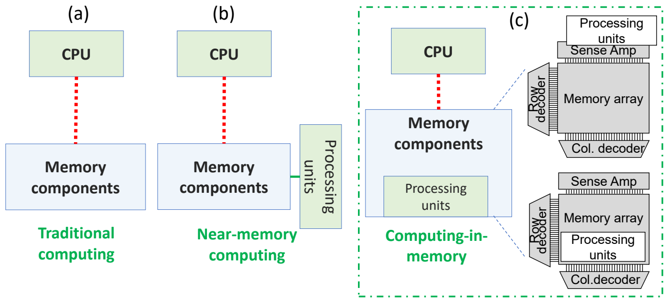


Figure 2: (a)A traditional computing, (b) a near-memory computing and a (c) computing-in-memory architectures.

2 BACKGROUND AND RELATED WORK

In this section, we discuss the terminology associated with CiM and provide a brief review of related work on the topic of CiM and emerging memory benchmarking.

2.1 Computing-in-Memory

In recent years, different approaches have emerged to address the overhead associated with data transfers in traditional computing systems (Fig. 2(a)). These approaches can generally be classified into two categories: near-memory processing (**NMP**, illustrated in Fig. 2(b)) and computing-in-memory (**CiM**, illustrated in Fig. 2(c)). With **NMP**, processing units (PUs) are placed close to the memory architecture, e.g., as proposed in [23, 24]. While NMP can leverage parallelism by exploiting the large memory internal bandwidth, the integration of density-optimized main memory and performance-optimized PUs on the same die may not be cost-effective due to fabrication challenges [25]. **CiM**, in turn, leverages changes in either the structure of memory cells or peripheral circuits to perform *in situ* computation in memory, i.e., at the array level [12]. By doing so, data transfers between a processor and the memory can be drastically reduced, which helps to save energy and shorten latency incurred by data transfers for a system that runs data-intensive applications.

2.2 Related Work

The variety of technologies and applications make the benchmarking of non-volatile memories and CiM designs from the device to the system level a non-trivial task. For instance, [26, 27] focus on benchmarking memories based on either CMOS and non-volatile emerging technologies. Their comparisons are carried out at cell and array levels, which gives us important insights as to what technologies might be more advantageous at several levels of the memory hierarchy (i.e., L1/L2/L3 caches, or main memory). However, as the focus of [26, 27] is not on evaluating CiM designs, there is still a need for a more broad benchmarking approach that looks at non-traditional architectures (e.g., the CiM) and measures their impact at the system and application levels.

In this regard, [16] proposes an evaluation flow for Deep Neural Networks (DNN) acceleration based on different technologies. Digital and analog CiM platforms are compared. Similar to the methodology described in this paper, [16] leverages device models in SPICE simulations. A 256×256 memory sub-array for each technology is simulated (including their respective CiM circuits), employing a 45 nm technology node. The SPICE simulation results provide inputs for simulation at a higher level, i.e., for a memory of 16Mbit size, using customized versions of the NVSIM [18] and CACTI [28] tools. A positive point of [16] is that it employs the same technology node and array size across all designs, which ensure uniform conditions for benchmarking. However, the methodology focuses only on CiM for DNN acceleration. In other words, although some designs, e.g., [10], could be used in a more general context, there is no evaluation for applications other than DNN acceleration. Furthermore, benchmarking the impact of CiM to the entire system is out of scope of their work.

The methodology described in this paper uses a similar approach as [16] for cell and array-level benchmarking (represented as phases 1 and 2 in Fig. 1). For system-level evaluation (phase 3), we rely on the framework of Eva-CiM [17] — a tool that enables evaluation of the impact of CiM with respect to system's energy consumption and performance, for a varied set of applications. A complete description of the phases of our benchmarking flow is given in Sec. 3, and relevant benchmarking results are presented in Sec. 4.

3 BENCHMARKING METHODOLOGY

In this section, we describe our bottom-up methodology for uniform benchmarking of CiM designs.

