

The Landscape of Compute-near-memory and Compute-in-memory: A Research and Commercial Overview

ASIF ALI KHAN, TU Dresden, Germany

JOÃO PAULO C. DE LIMA, TU Dresden and ScaDS.AI, Germany

HAMID FARZANEH, TU Dresden, Germany

JERONIMO CASTRILLON, TU Dresden and ScaDS.AI, Germany

In today's data-centric world, where data fuels numerous application domains, with machine learning at the forefront, handling the enormous volume of data efficiently in terms of time and energy presents a formidable challenge. Conventional computing systems and accelerators are continually being pushed to their limits to stay competitive. In this context, computing near-memory (CNM) and computing-in-memory (CIM) have emerged as potentially game-changing paradigms. This survey introduces the basics of CNM and CIM architectures, including their underlying technologies and working principles. We focus particularly on CIM and CNM architectures that have either been prototyped or commercialized. While surveying the evolving CIM and CNM landscape in academia and industry, we discuss the potential benefits in terms of performance, energy, and cost, along with the challenges associated with these cutting-edge computing paradigms.

1 INTRODUCTION

In conventional computing systems, the processor and memory are two independent entities connected via communication pathways, known as buses. When the CPU processes data, it requires fetching it from memory via the bus, conducting the necessary computations, and subsequently storing the results back in memory. This off-chip communication becomes a limiting factor for data-intensive workloads due to the limited transfer rate and high energy per bit of buses. For example, the data transfer between the logic (CPUs and GPUs) and memory chips (DRAM or flash memory) requires approximately 10–100 times more energy than the logic operation itself [1]. *Compute-near-memory* (CNM) and *compute-in-memory* (CIM) concepts address this bottleneck by enabling computations close to where the data resides. This is achieved either by implementing CMOS logic on or closer to the memory chip, or by leveraging the inherent physical properties of memory devices to perform computations in place.

The core concept behind CNM/CIM is not entirely new. However, the sudden surge in these systems can be attributed to two primary factors, namely, the exponential increase in the volume of data required for modern applications, and the technological readiness. Recent advancements in machine learning, particularly the emergence of generative AI and *large language models* (LLM), demand the processing of terabytes of data, substantial computational resources, and complex execution, thus highlighting the limitations of traditional computing systems. A recent study revealed that OpenAI utilized over 3600 of NVIDIA's HGX A100 servers, totaling around 29,000 GPUs, to train ChatGPT, resulting in a daily energy consumption of 564 MWh [2]. Projections indicate that by 2027, AI is expected to consume between 85 and 124 TWh annually, equivalent to approximately 0.5% of the world's total electricity consumption. It is no surprise that Microsoft has announced plans to develop its own nuclear reactors to power their data centers [3].

Currently, machine learning applications primarily leverage GPU accelerators like A100, H100, GH200, application-specific integrated circuits (e.g., Google's TPU), and dataflow processors as in the case of companies like GraphCore, Cerebras, Groq, and SambaNova [4]. Over the past few

Authors' addresses: Asif Ali Khan, TU Dresden, Dresden, Germany, asif.ali.khan@tu-dresden.de; João Paulo C. de Lima, TU Dresden and ScaDS.AI, Dresden, Germany, joao.lima@tu-dresden.de; Hamid Farzaneh, TU Dresden, Dresden, Germany, hamid.farzaneh@tu-dresden.de; Jeronimo Castrillon, TU Dresden and ScaDS.AI, Dresden, Germany, jeronimo.castrillon@tu-dresden.de.

years, CNM/CIM systems have also transcended their prototypical stages and successfully entered the market. The timing of these advancements in CIM/CNM systems is of paramount importance as it perfectly aligns with the AI revolution. As a result, numerous companies have emerged in the last few years offering CIM/CNM solutions for various use domains. This surge reflects a competitive landscape where these companies are striving to leverage the demand and cater to various market segments. All commercially available solutions hold the promise of significantly reducing execution time and energy consumption for data-intensive workloads.

This survey explores CNM and CIM architectures, detailing their technologies, fundamental concepts, working principles, and the evolving landscape in academia and industry. Addressing a broader audience, it provides foundational concepts for non-experts while delivering state-of-the-art insights for experts in the domain. It also summarizes the impact and challenges associated with adopting the novel CIM/CNM computing paradigms. Concretely, our discussion revolves around three key aspects:

- (1) **Key technologies and concepts:** In CNM systems, a specialized CMOS logic is integrated into the memory chip. This logic can be either general-purpose, as in UPMEM systems [5], or domain-specific, as in systems developed by Samsung [6, 7] and SK Hynix [8, 9], integrated within DRAM memory chips. While CNM significantly reduces data movement, it does not eliminate it. In contrast, CIM nearly eliminates data movement by performing computations within the same devices that store the data. A particularly noteworthy operation is the analog dot-product in memory, which is of significant importance to the machine learning domain and can be performed in constant time. Initially demonstrated in crossbar-configured resistive *non-volatile memory* (NVM) technologies like *phase change memory* (PCM) [10] and *resistive RAM* (RRAM) [11], this concept has also been shown with SRAM, *magnetic RAM* (MRAM) [12], and *ferroelectric field-effect transistor* (FeFET) [13]. While other arithmetic, search and boolean logic operations have also been demonstrated using CIM, they have received comparatively less attention.
- (2) **Commercial trends:** As the demand for fast and efficient computing systems continues to rise, the in-/near-memory computing market is experiencing rapid expansion. In 2022, this market was valued at USD 15.5 billion, with an anticipated *compound annual growth rate* (CAGR) of 17.5% over the next decade [14]. This growth is underscored by the proliferation of startups offering CIM and CNM solutions. Some of these companies have secured hundreds of millions of dollars in early funding rounds. While many of these companies provide innovative solutions for data-intensive applications (dominated by AI inference), there is no clear winner yet. At present, these solutions are predominantly based on SRAM technology, although solutions based on resistive NVM and flash technologies also exist [15]. This trend can be attributed to the mature tools and design processes for SRAM compared to emerging NVMs. However, considering the SRAM's scalability aspects and its static power consumption, it is likely that NVMs, particularly PCM, RRAM, MRAM, and FeFET, will progressively replace or complement SRAM as these technologies mature.
- (3) **Challenges:** Although CIM/CNM systems are at the tipping point, they are yet to make substantial inroads into the market. The predominant obstacle facing these systems is perhaps the absence of a software ecosystem, which renders programmability and optimization exceedingly challenging. This is also highlighted by a recent Meta article [16] stating, *We've investigated applying processing-in-memory (PIM) to our workloads and determined there are several challenges to using these approaches. Perhaps the biggest challenge of PIM is its programmability*. Other challenges requiring attention include: addressing reliability concerns associated with emerging NVMs (particularly in the CIM context), developing

novel performance models, profiling and analysis tools for these systems, which could be leveraged to exploit their potential effectively.

The remainder of this paper is structured as follows: Section 2 explains the terminology associated with these domains and provides insights into the conventional Von-Neumann computing approach, as well as the emerging data-centric paradigms. In Section 3, a comprehensive overview of promising memory technologies within the context of CIM and CNM systems is provided. Section 4 outlines various common CIM and CNM systems including very recent prototype chips from various industries. Lastly, Section 5 presents a comprehensive overview of the commercial landscape for these systems (start-ups), discussing their products details, target application domain, and funding status. Finally, Section 6 concludes the paper by summarizing our key observations and providing insights and recommendations into the future.

2 TERMINOLOGY AND BACKGROUND

This section highlights the bottleneck in the Von Neumann computing model by discussing its working mechanism, motivates the need for memory-centric computing, and explains the terminology.

2.1 Mainstream Von-Neumann Computing

As depicted in Figure 1a, the interaction between memory and the processor in the Von Neumann architecture is facilitated through address and data buses. However, because CPU performance significantly outpaces memory performance, the Von Neumann models are often bottlenecked by the memory. To address this challenge and mitigate the impact of larger memory access latencies on the CPU, modern processors incorporate a tiered hierarchy of caches. Caches are smaller memory units that, while being much smaller compared to main memory and storage, are notably faster. The first-level cache (L1) is typically integrated onto the CPU chip and operates nearly at CPU speed, enabling single-cycle access. L2 cache is usually shared by multiple cores and can vary in location depending on the design goals. Some systems even include an L3 cache, usually larger and situated off-chip.

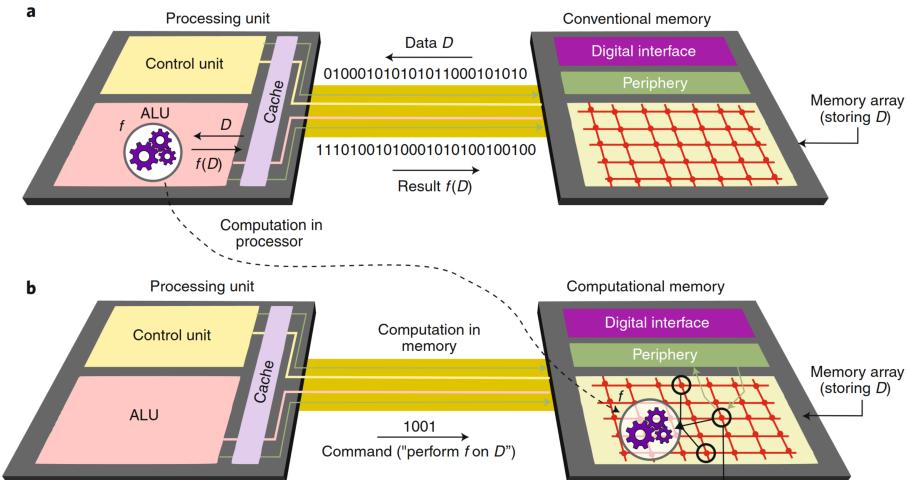


Fig. 1. (a) Conventional computing system where an operation f is performed on data D in the CPU (b) Memory-centric design where f is computed in the proximity of D and CPU is mainly working as a control unit [17].

Note that Von Neumann architectures are characterized by the sequential execution of instructions. However, in multi-core CPU systems, parallel execution of instructions at various levels of granularity, including instruction-level, data-level, and thread-level parallelisms, is supported. To enhance performance and energy efficiency in resource-constrained systems, specialized accelerators are often developed and integrated on the same chip. For instance, in application domains such as embedded systems, digital signal processing, and networking, *multiprocessor system-on-chip* (MPSoC) architectures are employed. These integrate multiple processor cores, memory, input/output interfaces, and potentially specialized hardware accelerators, enabling parallel execution of different tasks to meet specific constraints.

Although these designs may significantly enhance performance when compared to conventional CPU-only systems, the underlying design principle remains CPU-centric and follows the Von Neumann model of execution. Consequently, the performance improvements, largely resulting from concurrent execution, heavily rely on the nature of the application. In cases where an application is memory-bound, i.e., most of the execution time is spent on the memory accesses and not on the actual compute operations, the shared data bus is fully occupied and becomes a bottleneck. Even for compute-bound applications, where these architectures can yield substantial gains in execution time, power consumption remains largely unaffected and might even increase due to the complex structure of these systems.

2.2 Memory-centric computing

Unlike conventional computing systems where CPU has a central role and is responsible for all computations, most computations in the memory-centric designs are performed within or near memory. As depicted in Figure 1b, the core concept revolves around minimizing data transfer on the bus by relocating a substantial share of computations closer to the data (memory). The CPU's primary role becomes issuing commands and handling computations that cannot be effectively executed in close proximity to the memory.

The concept of memory-centric computing is not a novel one, but it has experienced a significant surge in recent years. Consequently, various terms have emerged, often referring to the same idea, and more detailed classifications have been introduced in architectural designs. This section aims to clarify the terminology surrounding these approaches.

2.2.1 Compute-in-memory. Computing systems can be broadly divided into two categories: *compute-in-memory* (CIM) systems and *compute-outside-memory* (COM) systems (see Section 2.1). In the literature, there are different names for similar things. These architectures are often named based on (1) the *location* of the compute units within the memory hierarchy (near cache or near main memory), or (2) based on the underlying paradigm, i.e., whether the memory device itself is used to implement computation (CIM), or whether extra CMOS-logic is added near the memory to perform computations (*compute-near-memory* (CNM)). Figure 2 shows an overview of different processor and memory system configurations. A compute operation can be a logic operation or an arithmetic operation such as addition and multiplication. CIM systems are also frequently referred to as *in-memory-computing* (IMC), *in-memory-processing* (IMP), *processing-in-memory* (PIM), *processing-using-memory* (PUM) or *logic-in-memory* (LIM) systems [18]. For the purposes of this report, we will use the term CIM.

2.2.2 Compute-near-memory. When operations are computed outside the memory (COM) using conventional computing cores (Fig 2.a), the architecture is a conventional Von Neumann system (see Section 2.1). On the other hand, if the computations are performed outside the memory but with a dedicated logic unit connected to the memory module via a high-bandwidth channel (Fig 2.b), the system is referred to as a *compute-near-memory* (CNM) or *near-memory-computing* (NMC) or

near-memory-processing (NMP), or *processing-near-memory* (PNM) system. In this report, we will restrict ourselves to calling it CNM.

In the CIM category, computations can be carried out using memory cells within the memory array, known as CIM-array (CIM-A) (see Fig 2.d). Alternatively, computations can occur in the memory peripheral circuitry, termed CIM-peripheral (CIM-P) (see Fig 2.c).

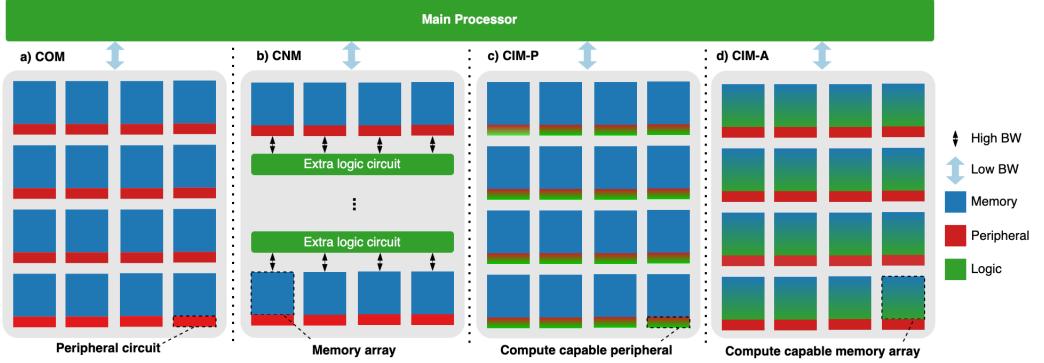


Fig. 2. High-level overview of systems where computation is performed a) COM (mainstream computing): outside of memory system, b) CNM: using a logic connected to the memory via the memory high-bandwidth channel, c) CIM-P: in the memory peripheral circuitry, and d) CIM-A: using memory cells within the memory array.

In CIM-A, memory cells are often modified to support logic design, e.g., in [19]. Sometimes, it also necessitates changes to the periphery to support the modified cells. Therefore, some literature further divides the CIM-A designs into basic CIM-A that do not require any modifications to the periphery, e.g., [20], and hybrid CIM-A that requires support from the peripheral circuit. A well-known example of a hybrid CIM-A is the MAGIC design [19] that requires extending the peripheral circuit to write multiple memory rows simultaneously.

Typical examples of CIM-P architectures are crossbars employing *analog-to-digital converters* (ADCs) and *digital-to-analog converters* (DACs) to implement *matrix-vector multiplication* (MVM) and other logic operations [21, 22]. Additionally, CIM-P designs employing customized sense amplifiers also exist [23]. Similar to CIM-A, CIM-P can be either basic, as in Pinatubo [23], requiring no changes to the memory array, or hybrid, as seen in ISAAC [22]. A summary of the terminology's classification is presented in Figure 3.

Both CIM-A and CIM-P can also be used together, wherein the memory array calculates partial results that are later post-processed or accumulated in the peripheral circuit. In such cases, it is referred to as a CIM architecture.

3 TECHNOLOGY OVERVIEW

In this section, we present an overview of the main memory cells used in various CNM and CIM systems, encompassing both volatile *static random-access memory* (SRAM), *dynamic random-access memory* (DRAM) and non-volatile types such as PCM, MRAM, RRAM and FeFETs. These memory technologies are versatile enough to serve as main memory for data storage and support CNM without requiring any modification to the memory chips. Additionally, we explore how these memory cells can be leveraged for in-memory computation, considering technological aspects such as performance, energy consumption, lifetime, CMOS compatibility, and other relevant factors. Before going into the individual technologies, let us first explain the different components of the memory subsystem.

