Processing-In-Memory (PIM) Technologies and Its Impacts on Computer Architecture: A Survey

Edreese Basharyar
Electrical & Computer Engineering Department
University of Nevada, Las Vegas
Las Vegas, United States
basharya@unlv.nevada.edu

Dylan Cazares
Electrical & Computer Engineering Department
University of Nevada, Las Vegas
Las Vegas, United States
cazard1@unlv.nevada.edu

Abstract—**Currently, Processing-In-Memory (PIM) technologies are at the forefront of modern efforts to overcome the limitations imposed by the Von Neumann bottleneck, integrating processing directly with memory units such as DRAM, SRAM, and other non-volatile memory types. In this survey, we delve into various cutting-edge PIM techniques developed over the past several years, outline its key technological advancements, evaluate the advantages and limitations within each of the techniques, and present a comparative analysis to discern their effectiveness into features such as energy efficiency, latency, and throughput. The analysis clearly identifies the conditions under which specific PIM techniques excel in, highlighting their potential to revolutionize future computing systems. This survey not only charts the current state of progress in PIM technologies but also serves as an indispensable resource for researchers and developers by providing a detailed and comprehensive snapshot of the field that can inform future technological developments and inspire new applications in high-performance and data-intensive computing environments.**

Keywords—Processing-In-Memory, Von Neumann bottleneck, PIM architectures, non-volatile memory, emerging memory technologies

I. Introduction

The idea of the Von Neumann architecture has been the predominant model for computing systems across all various sectors in today's society [1, 5]. The architecture separates the memory unit, which supplies data to the central processing unit, from the input and output devices that facilitate user interaction and display results. Its foundational functionality has been so influential that even modern architectures do not entirely replace the Von Neumann architecture, but they adapt and evolve its core principles to develop modernized versions that meet today's computing needs. However, despite its widespread use and efficient functionality, the Von Neumann Architecture faces significant challenges in terms of data movement. Transferring the data to and from memory consumes considerable time and energy, ultimately proving costly as computational speeds continue to escalate [2]. For instance, when large volumes of data need to be moved from the memory unit to the CPU, the long resulting times and inefficient energy expenditures fail to support the optimal performance and energy efficiency that it is intended to have. This issue is commonly referred to as both the 'memory wall' problem and the 'Von Neumann bottleneck' in recent years.

The Von Neumman bottleneck can be divided into three primary concerns, each of which significantly impacts the overall efficiency and performance of computing systems. First, moving data between memory arrays and the processing unit incurs significant delays, leading to noticeable latency and reduced efficiency. Second, the limitations of the bandwidth within the memory hierarchy can severely limit the flow of data. Third, data transfer between the computing unit and off-chip memory can consume a significant amount of energy, as much as around two orders of magnitude more than a standard floating point operation on the processing unit [4].

There are also several other factors that contribute to this issue as well, which make the Von Neumann bottleneck quite a persistent challenge in terms of pursuing energy efficiency and system performance [2]. DRAM, one of the most widely used volatile memory systems in modern computing devices, is typically mounted on a DIMM. However, the bandwidth and data transfer speeds are very limited due to the finite number of pins that exist, resulting in a bottleneck. Another significant factor is the disparity in operational frequency between the DRAM memory module and the CPU; the CPU operates at a much higher clock speed than DRAM, leading to delays and mismatches that often leave the CPU waiting for the memory module to retrieve data. The final significant issue is the attempt to create a one-size-fits-all memory architecture to address the Von Neumann bottleneck across all computing systems. However, with the wide variety of computing devices each with their own unique set of requirements, it is impossible to meet all of the needs to ensure optimal performance; some applications, for example, might require high throughput to handle large datasets, while others may need low-latency access in order to retrieve small, scattered data points.

To overcome all of these challenges presented, processing-in-memory (PIM) technology is considered as a modern, promising solution to finally mitigate the Von-Neumman bottleneck. The main premise of PIM is recognizing that separating the memory and computing units, as seen in the typical Von Neumann architecture, should instead be reimagined so that they are adapted to work closely together, with the potential to reduce data movement and latency by processing the data directly within or near the memory. We will examine the various hierarchical levels that PIM technologies can operate, such as cell-level operations, progressing to more complex operations, and then examining larger and memory-bound operations.