3.1 Cell level

Our CiM benchmarking evaluates figures-of-merit (FoM) such as energy and latency for different types of memory accesses — *read, write, and compute* — with CiM designs based on different technologies (phase 1 in Fig. 1). While the *read* and *write* accesses are similar to the accesses performed in regular RAM, the *compute* access emulates the logic/arithmetic CiM operations at the bitline level, which happens when two wordlines are asserted simultaneously [4, 10, 12, 29]. In order to measure FoMs for *read, write, and compute* at the cell level, we leverage SPICE simulations that include both the memory cell and the CiM peripheral circuitry contributions.

To evaluate **memory cells**, we leverage CMOS and emerging memories based on the device models of [18, 30–32]. Such device models may be based on either technology computer-aided design (TCAD) simulations or experimental data from real devices. To evaluate **CiM peripheral circuitry**, we implement customized sense amplifiers and adders at the peripheral circuitry for selected CiM designs, i.e., SRAM [20], DRAM [21], FeFET-RAM [12], STT-MRAM [9, 10] (2 different implementations), SOT-MRAM [9], and RRAM [22]. The results of the evaluation at the cell level (for both the memory and CiM circuitry components) are input to phase 2 of our benchmarking methodology to estimate array-level FoMs.

3.2 Array level

In phase 2 of our benchmarking methodology (see Fig. 1), we evaluate different CiM designs at the array level. For this purpose, we leverage DESTINY [19], and CACTI [28], which are memory simulation tools. DESTINY and NVSIM [18] are used for simulation of memories based on emerging technologies, and CACTI is used for CMOS-based memories. Importantly, DESTINY and NVSIM are built from the exact same framework. In fact, DESTINY can be considered as an extension of NVSIM with additional features, e.g. the support to 3D memories, which is why we chose to use DESTINY for implementing the phase 2 of our benchmarking methodology[1].

Note that DESTINY and CACTI tools do not support the evaluation of CiM designs and the FeFET memory cells [12] by default. For this reason, we have made modifications to the source code of these tools to enable support to CiM and FeFET-RAM evaluation. For instance, the latency, energy and leakage power of the CiM peripheral circuitry measured with SPICE simulations is included within a look-up structure inside the DESTINY/ CACTI source codes to specify customized sense amplifiers. Furthermore, for the FeFET technology, a new template for a memory cell that has separate read and write paths, with voltage-based writing mechanism is created inside the DESTINY framework. With these modifications, the FoMs at the array level include the leakage power, read/write/compute latency, and read/write/compute dynamic energy. Compute memory accesses are categorized into Boolean logic or word-wise addition.

The modified DESTINY and CACTI tools take two input files, i.e., one file for the **memory cell**, and another file for the **memory structure**. In the **memory cell** file, we specify the FoMs obtained from the cell-level evaluation. Furthermore, some device-specific parameters are required (e.g., read and write voltages), which are based on the device models [18, 30–32]. The area of the memory cell (in F^2) is also provided for estimating the total array area. Finally, in the **memory structure** file, we specify the total memory size we desire to evaluate, the data width, whether the memory is a scratch-pad memory (SPM) or a cache, and an optimization target for our design (e.g., write/read delay-optimized, write/read energy-optimized, or energy delay product (EDP)-optimized). The array-level evaluation obtained by DESTINY and CACTI are then used to estimate the impact of different CiM designs/technologies at the application level. Details about this phase of our benchmarking methodology are provided in Sec. 3.3.

[1]While we do not focus on evaluating CiM for 3D memories here, our methodology may be applied to these cases. Furthermore, the increasing memory density requirements of data-intensive applications make a strong case for implementations of CiM based on 3D memories.