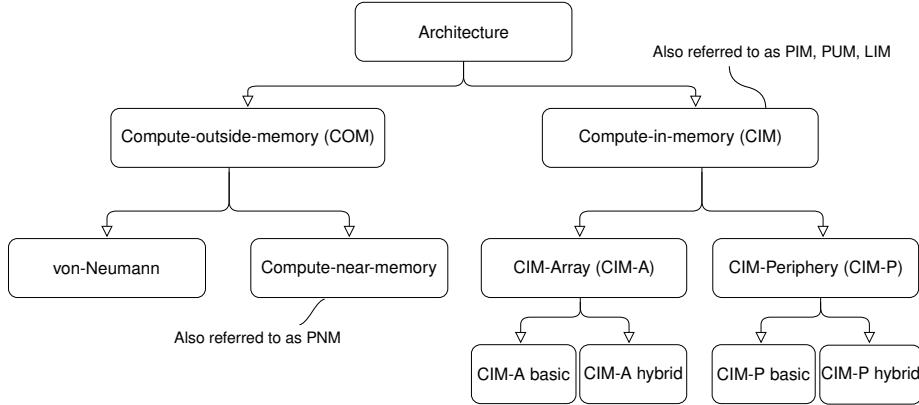


Fig. 3. CIM and CNM classification.

3.1 Memory subsystem

The memory subsystem typically consists of a four-level hierarchy, each with its own characteristics and access time. The fastest and smallest level is the CPU registers, while the slowest one in the hierarchy is storage devices (HDD/SSD). The CNM/CIM concepts have been proposed at different levels in the memory hierarchy, e.g., in-cache computing [24], in-DRAM [25], in-storage computing [26]. However, CNM/CIM at the main memory level have rightly gained more attention than others.

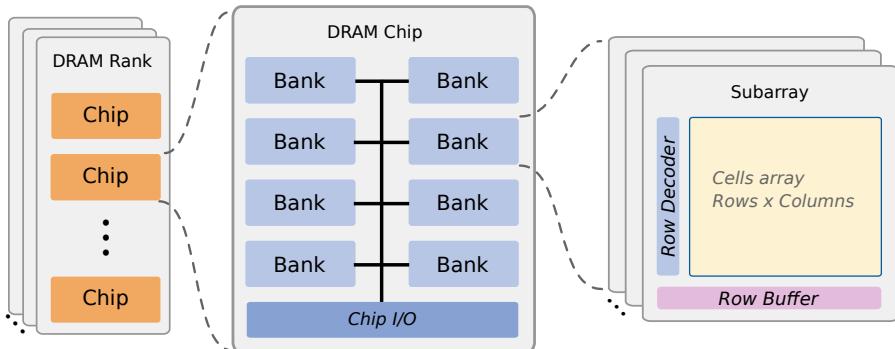


Fig. 4. Typical DRAM system organization.

The main memory is also typically organized hierarchically, as shown in Figure 4. The main components of the hierarchy are memory cells, subarrays, banks, and ranks. A cell is the fundamental working unit and serves as a building block to form arrays. An array is a 2D grid of cells connected via *word lines* and *bitlines*. A word line is a horizontal line that connects all the memory cells in a row, while bitlines, on the other hand, connect all cells in a column. Memory cells are placed at the intersection of word lines and bitlines. The combination of a specific word line and a bit line uniquely identifies each memory cell. When a particular memory cell needs to be accessed, the corresponding word line and bit line are activated, allowing data to be read from or written into that cell.

3.2 DRAM

DRAM is the most mature and widely used memory technology today. A DRAM cell is composed of a transistor and a capacitor. When the capacitor is fully charged, it represents the logical value 1, while a discharged capacitor represents the logical value 0. To access data from the DRAM array, the memory controller brings a particular row into the row buffer by sending an *activate* command. A *read* command is then issued to read specific column(s) from the row buffer and put them on the bus.

DRAM has scaled nicely for decades and has been used across application domains and systems ranging from HPC to portable devices. However, it is presently facing several challenges to remain the dominant technology. The increasing demand for higher capacity has put tremendous pressure on the DRAM capacitor size to shrink which makes it susceptible to errors. Also, the increase in capacity is significantly increasing the refresh power budget.

To address the escalating demands for higher bandwidth in modern applications, 3D stacked DRAM architectures, such as *high bandwidth memory* (HBM) (see Section 4.1.3), have been proposed. These architectures consist of stacked DRAM dies atop a logic layer, interconnected through *through-silicon vias* (TSVs), resulting in remarkably higher bandwidth. These structures are also employed in a series of CNM solutions, where the logic layer is used to implement custom logic and perform computations in closer proximity to the data [27, 28].

From the CIM perspective, the majority of in-DRAM implementations rely on charge sharing, wherein multiple rows are activated simultaneously. The shared charge is then utilized in a controlled manner to carry out various logic and data copy operations [25]. Moreover, cleverly manipulating the memory timing parameters, deviating from standard timings, has also been employed to implement different logic operations [29].

3.3 SRAM

SRAM is another mature memory technology that provides fast and efficient memory accesses. It is commonly used in caches, register files, and other high-speed memory applications where speed and low access latency are critical. An SRAM cell consists of multiple transistors arranged in a specific configuration to hold one bit of data. The most common configuration of an SRAM cell is a pair of cross-coupled inverters that are connected in a feedback loop, forming a latch.

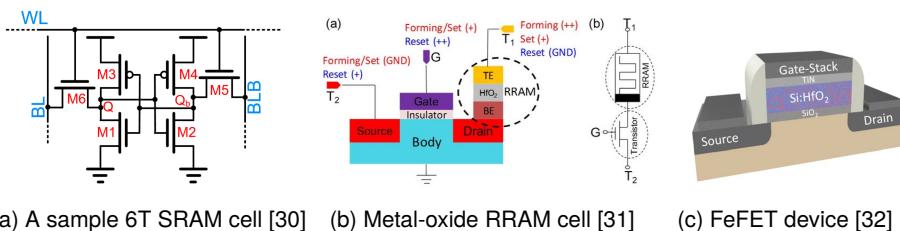


Fig. 5. Cell structures of various memory technologies

To read and store data on the cell, the bitline terminals, *bitline* (BL) and *bitline bar* (BLB) (see Fig. 5a), are precharged and discharged, and the wordline(*word line* (WL)) is activated or deactivated depending on the values reading/writing from/to the cell.

There have been proposals for in-SRAM computing, especially at the last-level cache, which can be considerably slower compared to the L1 cache (e.g., by an order of magnitude). Similar to DRAM, most in-SRAM computing architectures also leverage charge sharing in the bitlines. Specifically, precharging the bitlines in a controlled manner and activating multiple rows simultaneously enables

performing logic operations [24]. For bitwise multiplication in SRAM, research has demonstrated that the amplitude of the input voltage at the WL directly influences the discharge rate of the BLB. The voltage discharge on BLB, achieved within a specific timeframe, effectively represents a one-bit multiplication of the data stored in the SRAM cell.

3.4 Phase change memory (PCM)

PCM is resistive memory technology that employs reversible phase changes in materials to store data. The earliest demonstration of a 256-bit PCM prototype dates back to 1970 [33]. Today, PCM stands as one of the most extensively researched NVM technologies. A PCM device comprises a phase-changing material sandwiched between two electrodes (very similar to Fig. 5b), which transitions between crystalline (low resistance state) and amorphous (high resistance state) phases. These two resistance states represent binary logic states, i.e., 1 and 0.

Typically, PCM requires a relatively high programming current ($>200\mu\text{A}$), but this can be mitigated to less than $10\mu\text{A}$ by scaling down the device size [10, 34]. As PCM stores data based on resistance, it can be programmed to encompass more than two resistance states, allowing for multi-level cells to represent more than a single bit of information. Nevertheless, relying on multiple resistance states for prolonged periods poses challenges, as the device resistance tends to drift over time, making it difficult to discern between resistance states.

3.5 Resistive RAM (RRAM)

RRAM is another class of resistive memory technologies that utilizes the resistive switching phenomenon in metal oxide materials to store data [11]. As shown in Fig. 5b, a typical RRAM cell comprises a top and a bottom electrode with a thin oxide layer sandwiched in between. To achieve resistive switching, a high electric field is applied to the RRAM cell, leading to the creation of oxygen vacancies within the metal oxide layer. This process results in the formation of conductive filaments, causing the device state to transition from a high resistance to a low resistance (set) state. To revert to the high resistance (reset) state, the device is subjected to V_{RESET} , which breaks the conductive filament, allowing the oxygen ions to migrate back to the bulk. Compared to PCM, RRAM exhibits several advantages, including higher write endurance ($>10^{10}$), faster write operations, larger resistance on-off ratios, and improved scalability prospects [11]. However, ReRAM does suffer from inconsistent electrical characteristics, meaning it exhibits larger variations in resistance across different devices [10].

3.6 Magnetic RAM (MRAM)

MRAM store data in nano-scale ferromagnetic elements via magnetic orientation [35]. An MRAM cell is a *magnetic tunnel junction* (MTJ) device composed of two ferromagnetic layers, namely a fixed reference layer and a free layer, separated by an insulating layer. The free layer holds the data bit, and reading it involves passing an electric current and measuring its resistance. For data writing into an MRAM cell, various techniques can be used. The most common method is the *spin-transfer-torque* (STT), which utilizes spin-polarized electric current to change the free layer's magnetic orientation. *Spin-orbit-torque* (SOT)-MRAM, on the other hand, uses an in-plane current through the heavy metal layer to generate a spin current that exerts a torque on the magnetization of the free layer. The relative orientations of the free and fixed layers result in different resistance states. MRAM exhibits virtually unlimited endurance and acceptable access latency. However, it is faced with challenges such as a larger cell size and a smaller on/off resistance ratio, limiting an MRAM cell to store only one bit of data [36].

Device	SRAM	DRAM	RRAM	PCM	STT-MRAM	FeFET
Write time	1 – 10ns	> 20ns	> 10ns	~ 50ns	> 10ns	~ 10ns
Read time	1 – 10ns	> 20ns	> 10ns	> 10ns	> 10ns	~ 10ns
Drift	No	No	Weak	Yes	No	No
Write energy (per bit)	1 – 10fJ	10 – 100fJ	0.1 – 1pJ	100pJ	~ 100fJ	> 1fJ
Density	Low	Medium	High	High	Medium	High
Endurance	> 10 ¹⁶	> 10 ¹⁶	> 10 ⁵ – 10 ⁸	> 10 ⁵ – 10 ⁸	> 10 ¹⁵	> 10 ¹⁵
Retention	Low	Very Low	Medium	long	Medium	long

Table 1. A comparison of the key features across different mainstream CMOS and emerging memristive technologies [39].

3.7 Ferroelectric Field-Effect Transistor (FeFET)

Since the discovery of ferroelectricity in hafnium oxide, FeFETs have received considerable attention. FeFETs are non-volatile three-terminal devices, offering high I_{on}/I_{off} ratios and low read voltage. Unlike *metal-oxide-semiconductor* (MOS)-FETs, FeFETs incorporate a ferroelectric oxide layer in the gate stack, as shown in Figure 5c. The nonvolatility arises from hysteresis due to the coupling between the ferroelectric and CMOS capacitances (C_{FE} and C_{CMOS}). The three-terminal structure of FeFETs enables separate read and write paths. Reading involves sensing the drain-source current, while writing involves switching the ferroelectric polarization with an appropriate V_{gs} voltage. Unlike two-terminal devices with variable resistance, FeFETs do not require a drain-source current during the writing process, leading to low writing energy consumption [37]. There are various CIM architectures exploiting different properties of FeFETs. For instance, for boolean operations, [37] proposes precharging the bitlines followed by simultaneous activation of the target rows, and using differential sense amplifiers to discern the output.

3.8 Comparison and discussion

Table 1 presents a comparison between mainstream and emerging memory devices, discussed in the preceding sections, with respect to performance, reliability, and energy consumption, among others. This analysis gives insights into their suitability for different application domains. It is clear that no single memory device can optimize all metrics. Nonetheless, recent investigations into machine learning use cases show that different phases of machine learning tasks demand different memory device properties, potentially offering the opportunity to employ various devices for various application domains (or tasks within a domain) and achieve the best results [38].

PCM, RRAM, MRAM and FeFET fall under the category of memristive technologies, where devices can exhibit multiple resistance states. This characteristic has been effectively leveraged to perform MVM, as depicted in Figure 15a. Although the analog computation may not be entirely precise, some loss in accuracy is acceptable in many application domains, particularly for machine learning applications. Numerous CIM architectures have been proposed using this technique to accelerate neural network models (see Section 4.2).

Figure 6 shows the importance of various device properties for neural network training and inference. For training, frequent weight updates within the memory are crucial, making memory technologies like PCM and RRAM, with limited endurance and expensive write operations, poorly suitable for training acceleration. However, in inference, where operations are predominantly read-based with minimal writes to the crossbar array, the same technology could outperform others by orders of magnitude. Similarly, retention for training is the least important but is critical for inference.

The arguments around Figure 6 generally hold true for other metrics and other application domains. Although machine learning constitutes a significant area, it is not the only domain benefiting from the

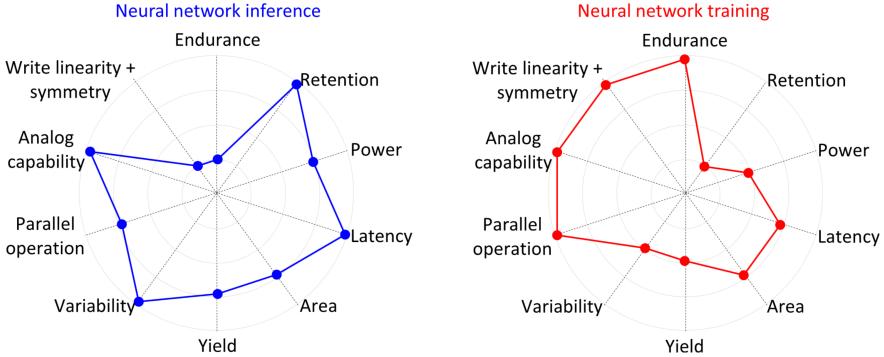


Fig. 6. A spider chart showing the importance of different attributes of the NVM technologies for the neural network training and inference. More distance from the center means more important [38].

CIM paradigm. Many other data-intensive application domains have also effectively exploited the CIM paradigm. Figure 7 shows a landscape of CIM applications, emphasizing precision considerations, computational complexity and memory access requirements. These applications are classified into three categories based on their precision demands. This data pertains to 2020. Over the past two years, additional application domains, such as databases, bioinformatics, and solving more complex algebraic tasks, have gained significant attention as well.

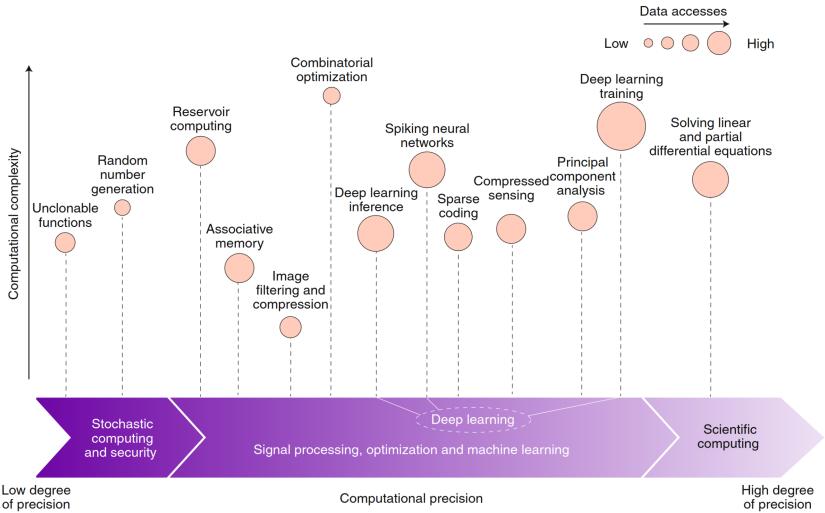


Fig. 7. The applications landscape for CIM and CNM [17].

4 SELECTED ARCHITECTURES

In this section, we discuss some prevalent CIM and CNM architectures, explaining their programming models and systems integration. It is important to highlight that there exist COM accelerators optimized for specific domains, achieving throughput similar to CNM/CIM counterparts, albeit at the expense of higher energy consumption. These accelerators are beyond the scope of this paper.

This section is structured as follows: In Section 4.1, we provide an overview of CNM systems, starting with academic designs and progressing to commercial CNM systems, including both planar 2D and stacked DRAM structures. Section 4.2 follows a similar organization for CIM systems employing various technologies. In Section 4.3, we conduct a comparative analysis of different CIM/CNM systems, while Section 5.20 outlines the key challenges faced by these innovative architectures.

4.1 CNM architectures

The core principle of compute-near-memory is to perform computations in the memory proximity by placing *processing units* (PUs) on/near the memory chip. The first CNM architecture dates back to the 1990s that aimed at integrating compute units with embedded DRAM on the same chip to achieve higher bandwidth. However, due to technological limitations and costly fabrication processes, even the promising initial CNM proposals like IRAM [40], DIVA [41], and FlexRAM [42] never commercialized.

In recent years, due to the advancements in integration and die-stacking technologies, CNM has regained interest within both industry and academia. PUs are being integrated at different locations within memory devices, including within the memory chip as well as outside the memory chip on the module level, i.e., dual in-line memory module (DIMM). A DIMM typically consists of multiple memory chips, each consisting of multiple ranks, banks, and subarrays. It is worth noting that some researchers also classify PUs integrated at the memory controller level as CNM. However, following the classification and terminology adopted in this report, we categorize it under the COM class.

In the following, we explain some of the common CNM architectures. We start by examining the planar 2D DRAM-based CNM designs, then transition to discussing the 2.5D and 3D DRAM-based CNM systems, and ultimately conclude on the NVM-based CNM architectures.

4.1.1 The UPMEM system. UPMEM is a recent commercial near-bank CNM system and is publicly available [5]. Figure 8 gives a detailed overview of the UPMEM architecture. The memory modules are divided into PIM-enabled memory and main memory (conventional). The PIM-enabled memory combines co-processors known as *data processing units* (DPUs) with conventional DDR4 DRAM on the same die.

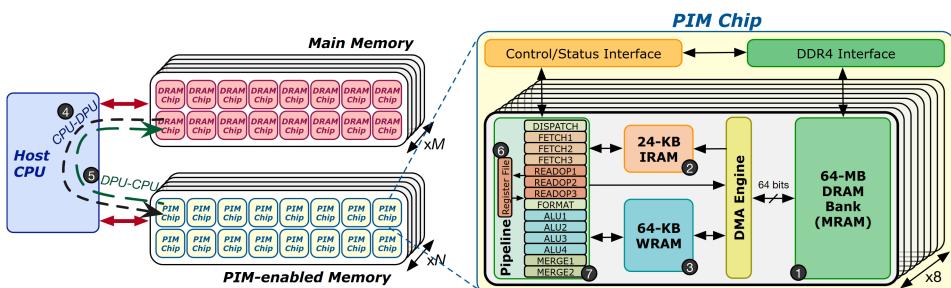


Fig. 8. An overview of the UPMEM architecture [43].

DPUs are 32-bit general-purpose RISC processors, comprising a 64kB SRAM-based scratchpad working memory known as WRAM, a 24kB SRAM-based instruction memory referred to as IRAM, and a shared main memory named MRAM, based on DRAM technology. As shown in the figure (lower left), each DIMM consists of 16 memory chips, with each chip housing 8 banks, and each bank containing one DPU. The latest UPMEM systems can support up to 20 DIMMs.

DPU-DPU communication: DPUs in UPMEM can have up to 24 hardware threads called tasklets.

Within the same DPU, tasklets can share data through MRAM and WRAM, however, DPUs can not communicate with each other directly and must go through the host for any possible data sharing.

Programmability: For programmability, UPMEM offers its own *software development kit* (SDK) consisting of an UPMEM compiler and runtime libraries. DPU programs are written in the C language including specific library calls. The runtime library provides functions for data and instruction transfers between different memory, e.g., MRAM-IRAM, MRAM-WRAM etc.; executing various functions on the DPUs; and synchronization (mutex locks, barriers, handshakes, and semaphores).

Although UPMEM claims they have an easily programmable SDK, programming the system has several challenges. The programmer is responsible for efficient mapping of executions and load-balancing on thousands of DPUs, managing data transfer, and ensuring coherence of data between CPU and DPUs.

4.1.2 CNM for MVM in DRAM. McDRAM [44] and MViD [45] (both involving Samsung Electronics) aimed at accelerating machine learning workloads by embedding *multiply-accumulate* (MAC) units within the DRAM bank. Similar to UPMEM, both McDRAM and MViD utilize 2D DRAM (LPDDR in these cases) and incorporate PUs within the memory banks to exploit the higher internal memory bandwidth. However, unlike UPMEM, these architectures are domain-specific and hence employ fixed functional units (MACs) instead of general-purpose programmable cores.

Figure 9 shows the McDRAM architecture along with the three locations (column decoder, bitline *sense amplifier* (SA)s, and I/O drivers) where MAC units were employed and evaluated. Each McDRAM chip consists of 4 banks and each bank has four 8-bit MAC units. The multiplication is performed in parallel by multiplying rows of the matrix with the input vector.

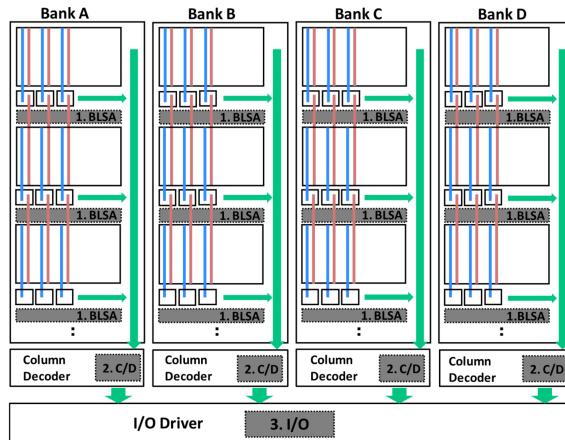


Fig. 9. The McDRAM architecture with three possible locations for MAC units [44].

Programmability: McDRAM is a fixed-function accelerator and offers a single interface function (matmul) that triggers the device driver to configure the control registers of the memory controller. It operates in two modes, memory and compute modes, determined by a configuration register. In compute mode, McDRAM performs MVM tasks. For the management of MVM within compute mode, it introduces six novel DRAM commands that leverage existing DRAM I/O signals, rendering no modifications to DRAM I/O signals. McDRAM, a fixed-function accelerator, employs a single interface function (matmul) triggering the device driver to configure memory controller control

registers. It operates in two modes, memory and compute, determined by a configuration register. In compute mode, McDRAM performs MVM introducing six novel DRAM commands for MVM management without modifying existing DRAM I/O signals.

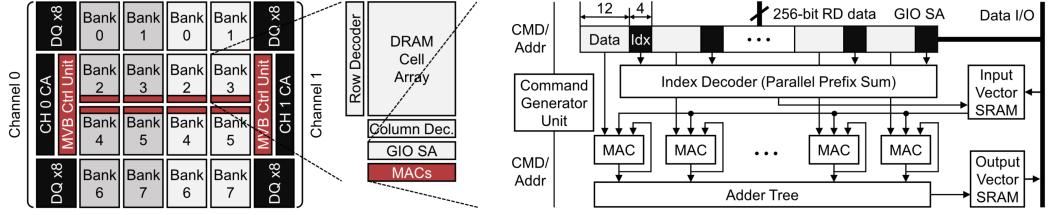


Fig. 10. The MVID architecture [45] where each bank has 16 MAC units.

The MVID architecture depicted in Figure 10 is similar to the McDRAM design and is specifically optimized for edge devices. Much like McDRAM, MVID incorporates MAC units within the DRAM I/O drivers to capitalize on the internal bandwidth of the DRAM. However, unlike McDRAM, MVID introduces a partitioning of memory banks into two categories: MVM banks that are equipped for MAC units and two SRAM structures to hold the input and output vectors and non-MVM banks (traditional). This division enables concurrent access to both types, meaning that multiplication operations in MVM banks can occur simultaneously with the CPU accesses to the non-MVM banks.

4.1.3 Samsung's CNM systems. Samsung is probably ahead of everyone in the race for commercial CNM systems. In the following, we discuss two of their recent promising (and complete) solutions.

PIM-HBM: Samsung has recently introduced a CNM architecture referred to as Function-in-Memory DRAM (FIMDRAM)[6] or PIM-HBM[7]. It incorporates 16 *single-instruction multiple-data* (SIMD) engines within the memory banks, enabling bank-level parallelism. As reported in [7], their design does not disrupt crucial elements on the memory side, such as the sub-array and bank in conventional DRAM, making its integration seamless and straightforward. Importantly, it does not require any modifications to contemporary commercial processor components, including DRAM controllers. It is designed for host processors to manage PIM operations via standard DRAM interfaces. This feature allows for a straightforward substitution of existing JEDEC-compliant DRAM with PIM-DRAM across various systems.

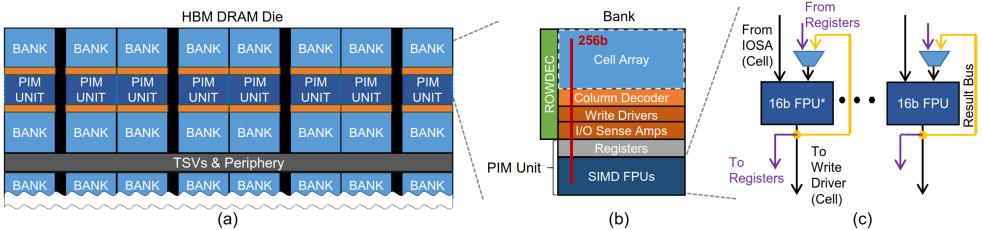


Fig. 11. Samsung's PIM-HBM (a) HBM die organization (b) Bank coupled with a PIM unit (c) PIM unit data path [7].

While Samsung reports that their design is compatible with any DRAM family, they have showcased its functionality using the 2.5D high bandwidth memory (HBM) DRAM. Figure 11 provides a

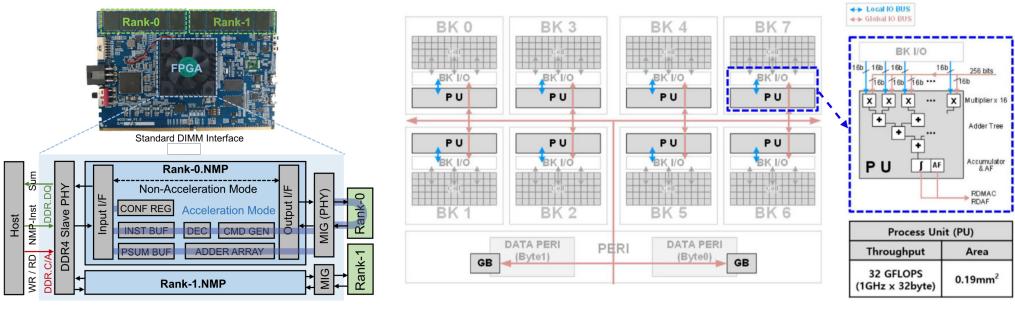
high-level view of this architecture. Each bank comprises 16 SIMD floating-point units (FPUs), with each FPU consisting of a 16-bit floating-point adder and a 16-bit floating-point multiplier. Furthermore, each FPU is equipped with data registers (GRFs), control and instruction registers (CRF, SRF), and an internal control unit. The internal control unit orchestrates operation sequences without necessitating modifications to the memory controller. When operating in PIM mode, the PIM execution units within all banks simultaneously respond to a standard DRAM column (Read or Write) command initiated by the host processor and execute a wide SIMD operation with deterministic latency in a lock-step manner.

Programmability: PIM-HBM comes with an *instruction set architecture* (ISA), a software stack, and a specific programming model. The software stack presents a native execution path that does not require any modifications to the input code. The framework takes the high-level representation of an application and transforms it into device code. Furthermore, it offers a direct execution path that permits direct invocation of various function calls using the “PIM custom op”. The PIM runtime includes a collection of modules responsible for tasks like operations offloading, memory allocation, and execution on the FPUs.

HBM-PIM is a commercial accelerator and, as per, Samsung is already used by companies. Here is an excerpt from Samsung’s newsroom:

“Xilinx has been collaborating with Samsung Electronics to enable high-performance solutions for data center, networking, and real-time signal processing applications starting with the Virtex UltraScale+ HBM family, and recently introduced our new and exciting Versal HBM series products,” said Arun Varadarajan Rajagopal, senior director, Product Planning at Xilinx, Inc. “We are delighted to continue this collaboration with Samsung as we help to evaluate HBM-PIM systems for their potential to achieve major performance and energy-efficiency gains in AI applications.”

AxDIMM (by Samsung-Facebook:) Samsung is also working on the development of an FPGA-enabled CNM platform named AxDIMM. In collaboration with Facebook, this solution has showcased its effectiveness in a personalized recommender system. As shown in Figure 12a, the CNM architecture (RANK) is the same as Samsung’s HBM-PIM, but the controlling unit is FPGA that starts the execution, maps computations to the RANK, and gets back the results. Like the HBM-PIM, AxDIMM has a complete software stack that allows programming the architecture without changing the input code or manually writing code using AxDIMM python API.



(a) Samsung-Facebook AxDIMM hardware module and architecture [46]

(b) AIM architecture [9].

Fig. 12. PUMA tile and core architectures [47].

For this product, Samsung also seems to be in discussion with SAP HANA. Here is another excerpt from the newsroom:

“SAP has been continuously collaborating with Samsung on their new and emerging memory technologies to deliver optimal performance on SAP HANA and help database acceleration,” said Oliver Rebholz, head of HANA core research & innovation at SAP. “Based on performance projections and potential integration scenarios, we expect significant performance improvements for in-memory database management system (IMDBMS) and higher energy efficiency via disaggregated computing on AXDIMM. SAP is looking to continue its collaboration with Samsung in this area”.

4.1.4 SK hynix’s accelerator-in-memory. SK hynix’s accelerator-in-memory (AiM) is another CNM system that targets the machine learning application domain [8, 9]. As stated in [9], “Samsung’s FIMDRAM is near commercialization, but the required HBM technology may prevent it from being applied to other applications due to its high cost”. AiM fundamentally follows a very similar design approach to Samsung’s FIMDRAM but utilizes GDDR6 instead.

Figure 12b provides an overview of the AiM architecture. As depicted, each bank is equipped with a processing unit (PU) that executes a MAC operation using 16 multiplier units and an adder tree. The adder tree can be deactivated for operations not requiring additions. Similar to the FIMDRAM design, pairs of banks can establish direct communication. For inter-group communication, an internal 2KB SRAM structure within the periphery facilitates the process.

Although the programming model is not explicitly explained, the presented set of commands in AiM implies an interface enabling interaction with the device for various operations. Some of these operations are particularly interesting, such as the ability to perform computations within banks of different granularities (1, 4, 16) and data movement functions that can be utilized to implement row-cloning within DRAM.

4.1.5 AxRAM. AxRAM targets optimizing for the off-chip memory communication bottleneck in GPUs by integrating approximate MAC units in the DRAM [48]. The fundamental idea is to exploit the inherent approximability of numerous GPU applications and perform approximate calculations directly within the memory banks, thereby reducing data movement and energy consumption. AxRAM leverages the concept of neural transformation, a technique that accelerates a wide range of applications by approximating specific sections of GPU code and transforming them into a neural representation composed primarily of MAC and *look-up table* (LUT) operations for nonlinear function calculation. The multiplications in the MAC operations are further approximated with limited iterations of shift-add and LUT accesses. These approximate units are connected to the wide data lines that connect the DRAM banks to the global I/O, keeping the banks and memory column unchanged.

Figure 13 shows a sample example where the GPU code is transformed into MAC and lookup operations. Once such patterns are identified and transformed, they are offloaded to the in-DRAM accelerator. The new instructions that invoke and configure the in-DRAM accelerators are added to the GPU’s ISA and are exposed to the compiler. As for the flow of execution, initially, all data is assumed to be in one memory chip. The GPU starts normal execution, and for the identified approximate regions, the GPU warps send an initiation request to the on-chip memory controller. The additional logic in the memory controller first sends an invalid signal to the on-chip caches (to ensure data consistency) and subsequently drives the in-DRAM accelerator to perform the computations and store the results in the designated location. To check whether the execution is completed, the memory controller periodically checks the memory-mapped mode register of the DRAM, which is updated by the accelerator. Once the controller detects that this register is set, it signals the GPU that execution is finalized, allowing the GPU to proceed with precise execution of the subsequent instructions.

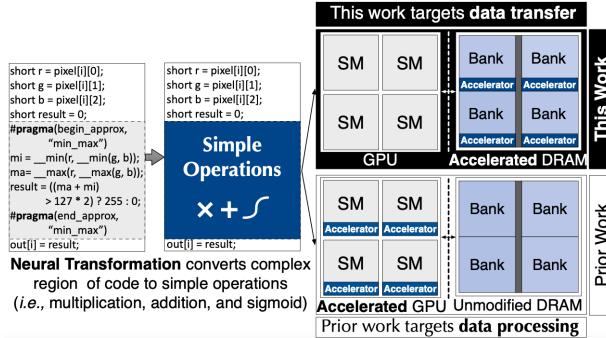


Fig. 13. Transformation of complex code into simple operations (left) and the AxRAM architecture compared to state-of-the-art (right) [48].