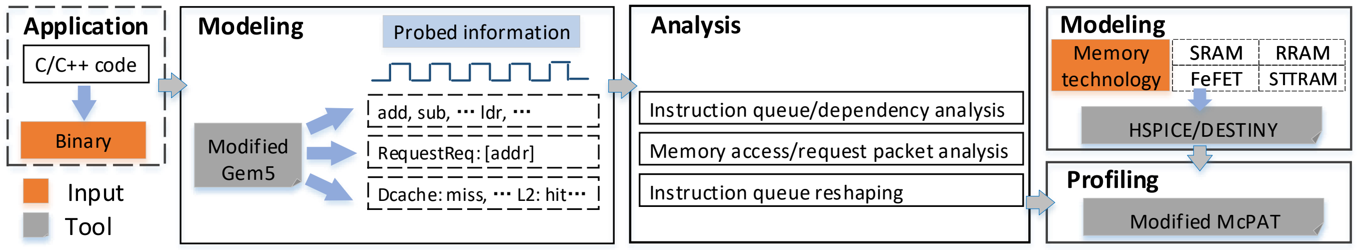
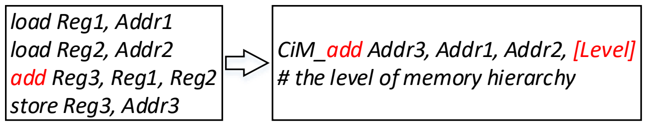


Figure 3: The Eva-CiM tool for CiM benchmarking at the system level (From [17]).



```
load Reg1, Addr1
load Reg2, Addr2
add Reg3, Reg1, Reg2
store Reg3, Addr3
```

```
CiM_add Addr3, Addr1, Addr2, [Level]
# the level of memory hierarchy
```

Figure 4: A direct *load-load-op-store* pattern that is appropriated for in-memory addition (From [17]).

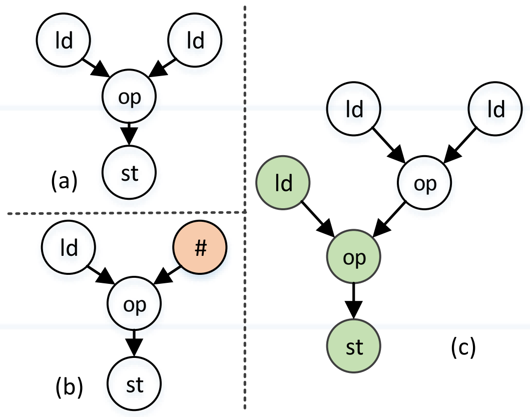


Figure 5: Variations of the *load-load-op-store* pattern. Unlike the regular pattern in (a), (b) replaces one source operand with an immediate value while (c) continues using the output before it is stored back to memory (From [17]).

3.3 System level

The system level evaluation of CiM corresponds to phase 3 in our benchmarking methodology outlined in Fig. 1. In this phase, we rely on a system evaluation tool, Eva-CiM [17]. We note that the operations that CiM can support largely depend on the underlying architecture and CiM circuitry. Furthermore, memory access patterns as well as the compiler can have a large impact on how effectively a CiM structure can be exploited. For example, in [10], the authors extend the original ISA with custom CiM instructions, dividing memory accesses for a given application into two categories: (*CiM-friendly* and *non-CiM-friendly*). The *CiM-friendly* operations (i.e., *AND*, *OR*, *XOR*, and *ADD*) are fed to the CiM for execution. The system in [10] assumes that CiM instructions are executed by a non-cacheable SPM module where data are accessed *in situ*. In this model, one *CiM-friendly* operation can always replace two memory reads. Thus, issues like memory hierarchy and locality of data are not taken into consideration. In contrast, our Eva-CiM in [17] automatically analyzes suitable offloaded instructions without enforcing either critically-defined ISA constraints or data locality assumption of the memory system.

Eva-CiM provides energy and performance estimates of the entire system for a given application based on various architectural setups. As part of its modeling strategy, Eva-CiM takes as input the array-level energy and latency results generated with modified DESTINY and CACTI tools (as described in Sec. 3.2). Eva-CiM employs GEM5 [33] as the backbone to simulate the behaviors of a given application program and fully captures the effects of CiM on both the host CPU and the complete memory hierarchy. It utilizes a modified McPAT and a system-wise profiler that combines the models at the application and CiM array levels to report the overall system energy and performance profiles. Hence, Eva-CiM is not limited to a particular technology/architecture nor a compiler, and supports various design space explorations, including: (i) practically analyzing if an application is CiM-friendly or not; (ii) comparing various device technologies to choose the more advanced one; (iii) determining the most appropriate CiM and memory architectures and trade-offs between energy and performance.