4.1.6 CNM systems based on 3D-stacked DRAM. All the CNM architectures discussed so far (except FIMDRAM) are based on planar 2D DRAM. However, the resurgence in CNM systems is also primarily attributed to HBM and HMC technologies that seamlessly combine logic and memory within the same package. There is a series of proposals for CNM systems leveraging these technologies. In the following, we discuss some of the prominent architectures.

TESSERACT [49] targets accelerating graph-based applications. Their design comprises a host processor and an HMC with multiple vaults, each housing an out-of-order processor. These processors exclusively access their local data partition, while inter-communication is achieved through a message-passing protocol. The host processor, however, can access the complete address space of the HMC. To capitalize on the substantial memory bandwidth available, they introduce prefetching mechanisms. **TOP-PIM** [50] is an architecture that proposes an *accelerated processing unit* (APU). Each APU integrates a GPU and a CPU on the same silicon die. These APUs are linked through high-speed serial connections to several 3D-stacked memory modules. APUs are general-purpose and support a series of applications ranging from graph processing to fluid and structure dynamics. The architecture allows code portability and easy programmability.

Active memory cube (AMC) [51] is also built upon HMC and proposes “lanes” in the HMC vault. Each AMC lane consists of a register file, a computational unit, and a load/store unit to support memory accesses. Communication among AMCs is only possible via the host processor. AMC also offers a compiler based on OpenMP for C/C++ and FORTRAN.

Heterogeneous reconfigurable logic (HRL) [52] leverages the logic layer in the 3D stacked HMC to implement heterogeneous coarse-grained (CGRAs) and fine-grained (FPGAs) logic blocks. The architecture separates routing networks for control and data signals, employing specialized units to efficiently handle branch operations and non-uniform data layouts commonly found in analytics workloads.

4.2 CIM architectures

Much like CNM, the concept of CIM systems is not entirely novel; however, it has gained significant momentum due to breakthroughs in various NVM devices over the past decade. Figure 14 shows a partial landscape of CIM systems, along with a corresponding timeline. Most of the depicted CIM accelerators originate from academia and are not taped out. However, in recent years, several semiconductor industry giants, including Intel, Samsung, TSMC, GlobalFoundries, and IBM, have invested in developing their own CIM prototypes, mostly focused on the machine learning case. IBM,

in particular, stands out among others when it comes to the development of CIM systems for different use cases.

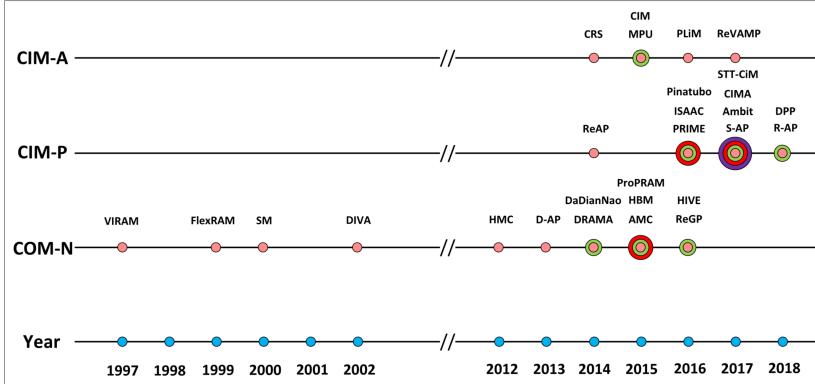


Fig. 14. A partial timeline of the evolution of CIM systems (data until 2018) [28]. The radius of the circle is proportional to the amount of papers published that year.

In this section, we overview some of the prominent CIM designs from academia and industry. However, before going into the details of individual CIM designs, we first introduce circuits that are typically used as basic CIM primitives in these architectures.

4.2.1 CIM primitives. Each of the CIM architectures discussed in the following sections is either based on a crossbar, content-addressable-memory, or a boolean and arithmetic logic unit. In the following, we explain all three of them.

Crossbar: A crossbar is a CIM configuration in which each input connects to every output through cross-points, comprising memory cells and selectors. Figure 15a shows a technology-independent crossbar configuration. As we will see in the following sections, crossbars are particularly useful for the machine learning domain as they can compute MVM in constant time.

CAM: *content-addressable-memory* (CAM) is associative memory that enables parallel searches for a given query (input) across all stored content within a CAM array. CAMs are used in pattern matching and search operations from various application domains including databases, networking, and machine learning [53]. Figure 15c shows a technology-independent 3×3 CAM structure.

Boolean and arithmetic logic in CIM: In this class of CIM, the CIM array facilitates a specific set of general operations, such as Boolean logic and arithmetic, to be executed using customized peripheral circuits integrated within the random-access memory (RAM). The operands need to be stored in different rows of an array in a column-aligned fashion where each column represents a bit position. For a CIM operation, multiple rows are typically activated simultaneously, and the output is sensed and inferred by the peripheral circuitry [23, 54]. Figure 15b shows a technology-independent structure implementing boolean logic.

4.2.2 ISAAC (by Hewlett Packard Enterprise). In-situ analog arithmetic in crossbars (ISAAC) [22] is among the first CIM accelerators with a complete design targeting *convolutional neural network* (CNN) in RRAM. As shown in Figure 16, ISAAC's architecture consists of multiple interconnected tiles via a concentrated-mesh (c-mesh) network. Each tile consists of 12 in-situ multiply-and-accumulate (IMA) units, a shift-and-add (S&A) unit, two sigmoid units, one max-pooling unit, an embedded DRAM (eDRAM) buffer for input data storage and an output register (OR) to accumulate (partial) results. Each

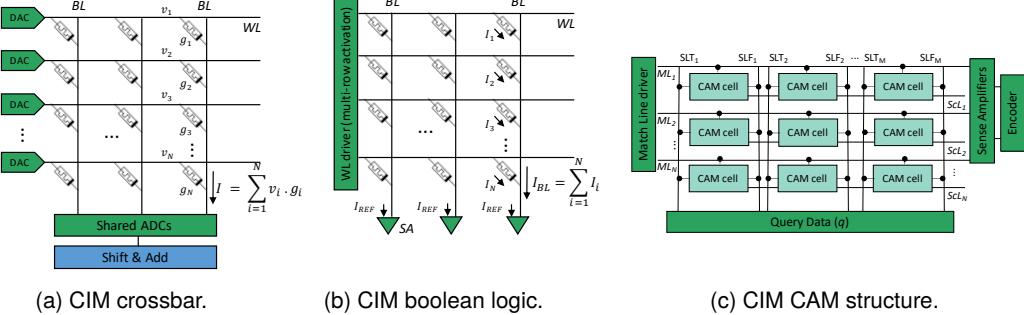


Fig. 15. Fundamental CIM primitives [53].

IMA integrates its own input register (IR), output register, S&A units, and eight 128×128 resistive crossbar arrays, also abbreviated XB or XBars, that share analog-to-digital converters (ADCs). Each XBar performs analog MVM (see Figure 15a) and is also equipped with a digital-to-analog converter (DAC) and an S&H circuitry. Communication within a tile is facilitated by a 32-bit inter-tile link.

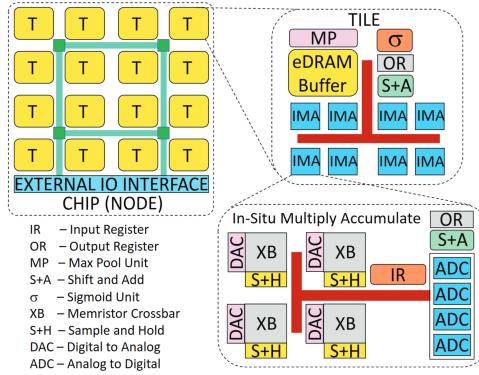


Fig. 16. ISAAC architecture hierarchy [22].

The ISAAC design uses dataflow pipelining to optimize IMA unit utilization and reduce buffering requirements. Depending on the network’s size, each CNN layer is mapped to one or multiple IMAs or tiles. Initially, input data is acquired through an I/O connection and stored within a tile’s eDRAM buffer. Before being fed to ReRAM XBars within each IMA, the data goes through DACs. Once processed by XBars, the generated feature maps are converted back to digital form and forwarded to max-pooling and activation units. The outcome of the NN layer is then accumulated within the S&A and OR units and subsequently written to a new eDRAM buffer (for the following layer). The depth of the pipeline corresponds to the depth of the neural network, which presents challenges when training *deep neural networks* (DNNs). Thus, ISAAC is specifically designed for inference and is not used for training. ISAAC has no mention of the design tools and programming interface.

4.2.3 PUMA (by Hewlett Packard Enterprise). PUMA (programmable ultra-efficient memristor-based accelerator) is a generalization of memristive crossbars to accelerate a range of ML inference workloads [47]. PUMA’s microarchitecture techniques exposed via dedicated ISA ensure the efficiency of in-memory computing and analog circuitry while providing a high degree of programmability. The

architecture is organized into three hierarchy levels: cores, tiles, and nodes. Nodes are connected and communicate via a chip-to-chip network. Each individual node consists of tiles that are connected via an on-chip network, where each tile comprises cores that communicate via shared memory, as shown in Figure 17a. A PUMA’s core consists of its own memory and functional units, including the XBar array, referred to as the MVM unit (MVMU), see Figure 17b.

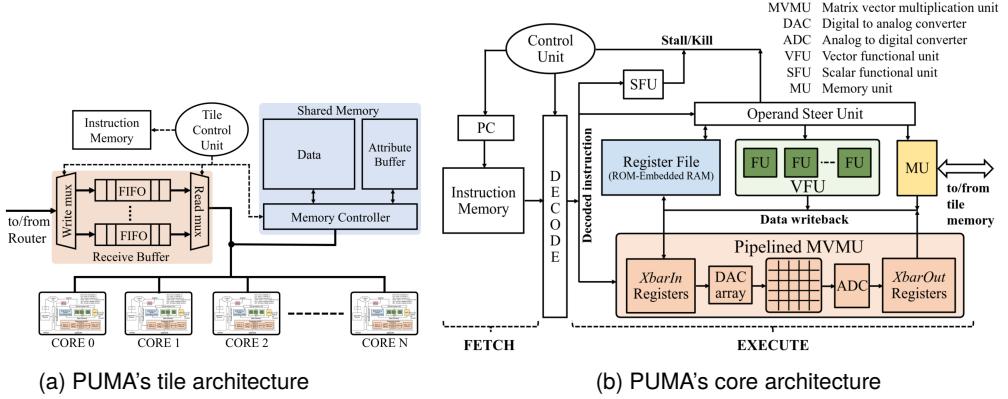


Fig. 17. PUMA tile and core architectures [47].

Unlike most other CIM architectures, which are data parallel, PUMA is a spatial architecture where distinct instructions are executed by each core or tile. Since manually writing code for such architectures is extremely difficult, particularly when they have thousands of cores, PUMA has a runtime compiler implemented as a C++ library. The compiler takes the high-level input code and extracts a dataflow graph from it. The graph is then divided into subgraphs, considering the sizes of MVMUs, and hierarchically assigned to MVMUs, cores, and tiles. The subgraph execution is carefully scheduled to ensure effective resource utilization while avoiding potential deadlocks. Given the constraint of serial read and write operations in RRAM, PUMA exclusively supports the ML inference. However, to facilitate training, PUMA has been repurposed in a follow-up work named PANTHER [55].

4.2.4 Pinatubo: Accelerating bulk bitwise logic operation. Pinatubo is a memristor-based architecture that harnesses data-level parallelism to conduct bulk bitwise operations [23]. Unlike the crossbar configurations, it performs computations in the digital domain by modifying the SAs. The system architecture is similar to a typical Von Neumann architecture that has a processor equipped with caches and a non-volatile main memory. Pinatubo then exploits the physical attributes of the NVM-based main memory and modifies the SAs to support it. The main idea is the operands are stored in different rows but the same columns in an array, the rows are activated in parallel and the accumulated current in the bitline is compared to a reference level in the SAs. For different logic gates, the memory controller changes the reference levels in the SAs to different stats.

The Pinatubo’s main memory structure is illustrated in Figure 18, comprising multiple banks divided into banks and mats. For operands within the same mat, the modified SAs work out of the box and can perform bitwise vector operations. For operations where the data is spread across different mats, whether within the same bank or not, additional logic gates are used for execution (within the global data line or global I/O). The architecture supports only logic operations.

For programmability, Pinatubo presents a software infrastructure containing both the programming model and runtime support components. The programming model offers two functions to allocate

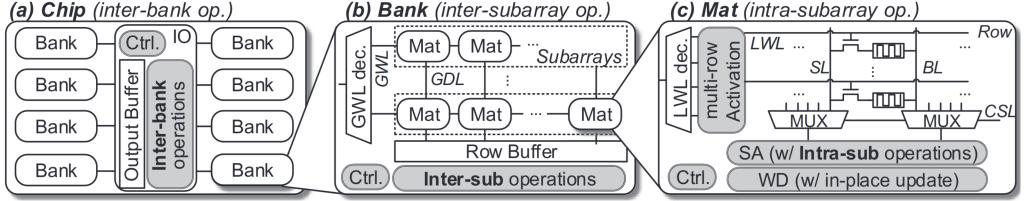


Fig. 18. Pinatubo architecture showing chip, bank and mat [23].

bit-vectors and perform bitwise operations. The runtime support facet encompasses adjustments to the C/C++ runtime library and the operating system (OS) and the development of a dynamic linked driver library. The runtime library ensures that bit-vectors are allocated to separate memory rows while the OS equipped with PIM-aware memory management, ensures intelligent invocation of the operations.

4.2.5 PRIME. PRIME [21] is another RRAM-based analog CIM accelerator. The architecture comprises multiple banks where each bank integrates eight subarrays (chips) which are further (logically) split into memory (Mem) units, two full function (FF) units, and one buffer. FFs can function conventionally as memory or in an NN computation mode, controlled by the PRIME controller. A typical FF unit is 256×256 RRAM cells, with 6-bit reconfigurable local SAs reading their outputs. During computation mode, RRAM resolution is 4-bit *multi-level cell* (MLC), shifting to *single-level cell* (SLC) in memory mode. Distinct crossbar arrays are utilized for storing positive and negative weights. The input to the mat comes from a 3-bit fixed point signal originating from a wordline decoder and driver (WDD). Analog subtraction and sigmoid functions within the NN are implemented in the modified column multiplexers within the RRAM arrays.

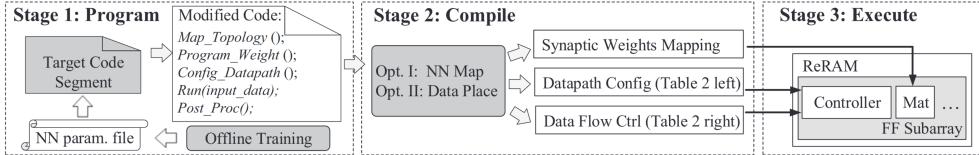


Fig. 19. PRIME: Source code to execution [21].

The execution of an *neural-network* (NN) on PRIME involves three stages. Firstly, the NN is mapped onto FF subarrays, and synaptic weights are programmed into ReRAM cells. In the optimization stage, depending on the NN size, mapping could occur in a single bank or across multiple banks. These first two stages are executed by the CPU. Subsequently, a series of generated instructions are transmitted to the PRIME controller in RRAM banks to perform computations. The presence of latches and OR gates facilitates pipelined computation within PRIME.

As shown in Figure 19, PRIME also comes with a compiler and an API, exposing device capabilities as function calls. The process from code to execution involves programming (coding), compiling (code optimization), and code execution. PRIME offers application programming interfaces (APIs) that empower developers to map NN topologies onto FFs and configure data paths etc.

4.2.6 Pipelayer. Pipelayer is another RRAM-based accelerator for CNNs that supports both training and inference [56]. The overall architecture of PipeLayer, shown in Figure 20 features RRAM crossbars, the spike Driver block to encode inputs as spikes and get rid of DACs, and integration

and fire components that eliminate ADCs. In write mode, the spike driver updates RRAM array weights with a 4-bit resolution. Within the cell, data processing occurs across morphable and memory subarrays, where memory subarrays are conventional memory arrays while morphable arrays can be configured in both compute and memory modes. Pipelayer leverages these morphable subarrays for different purposes in training and inference.

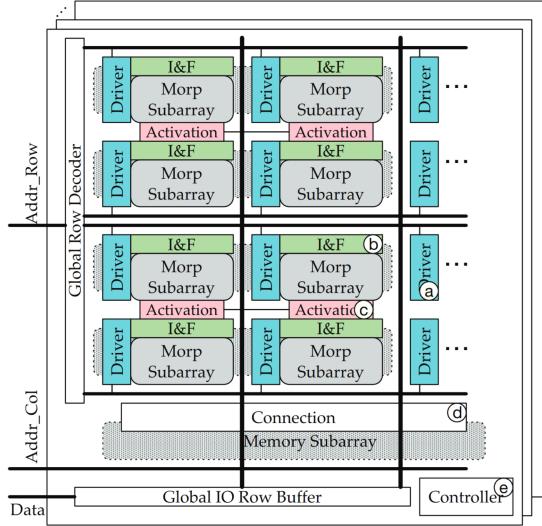


Fig. 20. An overview of the Pipelayer architecture [56].

PipeLayer allows interactive configuration of the system on a per-layer basis. It provides an API that has functions for different operations e.g., bidirectional transfer of data between the CPU main memory and PipeLayer, the topology_set function, where the number of compute groups can be specified by the programmer, the weight_load function to load either pre-trained weights during testing or initial weights during training into the arrays. Other functions include pipeline and mode set functions for the morphable subarrays.

There are many other RRAM-based analog and digital CIM accelerators. Some other common ones that are mostly taped-out and not discussed here include: AtomLayer [57], RIMAC[58], FORM [59], RRAMs for pattern recognition [60], RRAM accelerator for BNNs (ISSCC, 65 nm) [61], RRAM for edge processors (ISSCC, 55 nm) [62], analog RRAM with fully parallel MAC and extremely high TOPS/W (ISSCC, 130 nm but large array) [63].

4.2.7 In-DRAM computing. Ambit [25] is a DRAM-based CIM accelerator that, unlike all previous systems, leverages the analog capabilities of current DRAM technology for executing bulk bitwise operations. Ambit mainly comprises two components. First, Ambit-AND-OR implements *triple row activation* (TRA) in conventional DRAM. Like memristors, the idea is to activate three rows in parallel and leverage the *charge-sharing and charge accumulation principle*. TRA produces a bitwise majority function. Controlling the initial state of one of the three rows enable performing AND and OR operation. The second component of Ambit is Ambit-NOT, which uses the inverters in the DRAM SAs to implement the logic NOT operation. The basic components are then extended to implement other logic operations and accelerate bulk bitwise operations in multiple applications. With 8 DRAM banks, Ambit demonstrates a substantial improvement in bulk bitwise operation throughput compared to an Intel Skylake processor and the NVIDIA GTX 745 GPU.

A follow-up work on the bulk bitwise logic in DRAM, ComputeDRAM [29] demonstrated that by deliberately violating timing parameters between activation commands, certain existing off-the-shelf DRAM chips can implement the TRA operation of Ambit. This indicates that certain real-world off-the-shelf DRAM chips, despite not being intended for Ambit operations, can indeed perform in-DRAM AND and OR operations. This also suggests that the concepts introduced in Ambit might not be too far from practical implementation. If existing DRAM chips can perform such operations to some extent, then chips explicitly designed for such functions could potentially be even more capable.

4.2.8 In-SRAM computing. Neural Cache [64] is an SRAM-based CIM accelerator primarily targeting CNNs. The core operations of Neural Cache are bitwise AND and NOR operations, which are executed by simultaneously activating multiple rows (charge sharing). It repurposes the cache memory by modifying the peripheral circuitry to support operations such as convolution, pooling, quantization, and fully-connected layers, all performed at an 8-bit data precision. It is also capable of performing bit-serial operations like addition, subtraction, multiplication, comparison, search, and copy for larger data, utilizing carry latches linked to SAs. A transpose memory unit is introduced that facilitates the reorganization of data into bit-serial format within the memory when needed.

IMAC [30] is another SRAM-based CIM accelerator that uses the precharge circuit to perform multi-bit analog multiplication by encoding the bit significance in the pulse width of pre-charge pulse. IMAC also requires DAC/ADC converters to facilitate the conversion between digital and analog forms. There are many other instances of SRAM-based CIM designs, some even **taped-out** [65–68].

4.2.9 In-MRAM computing. In NVMs, Magnetic RAM (MRAM) is probably the most mature memory technology that is commercially available and is already used in many embedded devices (see Section 3.6). Therefore, it has also been intensively investigated in the CIM context and computing approaches implementing in-MRAM basic boolean logic operations and more complex arithmetic functions have been showcased. Like all other technologies, the basic CIM methods include bit-cell modification, reference adaptation, and in-memory analog computation.

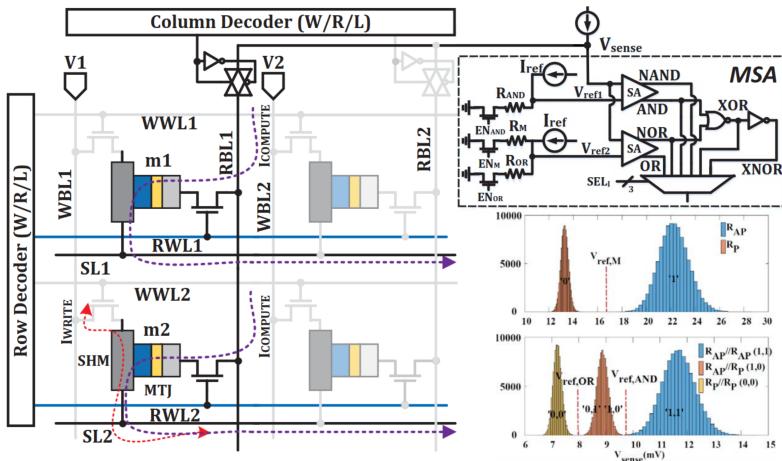


Fig. 21. A typical SOT-MRAM subarray architecture for in-place logic [69].

Figure 21 shows a typical subarray architecture of an in-MRAM CIM [69]. The important difference here compared to already known aspects is that it has separate read and write bit and word lines and three reference resistance states in the sensing circuitry (RAND/RM/ROR). RM is used to perform

normal memory operations, while RAND and ROR, as the names suggest, are used to implement AND and OR operations, respectively.

Similar to other technologies, the boolean logic is implemented with charge-sharing, and MAC is implemented in the analog domain with current accumulation. Some prominent MRAM-based CIM designs include analog MACs for TRA inference [70], MRAM-CIM for BNNs [71], and MRAM crossbar [72].

4.2.10 CIM using FeFETs. FeFETs have also been shown to implement in-place logic gates, addition, and content-addressable memories (CAMs). Notably, these logic operations can also be implemented with a single FeFET cell. For instance, if one operand is stored in a cell (or a set of cells), the other operand can be applied as input to perform logic operation [73], akin to the working principle of crossbars. Further, solutions proposing activating multiple rows and leveraging the bitline's charge sharing (as in other memory technology) have also been presented [37].

FeFETs have received particular interest in CAM designs. CAMs are associative memories that can perform parallel searches for a query across all stored contents within an array. FeFETs have been used to implement different types of CAMs for exact-search operations, approximate search operations, range-based search operations, or a combination of them [74, 75].

4.2.11 Latest industrial chips. In the previous sections, we have extensively discussed a variety of notable CIM and CNM solutions employing different technologies. While a few of these systems have been developed in collaboration with industry partners and a subset has undergone the tape-out process, the majority of these accelerators originate from academia. In this section, we specifically present CIM systems originating from the industrial sector in the last couple of years. Note that these CIM systems also primarily show prototypes showcasing various research outcomes, but they indicate their potential realization in the near future.

IBM’s PCM-based accelerators: For more than five years, IBM has been using its PCM device to do in-place operations for different use cases. Initially, they were working with a reservoir of devices (millions of them) and implementing the peripheral circuitry and additional CMOS logic in an FPGA. Their research has progressed to consolidate all components onto a single chip, as exemplified by HERMES, a core composed of 256×256 PCM array with ADCs, a local digital processing unit, and additional peripheries [76]. The core effectively executes a fully parallel 256×256 analog MVM, where each 8T4R unit cell encodes a positive/negative weight, with simultaneous handling of 256 8-bit digital inputs/outputs. Positive weights are encoded by combining the conductance of two PCM devices, while negative weights are represented by the other two PCMs within the unit cell.

This year, IBM announced a newer 64-core CIM chip designed and fabricated in 14-nm CMOS technology integrated with PCM [77]. The fully integrated chip comprises 64 cores, each with a size of 256×256 , connected through an on-chip communication network. It reportedly achieves an unparalleled maximal throughput of 63.1 TOPS at an energy efficiency of 9.76 TOPS/W for 8-bit input/output MVMs.

Samsung’s MRAM crossbar: Crossbar-based analog MVM is well-explored in RRAM and PCM technologies. However, implementing MRAM-based crossbars is challenging due to the inherent low resistance of these devices, which could lead to significant power consumption. In 2022, Samsung presented a 64×64 MRAM crossbar array to address the low-resistance issue by employing an

architecture that uses resistance summation (instead of current summation) for analog multiply-accumulate operations [72]. Compared to the IBM HERMES cores, Samsung's crossbar is significantly less sophisticated and limited in scale.

TSMC's in-SRAM accelerator: While other SRAM-based CIM chips exist, our focus is on the TSMC macro structure using standard 8T cells [78] due to its better noise margin, ensuring stable activation for multiple rows operations in the CIM mode, albeit with approximately 30% increased area.

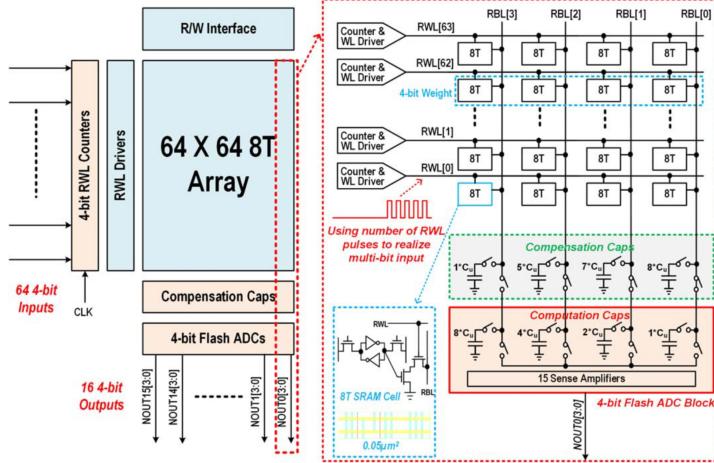


Fig. 22. TSMC's CIM SRAM structure [78].

The proposed design shown in Figure 22 has a 64×64 SRAM array and enables parallel computations of the multiply-and-average (MAV) operations. In a single cycle, the MAV computation of 64 4-bit inputs with 16 4-bit weight can be completed. The 4-bit input is represented by the number of read word line pulses which is precisely controlled by 4-bit digital counters. The 4-bit weight is achieved through charge sharing across binary-weighted computation capacitors. Each computation capacitor unit is constructed using the inherent capacitor of the SA within the 4-bit flash ADC to optimize space and minimize the kick-back effect. This 64×64 8T macro is fabricated using 7nm FinFET technology, exhibiting an energy efficiency of 351 TOPS/W and a throughput of 372.4 GOPS for 1024 (64×16) 4x4b MAV operations.

Intel's SRAM-based analog CIM design: Intel has recently proposed an SRAM-based CIM macro utilizing their 22nm Low-Power FinFET process [79]. Through the implementation of a 1-to-2 ratioed capacitor ladder (C-2C)-based charge domain computing scheme, the presented prototype chip (shown in Figure 23) achieves the capability to perform up to 2k MAC operations in a single clock cycle, alongside achieving a peak power efficiency of 32.2-TOPS/W with 8-bit precision for both input activation and weights. The chip also ensures accurate MVMs by restricting the computation error of less than 0.5%.

Bosch+Fraunhofer and GlobalFoundries+Fraunhofer FeFET based CIM designs: Fraunhofer is also actively working on exploring the manufacturability and scalability aspects of FeFET and MRAM devices at both the device and array levels. Together with GlobalFoundries, they have demonstrated a

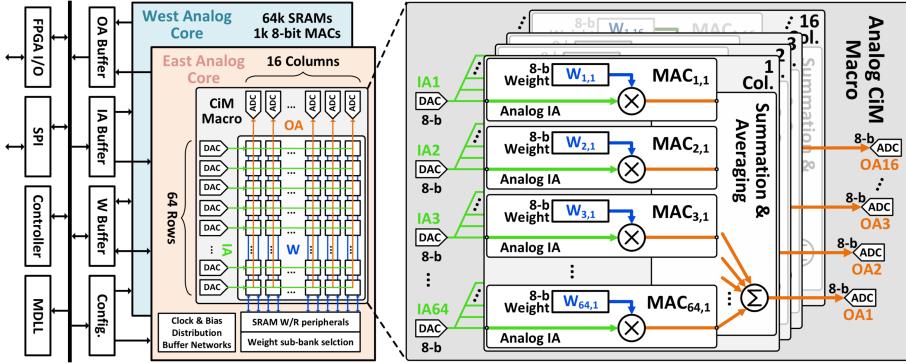


Fig. 23. Chip level architecture diagram of Intel's analog CIM design [79].

FeFET-based crossbar array for multiply-accumulate (MAC) operation [80]. The array was fabricated at GlobalFoundries with 28nm CMOS technology coupled with FeFET. To prevent the accumulation of errors on the bitline, the arrays were divided into 8×8 segments.

In a recent work, Fraunhofer and Robert Bosch demonstrated a CIM crossbar using multi-level FeFET cells. In the proposed design, the input is encoded into the applied voltage duration and magnitude while the weights are stored in the multi-level FeFET cells. The MAC output is the accumulated capacitor voltage that depends on the activation time and the number of FeFETs activated. This reportedly reduces the impact of variations and the achieved performance of 885.4 TOPS/W is also nearly-double compared to existing solutions.

HP's CAM designs: In a recent work, Hewlett Packard Labs proposed a memristive-based analog CAM for tree-based machine learning [81]. Analog CAMs are capable of performing searches based on analog signal levels rather than digital data comparison. The proposed design combines analog CAMs with traditional analog RAM and accelerates large random forest models with it. Figure 24 shows a high-level overview of the proposed system where the analog CAM can perform root-to-leaf evaluation of an entire tree in a single step.

4.3 Comparative analysis and discussion

The surge in CIM and CNM systems is largely attributed to the revolution in data-intensive applications. According to recent research by TSMC, traditional SRAM and DRAM technologies have effectively scaled to meet capacity and bandwidth demands in the past decades [82], but their future scalability is uncertain due to reaching inherent technological limits. This underscores the pivotal role that NVM will play in the future of computing.

Especially in edge scenarios such as automotive, augmented reality, and AI, where energy efficiency is paramount, NVM technologies are poised to play a pivotal role. As energy efficiency increases through specialized hardware, domain-specific architectures harnessing these NVMs for CIM and CNM solutions are anticipated to experience an unprecedented surge in the coming years. A recent article from Intel [79] compares the performance of conventional digital accelerators with the emerging analog and digital CIM and CNM accelerators. Conventional accelerators still achieve higher throughput because CIM systems are relatively less optimized, array sizes are small, and the peripheral circuitry overhead is non-negligible. Yet, they are orders of magnitude better in terms of

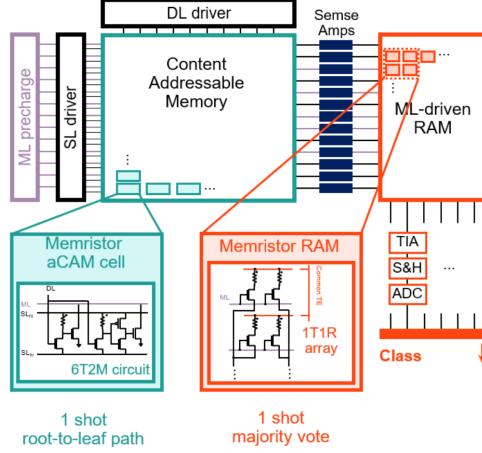


Fig. 24. An overview of the HP CIM system for tree-based learning [81].

power consumption. As of the time of writing, the most recent comparison depicted in [82] shows similar trends.

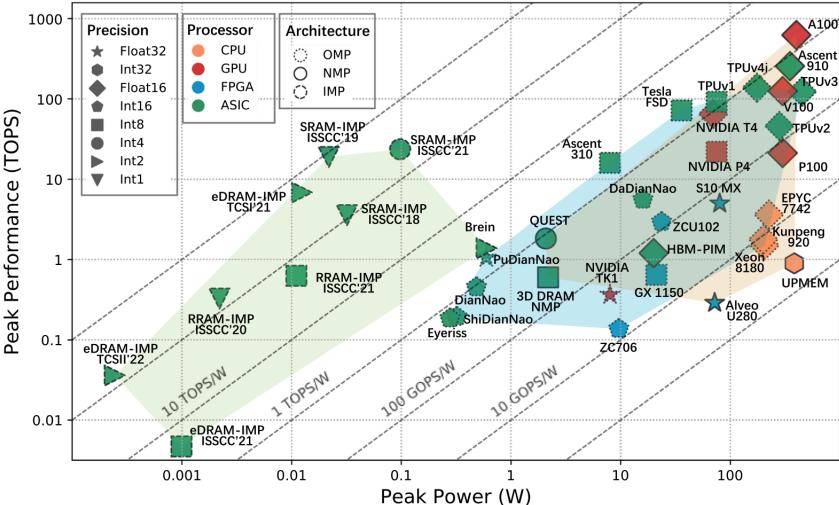


Fig. 25. Performance and power comparison of different *outside memory processing* (OMP) (we call it COM in this report), NMP (ours CNM) and IMP (ours CIM) [83].