The flow of the Eva-CiM tool is illustrated in Fig. 3, which encompasses **Modeling**, **Profiling**, **Analysis** and **Application** stages. A complete description of every stage can be found in [17]. Here, we focus on explaining the Eva-CiM's *analysis stage*, which selects instructions to be offloaded to a CiM module for in-memory execution. Eva-CiM analysis identifies these "CiM-capable" instructions by capturing an application's memory access and dependency-aware ISA traces. For example, Fig. 4 illustrates a direct *load-load-op-store* instruction pattern that could be offloaded to a CiM architecture supporting in-memory addition. The *load-load-op-store* instruction pattern can also appear in less-direct forms, as exemplified in Fig. 5. In order to fully identify CiM offloading candidates, Eva-CiM embeds a trace-driven analyzer in GEM5 to perform the committed instruction queue and dependency analysis and construct the instruction dependency graphs (IDGs). Compared with [10], Eva-CiM accounts for interactions between CPU and memory system, and the impacts of multi-level cache hierarchy, such as cache access miss, *etc*., in order to provide more generalized system-level CiM benchmarking and overall benefit measurements.

4 EVALUATION

In this section, we employ the bottom-up benchmarking methodology described in Sec. 3 to evaluate CiM designs based on different technologies at the array and system levels.

Table 1: Per operation FoM estimation results for different CiM designs. In the Area part, M denotes Memory die area, and C denotes Computation area overhead. (iso-capacity: 32Mbit-single Bank, Data Width: 512-bit)

Metrics		SOT-MRAM[9]	STT-MRAM[10]	STT-MRAM[9]	RRAM[22]	SRAM[20]	DRAM[21]	FEFET-RAM[12]
Non-volatility		Yes	Yes	Yes	Yes	No	No	Yes
Area (mm^2)		M: 7.06 C:~0.3	M: 2.14 C:~0.3	M: 6.22 C:~0.3	M: 3.34 C: ~2.5	M: 10.38 C: ~0.5	M: 4.53 C: ~0.04	M+C: 5.85
Read Latency (ns)		2.85	1.90	2.89	1.48	2.9	3.4 per access	1.89
Write Latency (ns)		2.59	5.29	11.55	20.9	2.7		1.58
Read Dynamic Energy (nJ)		0.57	0.37	0.65	0.38	0.34	0.66 per access	0.48
Write Dynamic Energy (nJ)		0.66	0.67	1.2	2.7	0.38		0.50
In-memory Logic Energy (nJ)	Boolean	~0.64	~0.46	~0.79	~1.13	~0.59	~0.75	~0.54
	32-bit ADD	~1.92	~1.59	~2.37	~3.4	1.18	~11.25	~0.61
In-memory Logic Latency (ns)	Boolean	~3.35	~1.92	~3.48	~1.9	3.1	~13.6	~2.03
	32-bit ADD	~3.82	~2.54	~3.98	~2.31	4.2	~51	~2.74
Leakage Power (mW)		550	-	722.4	587.6	5243	335.5	595.97
Endurance		~ 10^{14} - 10^{15} [34]	~ 10^{14} - 10^{15} [34]	~ 10^{14} - 10^{15} [34]	up to 10^{12}[35]	Unlimited	10^{15}	10^{12} [36]
Data over-written issue		No	No	No	No	No	Yes	No

4.1 Array-level evaluation

We explore the performance of 7 different CiM designs at the array level with the ISO-memory-capacity constraint. We developed a 32Mb, single-bank CiM module based on volatile SRAM [20] and DRAM [21] and several representative non-volatile memories including FeFET-RAM [12], STT-MRAM [9, 10], SOT-MRAM [9], and ReRAM [22]. Note that we develop each CiM module from scratch with an identical 45nm technology library. Table 1 lists eleven performance metrics and per operation FoM results for each CiM design. We list our observations below:

Area. We divide the area cost into two parts: memory die area (M), and computation area (C) which includes the controller, modified decoder, sense amplifier, etc. In terms of memory die area, the 6T-SRAM and 2T-1R SOT-MRAM CiM modules impose a relatively larger area than that of other designs. When considering computation area, the DRAM design [21] takes up 0.04 mm^2, which is much smaller than that of other CiMs. Overall, 1T-1R STT-MRAM [10], DRAM [21], ReRAM [22] and FeFET-RAM [12] require the smallest footprint for implementing a 32Mb CiM module, respectively.

Latency. The ReRAM crossbar [22], as reported in Table 1, shows the shortest read latency (1.48 ns) as compared with other CiM designs, but it has the longest write latency (20.9 ns). The FeFET-RAM CiM achieves the shortest write latency and the second shortest read latency compared to other technologies. Therefore, FeFET-RAM could be considered as a potential CiM design if W/R latency optimization is the main concern.

Dynamic energy & Leakage power. We report dynamic energy for read, write, and in-memory operations in Table 1. As shown, in terms of read energy, SRAM [20], STT-MRAM [10], and ReRAM [22] CiMs respectively consume the smallest energy compared with different technologies. However, when it comes to write energy, SRAM, FeFET-RAM and SOT-MRAM outperforms others. To measure the computation energy at the array level, the CiMs' capability to perform (N)AND/(N)OR and full adder functions are taken into consideration. Based on Table 1, the STT-MRAM [10] and FeFET-RAM [20] respectively consume the smallest computation energy compared to other technologies to perform in-memory operations. It is worth pointing out that despite the DRAM CiM design based on Ambit [21] consumes 0.75 nJ to perform (N)AND/(N)OR based Triple-Row Activation mechanism, it requires over 14 memory cycles to perform the addition operation to avoid overwriting data, which leads to much higher energy consumption compared to other CiM designs.

In terms of leakage power consumption, the DRAM (335.5 mW) and SOT-MRAM (550 mW) could be considered as the most power-efficient CiMs. However, we observe that the SRAM design consumes ~7-16× more power compared with other CiM modules.

4.2 System-level evaluation

Based on the array-level evaluation, we identify the FeFET-RAM as one of the most promising designs among recent CiM implementations based on non-volatile emerging technologies. Therefore, we use this design as a case study for the system-level evaluation employing Eva-CiM [17]. Besides FeFET-RAM-CiM, we evaluate a CMOS-based SRAM baseline for CiM implementation [20].

The system-level evaluation quantifies the benefit of the two CiM designs for a set of 17 benchmark programs picked from various application domains (the same set of programs used in [17]). In [17], in order to decide where in the memory hierarchy CiM should be placed, experiments with the SRAM-based CiM are conducted. The results of these experiments indicate that, when CiM is placed at the L1 cache or the L1+L2 caches simultaneously, CiM offers the most benefits due to taking better advantage of data locality. While the degree of benefit varies from application to application, the observation holds for all the benchmark programs evaluated.

Therefore, to compare the CiM designs based on SRAM and FeFET, we consider the CiM placed at the L1 cache. Fig. 6 presents the performance and energy comparisons between SRAM and FeFET-RAM, where both the energy and the performance improvements are normalized to the non-CiM baseline system using CMOS SRAM. We observe that FeFET-RAM outperforms CMOS SRAM by an average of 60% when considering energy savings, an improvement that is consistent across all the benchmarks. Regarding speedup, both technologies show similar benefits of about ~1.5× (on average) when compared to a conventional architecture (where processing does not take place in memory).