Table 2 presents a summary and comparison of the architectures discussed in this section. For brevity, we only compare important parameters, such as the underlying memory technology, available function (boolean logic, arithmetic, etc.), evaluation technique (simulation, prototype, analytic), programming model, application domain, and technology node.

5 COMMERCIAL LANDSCAPE

This section overviews CIM and CNM companies/startups, highlighting their products, underlying technologies, customers (when known), and tools. As not everything about companies is public,

Accelerator	Year	Technology	Type	Programming model	Logic unit	Implementation	Domain
McDRAM	2018	DRAM	CNM	Extended ISA	MAC	Hardware	AI
MViD	2020	DRAM	CNM	Extended ISA	MAC	Hardware	AI
PIM-HBM	2021	DRAM (HBM)	CNM	Full software-stack	FPU (add, Mul)	Hardware	AI
AiM	2022	DRAM (GDDR6)	CNM	API	MAC	Hardware	AI
AxRAM	2018	DRAM (GPU-based system)	CNM	API	MAC, LUTs	GPGPU-Sim	AI
TESSERACT	2015	DRAM (HMC)	CNM	API	CPU	Simulation	Graph processing
TOP-PIM	2014	DRAM (HMC)	CNM	OpenCL	CPU+GPU	Simulation	Graph, HPC
AMC	2015	DRAM (HMC)	CNM	OpenMP	CPU	Simulation	HPC
HRL	2015	DRAM (HMC)	CNM	MapReduce	CGRA+FPGA	Simulation	Data analytics
CIM architectures (Academia/Research Labs)							
ISAAC	2016	RRAM	CIM	NA	Analog Xbar	Analytical	AI
PUMA	2019	RRAM	CIM	Compiler	Xbar	PUMAsim (arch. simulation)	AI
Pinatubo	2016	RRAM	CIM	API, Runtime	Boolean logic	In-house simulator	Bitwise Logic
PRIME	2016	RRAM	CIM	Compiler+API	Xbar	Analytical	AI
PipeLayer	2017	RRAM	CIM	API	Xbar	Analytical	CNN (train + infer)
AtomLayer	2018	RRAM	CIM	NA	Xbar	Analytical	CNN (train + infer)
RIMAC	2023	RRAM	CIM	NA	Xbar (without DAC/ADC)	In-house simulator	DNN inference

Table 2. A summary of the presented architectures. They are grouped into three categories: CNM, CIM, and CIM (prototype chips/systems). All presented architectures are either simulation-based or prototype-based (no products).

we only include details that we extract from these companies’ websites or are known to us via our network.

5.1 Axelera

Axelera [84] is one of the notable Semiconductor startups in Europe. Founded in 2021 and backed by tech giants like Bitfury and IMEC, it had already taped out its first CIM chip, Thetis, in December 2021 (just four months after its founding). Today, it offers a fully integrated *system-on-chip* (SoC) powered by its Metis AI processing units (AIPU).

About the AI core, as per the company’s website: “Axelera AI has fundamentally changed the architecture of “compute-in-place” by introducing an SRAM-based digital in-memory computing (D-IMC) engine. In contrast to analog in-memory computing approaches, Axelera’s D-IMC design is immune to noise and memory non-idealities that affect the precision of the analog matrix-vector operations as well as the deterministic nature and repeatability of the matrix-vector multiplication results. Our D-IMC supports INT8 activations and weights, but the accumulation maintains full precision at INT32, which enables state-of-the-art FP32 iso-accuracy for a wide range of applications without the need for retraining”.

Axelera’s latest SoC consists of 4 cores and a RISC-V based control core. For programming these systems, Axelera provides an end-to-end integrated framework for application development. The high-level framework takes users along the development processes without needing to understand the underlying architecture or even the machine learning concepts.

Funding: “Axelera AI, the provider of the world’s most powerful and advanced solutions for AI at the Edge, announces new investors who have joined their oversubscribed Series A round, bringing the total amount raised to \$50 million. In the last several months, CDP Venture Capital, Verve Ventures, and Fractionelera have joined the round”, Axelera AI, May 22, 2023.

5.2 d-Matrix

d-Matrix is at the forefront of driving the transformation in data center architecture toward digital in-memory computing (DIMC) [85]. Founded in 2019, the company has received substantial support from prominent investors and strategic partners, including Playground Global, M12 (Microsoft Venture Fund), SK Hynix, Nautlius Venture Partners, Marvell Technology, and Entrada Ventures.

Leveraging their in-SRAM digital computing techniques, a chipset-based design, high-bandwidth BoW interconnects, and a full stack of machine learning and large language model tools and software, d-Matrix pioneers best-performing solutions for large-scale inference requirements. A full stack framework, compiler, and APIs (open-source as per the company’s website but couldn’t find the link). Their latest product Jayhawk II can scale up to 150 TOPS/W using 6nm technology and can handle LLM models up to 20× more inferences per second for LLM sizing to 40B parameters, compared to state-of-the-art GPUs.

Funding: Temasek, Playground Global and Microsoft Corp.

5.3 Gyrfalcon Technology

Gyrfalcon Technology [86] also leverages CNM to accelerate AI on the edge. They offer an AI processing in memory (APiM) architecture that combines a large MAC array directly with MRAM memory modules. As of the current date, their software stack is not available.

Funding: Private.

5.4 MemComputing

MemComputing [87], founded in 2016, uses a computational memory based on its self-organizing logic gates (SOLG). SOLGs are terminal-agnostic elements (memristor or memcapacitor) that implement various logic gates. Their target applications comprise industrial computations associated with optimizations, big data analytics, and machine learning. MemComputing provides a software stack and offers it as a software-as-a-service.

Funding: MemComputing mentions the US Space Force, ENSOS, NASA, Ball Aerospace, PSA, US Air Force, Canvass Labs and Defence Innovation Unit as partners.

5.5 Memverge

Memverge [88] is not directly doing any CIM or CNM but is relevant in the context. Backed by 9 investors including tech giants like Intel, SK hynix, the company’s main goal is to provide software designed to accelerate and optimize data-intensive applications. Their main target is to consider environments with “Endless Memory” and efficiently manage the memory to get more performance.

Latest news: “Samsung, MemVerge, H3 Platform, and XConn, today unveiled a 2TB Pooled CXL Memory System at Flash Memory Summit. The system addresses performance challenges faced by highly distributed AI/ML applications. These challenges include issues like spilling memory to slow storage when main memory is full, excessive memory copying, I/O to storage, serialization/deserialization, and Out-of-Memory errors that can crash an application.”, MemVerge, August 8, 2023.

5.6 Mythic

Mythic [89] offers an analog matrix processor (Mythic AMP) that uses their analog compute engine (ACE) based on flash memory array and ADCs. Mythic ACE also has a 32b RISC V processor, SIMD vector engine, and a 64KB SRAM along with a high-throughput network-on-chip (NoC). Mythic workflow in Figure 26 shows that the software stack takes a trained NN model, optimizes it, and compiles it to generate code for Mythic AMP. The optimization suit also transforms NN in a way that can be accelerated on the analog CIM system.

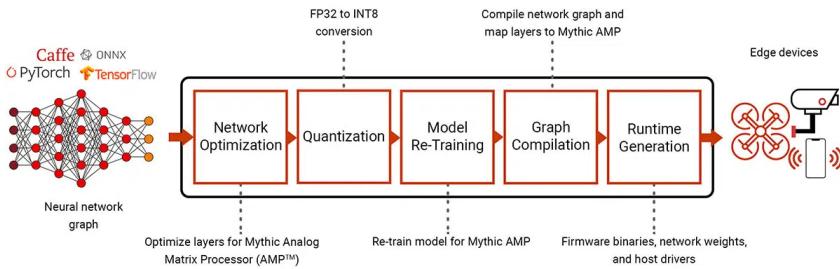


Fig. 26. Mythic AI workflow [89].

Funding: The company is supported by many investors: Micron, HP Enterprise, SoftBank, Future ventures, Lam Research, Threshold, Catapult, DCVC and UDC ventures.

5.7 NeuroBlad

Founded in 2018, NeuroBlade offers the SPU (SQL Processing Unit), the industry's first, proven processor architecture that delivers orders of magnitude improvement by departing from Von Neumann model [90]. Neuroblade is also a CNM architecture (more closed to near-storage computing) where they integrate custom RISC processors on the DRAM chip (very similar to UPMEM). The SPUs are installed as PCI-e cards that can be deployed in data centers. As for the software stack, the company offers an SDK along with a set of APIs that hide the complexity and programming model for these cores from the end user and also allow optimizing for maximum parallelism and efficiency.

Funding: NeuroBlade is funded by Stage one, Grove Ventures, UMC, PSMC Intel capitals, Pegatron, Marubeni, Marius Nacht, Corner and MediaTek.

5.8 Rain AI

Founded in 2017, Rain AI also focuses on radically cheaper AI computing [91]. The company has no hardware product yet but is aiming to be 100x better than GPU using their innovations in radical co-design (by looking at the algorithms and the CIM hardware at the same time). They are targeting AI training (along with the inference) on the edge with the ultimate goal of putting models the size of ChatGPT into chips of the size of a thumbnail. They are transforming the algorithms in a way that fundamentally matches the behavior of the analog memristive devices. As per the CEO, they have a few tap-outs planned for this year and the product (a complete platform) next year and they are working on a software stack for ease of use and ease of integration.

Funding: The company is funded by Y combinator S18, Sam Altman (CEO OpenAI), Liquid 2 Ventures, Loup Ventures, Airbus Ventures, and Daniel Gross (founder Pioneer).

5.9 SEMRON

Founded in 2020, Semron [92] promises to offer 3D solutions powered by analog CIM. At the core of their technology is their innovative CapRAM devices which are semiconductor devices that store multi-bit values in their variable capacitances (unlike variable resistance states in memristors). Since CapRAM is capacitive, the noise in calculations is much lower and the energy efficiency, as per their website, is unparalleled. Although Semron has the device technology, there are no details of its products, architecture, and software stack.

Funding: As per crunchbase, the company is funded by VentureOut.

5.10 SureCore

Surecore [93] is working on many low-power products including custom application-specific. They also have a product named “CompuRAM” that embeds arithmetic capability within the SRAM array to enable low-power AI on the edge. Besides working on SRAM-based solutions, in collaboration with Intrinsic, they have recently ventured into RRAM technology. No information is provided regarding the software stack.

Funding: The company is supported by Capital-E, Finance Yorkshire and Mercia Technologies.

5.11 Synthara

Synthara is a Zurich-based Semiconductor company that was founded in 2017 [94]. Their latest product, ComputeRAM, integrates SRAM-based CIM macros with proprietary elements to accelerate dot products. The solution delivers 50× compute efficiency and can be used for AI, digital signal processing, and linear algebra-heavy routines. The CIM-powered SRAM array can be operated just like conventional SRAM. ComputeRAM is not married to a specific ISA and can work with any host processor. Synthara also provides what they call Compiler hooks that can transparently offload any input application to their ComputeRAM accelerator, without changing or rewriting the code.

Funding: The company is supported by EU funding for research & innovation, High-tech Gründerfonds, Intel.ignite, FNSNF, multicoreware, ventureKick and others.

5.12 Syntiant

Founded in 2017, Syntiant also leverages DRAM-based CNM and utilizes standard CMOS processes to design their neural decision processors (NDPs) that perform direct processing of neural network layers from platforms like TensorFlow [95]. Syntiant also mainly targets AI on the edge having applications in many domains, including always-on voice, audio, image, and sensor applications.

Syntiant’s TinyML platform, powered by NDP101, aspires to democratize AI by presenting a comprehensive system for those interested in initiating their own model training for edge computing.

Funding: Syntiant is funded by prominent investors including, Atlantic Bridge, Rober Bosch Venture Capital, Embark Ventures, DHVC, Intel capitals, M12 (Microsoft ventures), and Motorola Solutions.

5.13 TetraMem

Founded in 2018, TetraMem is set to offer the industry’s most disruptive CIM technology for edge application [96]. TetraMem is also leveraging memristors for analog MAC operations, aiming at inference on the edge. Their systems are built upon their patented devices and co-design solutions.

TetraMem offers (1) Platform as a service (PaaS), a complete hardware and software platform designed to integrate into your own system; (2) Software as a service (SaaS), to help develop your NN edge application and integrate it into your system. Their verified full software stack provides an

unmatched experience on actual analog in-memory compute silicon; and (3) a neural processing unit (NPU) based on memristive technology.

TetraMem has recently announced a collaboration with Andes Technologies and together with their research collaborators have demonstrated a memristive device that can have thousands of conductance levels (unmatched) [97].

Funding: Private.

5.14 EnCharge AI

Founded in 2022, EnCharge AI promise to offer an end-to-end scalable architecture for AI inference [98]. They leverage SRAM-based CIM arrays for analog MVM operations and combine them with SIMD CNM logic to perform custom element-wise operations. The architecture comprises an array of CIM units (CIMUs), an on-chip network interconnecting CIMUs, buffers, control circuitry, and off-chip interfaces. Each CIMU is equipped with an SRAM-based CIM array featuring ADCs to convert computed outputs into digital values. Additionally, CIMUs house SIMD units and FPUs with a custom instruction set, along with buffers dedicated to both computation and data flow. According to the company's official website, they offer a software platform that fits with standard ML frameworks, such as PyTorch, TensorFlow, and ONNX. This also allows the implementation of various ML models and their customizations. Specific implementation details about the software stack are not available.

Funding: Encharge AI is funded by AlleyCorp, Scout Ventures, Silicon Catalyst Angels, Schams Ventures, E14 Fund, and Alumni Ventures. At their launch in December 2022, they announced securing \$21.7 Mio. in their series A round.

5.15 Re(conceive) AI

Re(conceive) is another CIM startup founded in 2019 that promises offering “the most power AI accelerator” [99]. As per their website, re(conceive) are pioneers in realizing the complete potential of CMOS-based analog in-memory AI computing, achieving the utmost efficiency among all known AI accelerators. However, no specific details are available on the company's funding and technology (hardware/software).

5.16 Fractile AI

Established in 2022 by a team of Oxford University scientists, Fractile [100] aims to transform the world by enabling large language models' (LLM) inference at speeds up to 100 times faster than Nvidia's most recent H100 GPUs. This increase in performance primarily arises from in-memory computations. However, the details of the technology, both hardware and software, as well as the company's funding particulars, remain undisclosed.

5.17 Untether AI

Founded in 2018 [101], Untether's main design integrates RISC-V cores on the SRAM chips for processing AI workloads. Their latest product, the tsunAImi accelerator card provides a phenomenal 2 POPS of compute power, twice the amount of any available product. This compute power translates into over 80,000 frames per second of ResNet-50 throughput, three times the throughput of any product on the market. Untether AI provides an automated SDK for its products. The SDK takes a network model implemented in common machine learning frameworks like TensorFlow and PyTorch and lowers it into the kernel code that runs on these RISC-V processors. It automatically takes care of low-level optimizations, providing extensive visualization, a cycle-accurate simulator, and an easily adoptable runtime API.

Company	Use-Case	Technology	Solution	Programmability	Funding (Mio. \$) (PitchBook)	
Axelera d-Matrix	AI on the Edge AI inference in datacenters	SRAM (digital MAC) SRAM (digital MAC)	Hardware-SoC Chiplets	SDK provided Open-source Framework(s)	63.72 (Early stage VC) 161.3 (Early stage VC)	
Synthara	AI, DSP, Linear algebra	SRAM (dot product)	Accelerator	Compiler available	3.33 (Grant)	
Mythic	AI on the Edge	Flash (Analog computing)	Accelerator, processor	Software stack (does rewriting, opt. mapping)	177.41 (Later stage VC)	
Surecore SEMRON	AI on the edge	SRAM (CNM)	Chip	No details	11.16 (Later stage VC)	
Untether AI	AI on the edge	Memcapacitor	3D-Chip (planned)	No details	1.63 (Seed round)	
	AI everywhere	SRAM+RISC-V (CNM)	Chips, accelerator	SDK and simulator (Toolkit)	153.52 (Early stage VC)	
Syntiant	AI on the edge	SRAM+ARM (CNM)	MCUs	Processor	Available	121.43 (Later stage VC)
Neuroblade	Analytics	DRAM+RISC (CNM)	cores	Processor	Set of APIs	110.43 (Debt - General)
Rain AI	LLMs on the edge (Training)	Memristors		Processors	NA	64.04 (Later stage VC)
TetraMem	Edge applications	Memristors		Processors, software stack	HDK, SDK	NA
Gyrfalcon Tech	AI on the edge	CNM (MACs with MRAM)	Chip	NA	68.0 (Debt - PPP)	
UPMEM EnCharge AI	General-purpose AI inference	DRAM+RISC cores SRAM (CIM) + SIMD (CNM)	System Chip	APIs Software available	15.5 (Later stage VC) 21.7 (Angel - individ)	
Re(conceive)	AI inference	SRAM (analog CIM)	Chip	NA	NA	
Fractile	LLMs inference	NA	Chip	NA	NA	

Table 3. CIM/CNM companies, with their products, technologies, and funding status.