5 CONCLUSION

In this paper, we present our methodology for *uniform* benchmarking of CiM designs based on different memory technologies. We

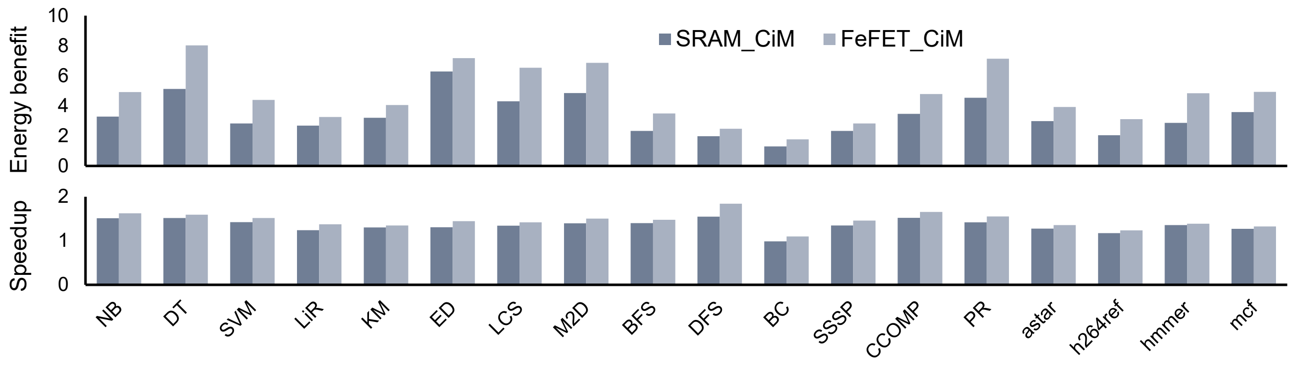


Figure 6: Benefits for CMOS-SRAM-CiM v.s. FeFET-RAM-CiM: Energy improvement (top); Performance improvement (bottom). (From [17]).

analyze various options of CiM designs from devices to the system level. Our benchmarking methodology is employed to comprehensively analyze the array-level performance of 7 recent CiM designs implemented by volatile memories including SRAM [20] and DRAM [21] and representative non-volatile memories such as FeFET-RAM [12], STT-MRAM [9, 10] (with 2 different implementations), SOT-MRAM [9], and RRAM [22]. We further describe a system-level evaluation of the FeFET-RAM-based CiM (a design with superior overall FoMs) in terms of performance and energy efficiency, using Eva-CiM[17]. The results of the system-level evaluation show that the FeFET-RAM-based CiM outperforms a CMOS SRAM-CiM baseline by an average of 60% across a set of 17 benchmarks when considering energy savings. For speedup, both technologies offer similar benefit of about ~1.5× when compared to a conventional CMOS processor where processing does not happen in memory.

ACKNOWLEDGMENT

This work was supported in part by the U.S. National Science Foundation under grant 1640081 through the E2CDA program, and NRI, a Semiconductor Research Corporation (SRC) program.