Funding: Untether's investors include CPPIB, GM Ventures, Intel Capital, Radical Ventures, and Tracker Capital.

5.18 UPMEM Technology

Founded in 2015, UPMEM is a tech company offering programmable CNM systems for data-intensive applications. See more details on the architecture and programmability in Section 4.1.1.

Funding: The company is funded by Western Digital, Partech, and super nova invest.

5.19 Summary

Table 3 and Figure 27 summarize the discussion in this section and provides a landscape of CIM, CNM companies, their products, technologies, and funding status. Please note that this compilation is not exhaustive; it includes only companies known to us and those that, based on our understanding, fall within the CIM and CNM categories. As Figure 27 clearly illustrates, the current landscape is predominantly characterized by conventional technologies, with a notable absence of a comprehensive software ecosystem.

5.20 Open challenges

CIM and CNM systems have already entered the market, yet a series of open challenges are expected to become more pronounced as time progresses. It will take years to understand how these units will harmonize within the overall system architecture and determine their optimal utilization. In the following, we briefly discuss the important categories.

Materials: During the era of Moore's law in computing, the primary focus was on refining transistors to be smaller, faster, and more energy-efficient. The selection of materials was confined to only

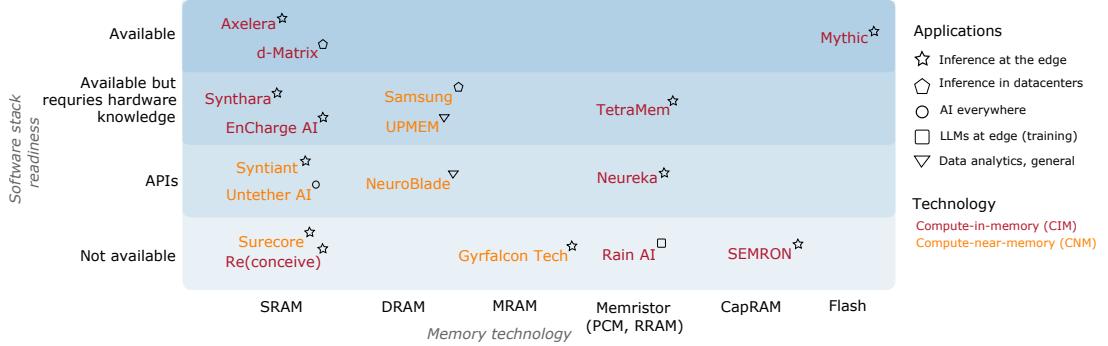


Fig. 27. A landscape of CIM and CNM companies, highlighting their technologies, target applications and software stack readiness.

those compatible with manufacturing processes. However, the limitations of these materials to scale further are now exposed. As a result, new materials have emerged and further research is needed to investigate novel materials (to enable further transistor scaling: hopes with carbon nanotube, and novel memory devices).

Devices: Mainstream computing has largely relied on digital logic and binary storage. Nonetheless, the emerging wave of computing architectures, particularly CIM requires novel multi-state devices allowing both analog and digital operations. Existing devices, memristors in particular, do offer such properties but have reliability and other associated challenges.

Integration: We have seen various architectures based on various technologies. As is evident, there is no on-technology-fits-all solution. Eventually, CIM modules based on different technologies need to be integrated into the same to get the best out of all these technologies. This poses integration challenges that have received little to no attention.

Processing systems: These novel architectures require new tools, algorithms, cost models, and software solutions. All of them are crucial to understanding these architectures, enabling their design space exploration, and making them accessible to a larger audience.

While every challenge holds significance and demands attention, programmability and user-friendliness are the most important ones from the user's standpoint. Following is an excerpt from Facebook's recent article on their inference accelerator that highlights the same.

"We've investigated applying processing-in-memory (PIM) to our workloads and determined there are several challenges to using these approaches. Perhaps the biggest challenge of PIM is its programmability".

In response to the challenges associated with programmability, we have ourselves been working on high-level programming and compilation frameworks for CNM and CIM systems [102–105]. We have developed reusable abstractions and demonstrated compilation flows for CIM systems with memristive crossbars, CAMs, CIM-logic modules, and for CNM systems like UPMEM and Samsung CNM. However, much more cross-layer work is needed to improve automation [106], in particular for heterogeneous systems integrating several paradigms and technologies.

6 CONCLUSIONS

This paper overviews the landscape of compute-near-memory (CNM) and compute-in-memory (CIM) paradigms. It starts with an explanation of the Von Neumann bottleneck, the necessity of novel CIM/CNM paradigms, and the key terminology used in the related literature. It offers a comprehensive

background on major memory technologies and emphasizes the importance of heterogeneous systems. The paper overviews prominent CIM and CNM designs from both academia and industry. In contrast to other studies in the literature that focus on either application domains or memory technologies, this paper concentrates on designs that have either successfully transitioned into product offerings or have reached a stage where commercialization is a feasible prospect. We explain prevalent CNM architectures, including microarchitectural details, associated technologies, software frameworks, and the results achieved (usually measured as throughput). Subsequently, we survey the landscape of CIM systems, explaining prevailing CIM designs that use prominent technologies such as SRAM, DRAM, MRAM, RRAM, PCM, and FeFET. We overview CIM chips from industrial giants (research centers), spanning from earlier designs like ISAAC and PUMA by Hewlett Packard Enterprise to the most cutting-edge chips from IBM, Samsung, TSMC, Intel, Meta (Facebook), Bosch, Fraunhofer, and GlobalFoundries. Current trends in industrial research show that while conventional SRAM and DRAM technologies are ready to be leveraged in CIM/CNM systems, emerging technologies like PCM, RRAM, MRAM, and FeFETs are also poised to make partial inroads, particularly for selected operations, such as dot products and pattern matching.

Finally, we describe the landscape of CIM and CNM start-ups, highlighting the emergence of numerous new companies in recent years that have introduced innovative solutions to cater to the thriving demands of AI and other data-intensive application domains. These companies are targeting a diverse range of market segments, spanning from power-efficient edge applications (AI at the edge) to high-performance data center servers (e.g., for AI training), and many have successfully secured substantial funding (hundreds of millions) in their initial funding rounds. The paper shows that SRAM technology currently dominates this landscape. However, with active research and breakthroughs in emerging NVMs (demonstrated by recent industrial chips), it is anticipated that NVMs will play a more prominent role in these paradigms in the near future.

The paper highlights that CIM and CNM technologies (i) harbor significant potential to outperform conventional systems, and (ii) have already made inroads into the market. However, their true potential remains untapped. This is attributed to a number of challenges, including the lack of accurate design space exploration tools, programming frameworks, and a comprehensive software ecosystem in general, and cost and performance models that can be leveraged to guide static and runtime optimizations for these systems.

CNM and CIM computing is an extremely active field. We believe that we have captured a representative snapshot of this field, early in year 2024, and remain excited about how technologies, devices, architectures and tools will continue to develop moving forward.

ACKNOWLEDGEMENTS

This work was supported by Vsquared Ventures (VSQ). Special thanks to Max Odendahl (Venture Partner at VSQ) for his feedback on previous versions of the manuscript. This work was also supported by the German Research Council (DFG) through the HetCIM project (project number 502388442) in the context of the DFG Priority Program on Disruptive Memory Technologies (SPP2377 <https://spp2377.uos.de>) and the German Federal Ministry of Education and Research (BMBF, project number 01IS18026A-D) by funding the competence center for Big Data and AI ScaDS.AI Dresden/Leipzig (<https://scads.ai>).

REFERENCES

- [1] S. Li, A. O. Glova, X. Hu, P. Gu, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, “Scope: A stochastic computing engine for dram-based in-situ accelerator,” in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018, pp. 696–709.

- [2] A. de Vries, “The growing energy footprint of artificial intelligence,” *Joule*, vol. 9, no. 4, p. 1–4, Oct 2023. [Online]. Available: <https://doi.org/10.1016/j.joule.2023.09.004>
- [3] J. Calma. (2023, September) Microsoft is going nuclear to power its ai ambitions. The Verge. [Online]. Available: <https://www.theverge.com/2023/9/26/23889956/microsoft-next-generation-nuclear-energy-smr-job-hiring>
- [4] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, “Ai and ml accelerator survey and trends,” in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2022, pp. 1–10.
- [5] F. Devaux, “The true processing in memory accelerator,” in *2019 IEEE Hot Chips 31 Symposium (HCS)*. IEEE Computer Society, 2019, pp. 1–24.
- [6] Y.-C. Kwon, S. H. Lee, J. Lee, S.-H. Kwon, J. M. Ryu, J.-P. Son, O. Seongil, H.-S. Yu, H. Lee, S. Y. Kim *et al.*, “25.4 a 20nm 6gb function-in-memory dram, based on hbm2 with a 1.2 tflops programmable computing unit using bank-level parallelism, for machine learning applications,” in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64. IEEE, 2021, pp. 350–352.
- [7] S. Lee, S.-h. Kang, J. Lee, H. Kim, E. Lee, S. Seo, H. Yoon, S. Lee, K. Lim, H. Shin *et al.*, “Hardware architecture and software stack for pim based on commercial dram technology: Industrial product,” in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 43–56.
- [8] M. He, C. Song, I. Kim, C. Jeong, S. Kim, I. Park, M. Thottethodi, and T. Vijaykumar, “Newton: A dram-maker’s accelerator-in-memory (aim) architecture for machine learning,” in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 372–385.
- [9] S. Lee, K. Kim, S. Oh, J. Park, G. Hong, D. Ka, K. Hwang, J. Park, K. Kang, J. Kim *et al.*, “A 1ynm 1.25 v 8gb, 16gb/s/pin gddr6-based accelerator-in-memory supporting 1tflops mac operation and various activation functions for deep-learning applications,” in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65. IEEE, 2022, pp. 1–3.
- [10] H.-S. Wong and S. Salahuddin, “Memory leads the way to better computing,” *Nature nanotechnology*, vol. 10, pp. 191–4, 03 2015.
- [11] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, “Metal–oxide rram,” *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.
- [12] W. J. Gallagher and S. S. P. Parkin, “Development of the magnetic tunnel junction mram at ibm: From first junctions to a 16-mb mram demonstrator chip,” *IBM J. Res. Dev.*, vol. 50, no. 1, pp. 5–23, Jan. 2006. [Online]. Available: <http://dx.doi.org/10.1147/rd.501.0005>
- [13] J. Hoffman, X. Pan, J. W. Reiner, F. J. Walker, J. P. Han, C. H. Ahn, and T. P. Ma, “Ferroelectric field effect transistors for memory applications,” *Advanced Materials*, vol. 22, no. 26–27, pp. 2957–2961, 2010. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.200904327>
- [14] P. M. Research, “Market study on in-memory computing: Adoption of fast-processing databases fuels the demand. report pmrrep33026,” 2022.
- [15] P. Radojković, P. Carpenter, P. Esmaili-Dokht, R. Cimadomo, H.-P. Charles, S. Abu, and P. Amato, “Processing in memory: the tipping point,” *White paper: Processing in Memory: the Tipping Point*, 2021.
- [16] M. Anderson, B. Chen, S. Chen, S. Deng, J. Fix, M. Gschwind, A. Kalaiah, C. Kim, J. Lee, J. Liang *et al.*, “First-generation inference accelerator deployment at facebook,” *arXiv preprint arXiv:2107.04140*, 2021.
- [17] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, “Memory devices and applications for in-memory computing,” *Nature Nanotechnology*, pp. 1–16, 2020.
- [18] F. Ottati, G. Turvani, G. Masera, and M. Vacca, “Custom memory design for logic-in-memory: Drawbacks and improvements over conventional memories,” *Electronics*, vol. 10, no. 18, p. 2291, 2021.
- [19] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, “Magic—memristor-aided logic,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014.
- [20] E. Lehtonen, J. Poikonen, and M. Laiho, *Memristive Stateful Logic*, 01 2014, pp. 603–623.
- [21] P. Chi *et al.*, “Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory,” in *ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 27–39.
- [22] A. Shafiee *et al.*, “Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.
- [23] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, “Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories,” in *Proceedings of the 53rd Annual Design Automation Conference*, 2016, pp. 1–6.
- [24] W. A. Simon, Y. M. Qureshi, M. Rios, A. Levisse, M. Zapater, and D. Atienza, “Blade: An in-cache computing architecture for edge devices,” *IEEE Transactions on Computers*, vol. 69, no. 9, pp. 1349–1363, 2020.

- [25] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 273–287.
- [26] D. Fakhry, M. Abdelsalam, M. W. El-Kharashi, and M. Safar, "A review on computational storage devices and near memory computing for high performance applications," *Memories-Materials, Devices, Circuits and Systems*, p. 100051, 2023.
- [27] G. Singh, L. Chelini, S. Corda, A. J. Awan, S. Stuijk, R. Jordans, H. Corporaal, and A.-J. Boonstra, "Near-memory computing: Past, present, and future," *Microprocessors and Microsystems*, vol. 71, p. 102868, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0141933119300389>
- [28] A. Gebregiorgis, H. A. Du Nguyen, J. Yu, R. Bishnoi, M. Taouil, F. Catthoor, and S. Hamdioui, "A survey on memory-centric computer architectures," *J. Emerg. Technol. Comput. Syst.*, vol. 18, no. 4, oct 2022. [Online]. Available: <https://doi.org/10.1145/3544974>
- [29] F. Gao, G. Tziantziulis, and D. Wentzlaff, "Computedram: In-memory compute using off-the-shelf drams," in *Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture*, 2019, pp. 100–113.
- [30] M. Ali, A. Jaiswal, S. Kodge, A. Agrawal, I. Chakraborty, and K. Roy, "Imac: In-memory multi-bit multiplication and accumulation in 6t sram array," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 8, pp. 2521–2531, 2020.
- [31] Z.-R. Wang, Y.-T. Su, Y. Li, Y.-X. Zhou, T.-J. Chu, K.-C. Chang, T.-C. Chang, T.-M. Tsai, S. M. Sze, and X.-S. Miao, "Functionally complete boolean logic in 1t1r resistive random access memory," *IEEE Electron Device Letters*, vol. 38, no. 2, pp. 179–182, 2016.
- [32] A. Kazemi, F. Müller, M. M. Sharifi, H. Errahmouni, G. Gerlach, T. Kämpfe, M. Imani, X. S. Hu, and M. Niemier, *Scientific reports*, vol. 12, no. 1, p. 19201, 2022.
- [33] R. Neale, D. Nelson, and G. E. Moore, "Nonvolatile and reprogrammable, the read-mostly memory is here," *Electronics*, vol. 43, no. 20, pp. 56–60, 1970.
- [34] G. W. Burr, M. J. Brightsky, A. Sebastian, H.-Y. Cheng, J.-Y. Wu, S. Kim, N. E. Sosa, N. Papandreou, H.-L. Lung, H. Pozidis *et al.*, "Recent progress in phase-change memory technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 146–162, 2016.
- [35] Z. Guo, J. Yin, Y. Bai, D. Zhu, K. Shi, G. Wang, K. Cao, and W. Zhao, "Spintronics for energy-efficient computing: An overview and outlook," *Proceedings of the IEEE*, vol. 109, no. 8, pp. 1398–1417, 2021.
- [36] A. Kent and D. Worledge, "A new spin on magnetic memories," *Nature nanotechnology*, vol. 10, pp. 187–91, 03 2015.
- [37] D. Reis, M. Niemier, and X. S. Hu, "Computing in memory with fefets," in *Proceedings of the international symposium on low power electronics and design*, 2018, pp. 1–6.
- [38] J. D. Kendall and S. Kumar, "The building blocks of a brain-inspired computer," *Applied Physics Reviews*, vol. 7, no. 1, 2020.
- [39] V. Milo, G. Malavena, C. Compagnoni, and D. Ielmini, "Memristive and cmos devices for neuromorphic computing," *Materials*, vol. 13, p. 166, 01 2020.
- [40] D. Patterson, K. Asanovic, A. Brown, R. Fromm, J. Golbus, B. Gribstad, K. Keeton, C. Kozyrakis, D. Martin, S. Perissakis, R. Thomas, N. Treuhaft, and K. Yelick, "Intelligent ram (iram): the industrial setting, applications, and architectures," in *Proceedings International Conference on Computer Design VLSI in Computers and Processors*, 1997, pp. 2–7.
- [41] J. Draper, J. Chame, M. Hall, C. Steele, T. Barrett, J. LaCoss, J. Granacki, J. Shin, C. Chen, C. W. Kang *et al.*, "The architecture of the diva processing-in-memory chip," in *Proceedings of the 16th international conference on Supercomputing*, 2002, pp. 14–25.
- [42] Y. Kang, W. Huang, S.-M. Yoo, D. Keen, Z. Ge, V. Lam, P. Pattnaik, and J. Torrellas, "Flexram: Toward an advanced intelligent memory system," in *2012 IEEE 30th International Conference on Computer Design (ICCD)*. IEEE, 2012, pp. 5–14.
- [43] J. Gómez-Luna, I. E. Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a new paradigm: An experimental analysis of a real processing-in-memory architecture," *arXiv preprint arXiv:2105.03814*, 2021.
- [44] H. Shin, D. Kim, E. Park, S. Park, Y. Park, and S. Yoo, "Mcdrum: Low latency and energy-efficient matrix computations in dram," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2613–2622, 2018.
- [45] B. Kim, J. Chung, E. Lee, W. Jung, S. Lee, J. Choi, J. Park, M. Wi, S. Lee, and J. H. Ahn, "Mvid: Sparse matrix-vector multiplication in mobile dram for accelerating recurrent neural networks," *IEEE Transactions on Computers*, vol. 69, no. 7, pp. 955–967, 2020.
- [46] L. Ke, X. Zhang, J. So, J.-G. Lee, S.-H. Kang, S. Lee, S. Han, Y. Cho, J. H. Kim, Y. Kwon *et al.*, "Near-memory processing in action: Accelerating personalized recommendation with axdimm," *IEEE Micro*, vol. 42, no. 1, pp. 116–127, 2021.