REFERENCES

[1] C. P. Chen et al. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314 – 347, 2014.
[2] W. A. Wulf et al. Hitting the memory wall: implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24, 1995.
[3] S. Aga, et al. Compute caches. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 481–492, 2017.
[4] S. Jeloka, et al. A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6T Bit Cell Enabling Logic-in-Memory. *JSSC*, 51(4):1009–1021, 2016.
[5] M. Kang, et al. An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM. In *ICASSP*, 2014.
[6] D. Reis, et al. A Fast and Energy Efficient Computing-in-Memory Architecture for Few-Shot Learning Applications. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 127–132, 2020.
[7] N. Talati, et al. Logic Design Within Memristive Memories Using Memristor-Aided loGIC (MAGIC). *TNANO*, 15(4):635–650, 2016.
[8] B. Li, et al. RRAM-Based Analog Approximate Computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(12):1905–1917, 2015.
[9] S. Angizi et al. Graphs: A graph processing accelerator leveraging sot-mram. In *2019 DATE*, pages 378–383. IEEE, 2019.
[10] S. Jain, et al. Computing in Memory With Spin-Transfer Torque Magnetic RAM. *TVLSI*, PP(99):1–14, 2017.
[11] F. Parveen, et al. Low power in-memory computing based on dual-mode SOT-MRAM. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6, 2017.
[12] D. Reis, et al. Computing in memory with FeFETs. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 1–6, 2018.
[13] A. F. Laguna, et al. Ferroelectric FET Based In-Memory Computing for Few-Shot Learning. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, pages 373–378, 2019.
[14] Y. Zhu et al. Statistical training for neuromorphic computing using memristor-based crossbars considering process variations and noise. In *DATE*, pages 1590–1593. IEEE, 2020.
[15] S. Zhang et al. Aging-aware lifetime enhancement for memristor-based neuromorphic computing. In *DATE*, pages 1751–1756. IEEE, 2019.
[16] S. Angizi, et al. Accelerating Deep Neural Networks in Processing-in-Memory Platforms: Analog or Digital Approach? In *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 197–202. IEEE, 2019.
[17] D. Gao, et al. Eva-CiM: A System-Level Performance and Energy Evaluation Framework for Computing-in-Memory Architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2020.
[18] X. Dong, et al. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):994–1007, 2012.
[19] M. Poremba, et al. DESTINY: A tool for modeling emerging 3D NVM and eDRAM caches. In *DATE*, 2015.
[20] C. Eckert et al. Neural cache: Bit-serial in-cache acceleration of deep neural networks. In *2018 ISCA*, pages 383–396. IEEE, 2018.
[21] V. Seshadri et al. Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology. In *2017 MICRO*, pages 273–287. IEEE, 2017.
[22] T. Tang et al. Binary convolutional neural network on rram. In *2017 ASP-DAC*, pages 782–787. IEEE, 2017.
[23] D. Patterson, et al. Intelligent RAM (IRAM): chips that remember and compute. In *ISSCC*, 1997.
[24] R. Nair, et al. Active Memory Cube: A processing-in-memory architecture for exascale systems. *IBM J Res Dev.*, 59(2/3):17:1–17:14, 2015.
[25] D. Patterson, et al. A case for intelligent RAM. *IEEE Micro*, 17(2), 1997.
[26] K. Prall. Benchmarking and Metrics for Emerging Memory. In *2017 IEEE International Memory Workshop (IMW)*, pages 1–5, 2017.
[27] D. Reis, et al. Design and Analysis of an Ultra-Dense, Low-Leakage, and Fast FeFET-Based Random Access Memory Array. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 5(2):103–112, 2019.
[28] K. Chen, et al. CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 33–38. IEEE, 2012.
[29] S. Angizi, et al. AlignS: A Processing-In-Memory Accelerator for DNA Short Read Alignment Leveraging SOT-MRAM. *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.
[30] R. Vattikonda, et al. Modeling and minimization of pmos nbti effect for robust nanometer design. In *DAC*, 2006.
[31] A. Aziz, et al. Physics-based circuit-compatible spice model for ferroelectric transistors. *IEEE Electron Device Lett.*, 37(6):805–808, 2016.
[32] S. Ikeda, et al. A perpendicular-anisotropy CoFeB–MgO magnetic tunnel junction. *Nature materials*, 9(9):721–724, 2010.
[33] N. L. Binkert, et al. The gem5 simulator. *SIGARCH Computer Architecture News*, 39(2):1–7, 2011.
[34] J. Kan et al. Systematic validation of 2x nm diameter perpendicular mtj arrays and mgo barrier for sub-10 nm embedded stt-mram with practically unlimited endurance. In *2016 IEDM*, pages 27–4. IEEE, 2016.
[35] C.-W. Hsu et al. Self-rectifying bipolar tao x/tio 2 rram with superior endurance over 10 12 cycles for 3d high-density storage-class memory. In *2013 Symposium on VLSI Technology*, pages T166–T167. IEEE, 2013.
[36] C.-H. Cheng et al. Low-leakage-current dram-like memory using a one-transistor ferroelectric mosfet with a hf-based gate dielectric. *IEEE electron device letters*, 35:138–140, 2013.