- [47] A. Ankit et al., "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 715–731.
- [48] A. Yazdanbakhsh, C. Song, J. Sacks, P. Lotfi-Kamran, H. Esmaeilzadeh, and N. S. Kim, "In-dram near-data approximate acceleration for gpus," in *Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques*, 2018, pp. 1–14.
- [49] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, 2015, pp. 105–117.
- [50] D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski, "Top-pim: Throughput-oriented programmable processing in memory," in *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*, 2014, pp. 85–98.
- [51] R. Nair, S. F. Antao, C. Bertolli, P. Bose, J. R. Brunheroto, T. Chen, C.-Y. Cher, C. H. Costa, J. Doi, C. Evangelinos et al., "Active memory cube: A processing-in-memory architecture for exascale systems," *IBM Journal of Research and Development*, vol. 59, no. 2/3, pp. 17–1, 2015.
- [52] M. Gao and C. Kozyrakis, "Hrl: Efficient and flexible reconfigurable logic for near-data processing," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. Ieee, 2016, pp. 126–137.
- [53] D. Reis, A. F. Laguna, M. Niemier, and X. S. Hu, "In-memory computing accelerators for emerging learning paradigms," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, 2023, pp. 606–611.
- [54] L. Xie, H. A. Du Nguyen, J. Yu, A. Kaichouhi, M. Taouil, M. AlFailakawi, and S. Hamdioui, "Scouting logic: A novel memristor-based logic design for resistive computing," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2017, pp. 176–181.
- [55] A. Ankit, I. El Hajj, S. R. Chalamalasetti, S. Agarwal, M. Marinella, M. Foltin, J. P. Strachan, D. Milojevic, W.-M. Hwu, and K. Roy, "Panther: A programmable architecture for neural network training harnessing energy-efficient reram," *IEEE Transactions on Computers*, vol. 69, no. 8, pp. 1128–1142, 2020.
- [56] L. Song et al., "Pipeler: A pipelined reram-based accelerator for deep learning," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2017, pp. 541–552.
- [57] X. Qiao, X. Cao, H. Yang, L. Song, and H. Li, "Atomlayer: A universal reram-based cnn accelerator with atomic layer computation," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.
- [58] P. Chen, M. Wu, Y. Ma, L. Ye, and R. Huang, "Rimac: An array-level adc/dac-free reram-based in-memory dnn processor with analog cache and computation," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, 2023, pp. 228–233.
- [59] G. Yuan, P. Behnam, Z. Li, A. Shafiee, S. Lin, X. Ma, H. Liu, X. Qian, M. N. Bojnordi, Y. Wang et al., "Forms: Fine-grained polarized reram-based in-situ computation for mixed-signal dnn accelerator," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 265–278.
- [60] Z. Wang, S. Joshi, S. Savel'Ev, W. Song, R. Midya, Y. Li, M. Rao, P. Yan, S. Asapu, Y. Zhuo et al., "Fully memristive neural networks for pattern classification with unsupervised learning," *Nature Electronics*, vol. 1, no. 2, pp. 137–145, 2018.
- [61] W.-H. Chen, K.-X. Li, W.-Y. Lin, K.-H. Hsu, P.-Y. Li, C.-H. Yang, C.-X. Xue, E.-Y. Yang, Y.-K. Chen, Y.-S. Chang et al., "A 65nm 1mb nonvolatile computing-in-memory reram macro with sub-16ns multiply-and-accumulate for binary dnn ai edge processors," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 494–496.
- [62] C.-X. Xue, W.-H. Chen, J.-S. Liu, J.-F. Li, W.-Y. Lin, W.-E. Lin, J.-H. Wang, W.-C. Wei, T.-W. Chang, T.-C. Chang et al., "24.1 a 1mb multibit reram computing-in-memory macro with 14.6 ns parallel mac computing time for cmn based ai edge processors," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2019, pp. 388–390.
- [63] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C.-X. Xue, W.-H. Chen et al., "33.2 a fully integrated analog reram based 78.4 tops/w compute-in-memory chip with fully parallel mac computing," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 500–502.
- [64] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaauuw, and R. Das, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *2018 ACM/IEEE 45Th annual international symposium on computer architecture (ISCA)*. IEEE, 2018, pp. 383–396.
- [65] M. Kang, S. K. Gonugondla, S. Lim, and N. R. Shanbhag, "A 19.4-nj/decision, 364-k decisions/s, in-memory random forest multi-class inference accelerator," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 7, pp. 2126–2135, 2018.
- [66] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-mb in-memory-computing cnn accelerator employing charge-domain compute," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019.
- [67] A. Biswas and A. P. Chandrakasan, "Conv-sram: An energy-efficient sram with in-memory dot-product computation for low-power convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, 2018.
- [68] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "Xnor-sram: In-memory computing sram macro for binary/ternary deep neural networks," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, 2020.

- [69] S. Angizi, Z. He, A. S. Rakin, and D. Fan, “Cmp-pim: an energy-efficient comparator-based processing-in-memory neural network accelerator,” in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.
- [70] J. Doevenspeck, K. Garello, B. Verhoeft, R. Degraeve, S. Van Beek, D. Crotti, F. Yasin, S. Couet, G. Jayakumar, I. Papistas *et al.*, “Sot-mram based analog in-memory computing for dnn inference,” in *2020 IEEE Symposium on VLSI Technology*. IEEE, 2020, pp. 1–2.
- [71] L. Chang, X. Ma, Z. Wang, Y. Zhang, Y. Xie, and W. Zhao, “Pxnor-bnn: In/with spin-orbit torque mram preset-xnor operation-based binary neural networks,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2668–2679, 2019.
- [72] S. Jung, H. Lee, S. Myung, H. Kim, S. K. Yoon, S.-W. Kwon, Y. Ju, M. Kim, W. Yi, S. Han *et al.*, “A crossbar array of magnetoresistive memory devices for in-memory computing,” *Nature*, vol. 601, no. 7892, pp. 211–216, 2022.
- [73] E. Breyer, H. Mulaosmanovic, T. Mikolajick, and S. Slesazeck, “Reconfigurable nand/nor logic gates in 28 nm hkmg and 22 nm fd-soi fet technology,” in *2017 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2017, pp. 28–5.
- [74] X. Yin, C. Li, Q. Huang, L. Zhang, M. Niemier, X. S. Hu, C. Zhuo, and K. Ni, “Fecam: A universal compact digital and analog content addressable memory using ferroelectric,” *IEEE Transactions on Electron Devices*, vol. 67, no. 7, pp. 2785–2792, 2020.
- [75] A. Kazemi, M. M. Sharifi, A. F. Laguna, F. Müller, R. Rajaei, R. Olivo, T. Kämpfle, M. Niemier, and X. S. Hu, “In-memory nearest neighbor search with fetef multi-bit content-addressable memories,” in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1084–1089.
- [76] R. Khaddam-Aljameh, M. Stanisavljevic, J. F. Mas, G. Karunaratne, M. Braendli, F. Liu, A. Singh, S. M. Müller, U. Egger, A. Petropoulos *et al.*, “Hermes core—a 14nm cmos and pcm-based in-memory compute core using an array of 300ps/lsb linearized cco-based adcs and local digital processing,” in *2021 Symposium on VLSI Circuits*. IEEE, 2021, pp. 1–2.
- [77] M. Le Gallo, R. Khaddam-Aljameh, M. Stanisavljevic, A. Vasilopoulos, B. Kersting, M. Dazzi, G. Karunaratne, M. Brändli, A. Singh, S. M. Mueller *et al.*, “A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference,” *Nature Electronics*, pp. 1–14, 2023.
- [78] Q. Dong, M. E. Sinangil, B. Erbagci, D. Sun, W.-S. Khwa, H.-J. Liao, Y. Wang, and J. Chang, “15.3 a 351tops/w and 372.4 gops compute-in-memory sram macro in 7nm finfet cmos for machine-learning applications,” in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 242–244.
- [79] H. Wang, R. Liu, R. Dorrance, D. Dasalukunte, D. Lake, and B. Carlton, “A charge domain sram compute-in-memory macro with c-2c ladder-based 8-bit mac unit in 22-nm finfet process for edge inference,” *IEEE Journal of Solid-State Circuits*, vol. 58, no. 4, pp. 1037–1050, 2023.
- [80] S. De, F. Mueller, N. Laleni, M. Lederer, Y. Raffel, S. Mojumder, A. Vardar, S. Abdulazhanov, T. Ali, S. Dünkel *et al.*, “Demonstration of multiply-accumulate operation with 28 nm fetef crossbar array,” *IEEE Electron Device Letters*, vol. 43, no. 12, pp. 2081–2084, 2022.
- [81] G. Pedretti, C. E. Graves, S. Serebryakov, R. Mao, X. Sheng, M. Foltin, C. Li, and J. P. Strachan, “Tree-based machine learning performed in-memory with memristive analog cam,” *Nature communications*, vol. 12, no. 1, p. 5806, 2021.
- [82] K. Akarvardar and H.-S. P. Wong, “Technology prospects for data-intensive computing,” *Proceedings of the IEEE*, vol. 111, no. 1, pp. 92–112, 2023.
- [83] C. Zhang, H. Sun, S. Li, Y. Wang, H. Chen, and H. Liu, “A survey of memory-centric energy efficient computer architecture,” *IEEE Transactions on Parallel and Distributed Systems*, 2023.
- [84] “Axeler,” <https://www.axeler.ai/digital-in-memory-computing-for-deep-learning-acceleration/>.
- [85] “d-matrix,” <https://www.d-matrix.ai/>.
- [86] “Gyrfalcon tech,” <https://www.gyrfalcontech.ai/about-us/company-overview/>.
- [87] “Memcpu,” <https://www.memcpu.com/>.
- [88] “Memverge,” <https://memverge.com/company/>.
- [89] “mythic,” <https://mythic.ai/>.
- [90] “Neuroblade,” <https://www.neuroblade.com/product/>.
- [91] “Rain,” <https://rain.ai/about-us/>.
- [92] “Semron,” <https://www.semron.ai>.
- [93] “Surecore,” <https://www.sure-core.com>.
- [94] “Synthara,” <https://www.synthara.ai>.
- [95] “Syntiant,” <https://www.syntiant.com/>.
- [96] “Tetramem,” <https://www.tetramem.com>.
- [97] M. Rao, H. Tang, J. Wu, W. Song, M. Zhang, W. Yin, Y. Zhuo, F. Kiani, B. Chen, X. Jiang *et al.*, “Thousands of conductance levels in memristors integrated on cmos,” *Nature*, vol. 615, no. 7954, pp. 823–829, 2023.
- [98] “Encharge ai,” <https://enchargeai.com>.

- [99] “Reconceive,” <https://www.re-conceive.com/home>.
- [100] “Fractile,” <https://www.fractile.ai/>.
- [101] “Untether,” <https://www.untether.ai/>.
- [102] A. Siemieniuk, L. Chelini, A. A. Khan, J. Castrillon, A. Drebes, H. Corporaal, T. Grosser, and M. Kong, “OCC: An automated end-to-end machine learning optimizing compiler for computing-in-memory,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 41, no. 6, pp. 1674–1686, Aug. 2021. [Online]. Available: <https://ieeexplore.ieee.org/document/9502921>
- [103] A. A. Khan, H. Farzaneh, K. F. Friebel, L. Chelini, and J. Castrillon, “Cinm (cinnamon): A compilation infrastructure for heterogeneous compute in-memory and compute near-memory paradigms,” *arXiv preprint arXiv:2301.07486*, 2022.
- [104] H. Farzaneh, J. P. C. de Lima, M. Li, A. A. Khan, X. S. Hu, and J. Castrillon, “C4cam: A compiler for cam-based in-memory accelerators,” *arXiv preprint arXiv:2309.06418*, 2023.
- [105] J. P. C. de Lima, A. A. Khan, H. Farzaneh, and J. Castrillon, “Full-stack optimization for cam-only dnn inference,” in *Proceedings of the 2024 Design, Automation and Test in Europe Conference (DATE)*, ser. DATE’24. IEEE, Mar. 2024, pp. 1–6.
- [106] J. Ryckaert, M. Niemier, Z. Enciso, M. M. Sharifi, X. S. Hu, I. O’Connor, A. Graening, R. Sharma, P. Gupta, J. Castrillon, J. P. C. de Lima, A. A. Khan, and H. Farzaneh, “Smoothing disruption across the stack: Tales of memory, heterogeneity, and compilers,” in *Proceedings of the 2024 Design, Automation and Test in Europe Conference (DATE)*, ser. DATE’24. IEEE, Mar. 2024, pp. 1–6.

ACRONYMS

CIM <i>compute-in-memory</i>	1
IMC <i>in-memory-computing</i>	4
IMP <i>in-memory-processing</i>	4
LIM <i>logic-in-memory</i>	4
PIM <i>processing-in-memory</i>	4
PUM <i>processing-using-memory</i>	4
CNM <i>compute-near-memory</i>	4
NMC <i>near-memory-computing</i>	4
PNM <i>processing-near-memory</i>	4
NMP <i>near-memory-processing</i>	4
ADC <i>analog-to-digital converter</i>	5
APU <i>accelerated processing unit</i>	16
BL <i>bitline</i>	7
BLB <i>bitline bar</i>	7
CIM-A CIM-array	5
CIM-P CIM-peripheral	5
CAM <i>content-addressable-memory</i>	17
CNN <i>convolutional neural network</i>	17
COM <i>compute-outside-memory</i>	4
DAC <i>digital-to-analog converter</i>	5
DNN <i>deep neural network</i>	18
DPU <i>data processing unit</i>	11
DRAM <i>dynamic random-access memory</i>	5

CNM/CIM Landscape	41
FeFET ferroelectric field-effect transistor	2
FPU floating-point unit	13
HBM high bandwidth memory	7
ISA instruction set architecture	14
LUT look-up table	15
MAC multiply-accumulate	12
MLC multi-level cell	20
MOS metal-oxide-semiconductor	9
MRAM magnetic RAM	2
MTJ magnetic tunnel junction	8
MPSoC multiprocessor system-on-chip	4
NN neural-network	20
NVM non-volatile memory	2
OMP outside memory processing	26
PCM phase change memory	2
PU processing unit	11
RRAM resistive RAM	2
SDK software development kit	12
SIMD single-instruction multiple-data	13
SLC single-level cell	20
SoC system-on-chip	26
SRAM static random-access memory	5
SOT Spin-orbit-torque	8
STT spin-transfer-torque	8

TRA <i>triple row activation</i>	21
TSVs <i>through-silicon vias</i>	7
WL <i>word line</i>	7
CAGR <i>compound annual growth rate</i>	2
MVM <i>matrix-vector multiplication</i>	5
SA <i>sense amplifier</i>	12
LLM <i>large language models</i>	1