However, another level of complexity is added when these approaches are used by a wide variety of modern memory technologies that are used in computing systems [3] (e.g., DRAM, embedded DRAM [eDRAM], ferroelectric DRAM [FeRAM], hybrid memory cube [HMC], high-bandwidth memory [HBM] , resistive RAM [ReRAM], and phase change memory [PCM]. In this paper, we will analyze these concepts in greater detail, review how the idea of PIM has evolved in recent years, compare their respective advantages and drawbacks, identify the current leading technologies at the forefront of research, and highlight essential insights for future researchers aiming to advance new innovations in PIM.

II. Memory Technologies

A. DRAM

DRAM, also known as dynamic random access memory, is a type of volatile memory widely used as the main system memory in modern computing systems and other electronic devices [5-7]. Due to decades of development and research, it has become the standard memory technology, supporting rapid data access, high-performance computing, and is affordable for both consumer and enterprise applications. DRAM memory systems consist of three main components, the memory controller, the memory bus, and several DRAM chips arranged in Dual-Inline Memory Modules (DIMMs) [2]. Inside each of these main components consist of several individual features. The memory controller has several roles: (1) overseeing the timing of read, write, and refresh operations; (2) managing data transfers between the CPU and the memory modules via the data bus, and (3) controlling the power supply to the memory modules to ensure efficient power consumption [5]. The memory bus serves as the main communication channel between the memory controller and the DRAM devices. For each DRAM device, there typically consists of a memory cell, row and column decoders, sense amplifiers, and row buffers. A DRAM device uses a memory cell to store data, which can be built with only one transistor and one capacitor; it has the capability to represent the logical values ‘1’ ,when the capacitor is fully charged, and a ‘0’ when the capacitor is discharged. An array structure is formed with the use of word-lines and bit-lines: the word-lines activate a specific row of memory cells, which is selected by a row decoder to grant simultaneous access to all cells in that row and is loaded into a row buffer, and specific data is selected based on a column address which is then transmitted to the CPU through the memory bus [5].

Understanding the operations involved in accessing data in DRAM is essential to comprehending the foundational principles of emerging modern technologies. The process begins when the memory controller receives a request from the CPU for data at a specific memory address. This address is translated into a specific row and column location within the DRAM module. Then, the memory controller activates the appropriate row by sending a ‘row address strobe’ (RAS) signal, which sends the data to a row buffer.

For a DRAM read operation, the memory controller issues a read command to access the required column address within the active row. After the selected data is placed on the data bus and transmitted to the CPU, the original data in the row is cleared as part of the self-destructive read process. Therefore, the data must be rewritten back to the row before closing it. In order for this process to be accomplished, the memory controller initiates a precharge operation. During this process, the sense amplifiers are reset to their initial state, the bit lines are charged to a typical reference voltage for accurate data detection in the subsequent read operation. Additionally, the data from the active row is rewritten into the memory cells before the row is closed, ensuring that the DRAM is ready for the next operation.

As semiconductor technology advances over the past years, manufacturers are pushing DRAM chips to smaller feature sizes in order to fit more memory cells within a given chip area, which is labeled as 'extreme scaling' [5], [8]. To achieve this desired scaling, DRAM capacitors must be reduced to their smallest possible size, which consequently decreases the amount of charge they can hold. This shrinkage makes memory cells more prone to data retention issues, interference between neighboring cells, and a circuit-level phenomenon that is known as RowHammer [9]. RowHammer can cause visible bit flips in DRAM regions physically near rapidly accessed rows. This vulnerability poses a significant challenge for modern system designers because it exploits the fundamental behavior of a DRAM which cannot be easily altered. As technology scaling becomes more aggressive and circuit components are packed more densely, the susceptibility to RowHammer increases, which makes the scalability of DRAM cells a major problem over recent years [10].

B. HMC

The Hybrid Memory Cube (HMC) is a promising next-generation dynamic random access memory technology designed to overcome the limitations of traditional DRAM. HMC is based on a true 3-dimensional (3D) stacked architecture, where multiple DRAM dies are placed atop a logic die. Figure 1 provides a visual schematic of this HMC architecture. This stacking forms a structure known as a vault, with the logic die containing a vault controller responsible for managing all memory operations, including simple atomic commands such as arithmetic, boolean, comparison, and bitwise operations [11]. HMC executes memory-intensive operations within or near the memory itself, categorizing it as a Processing-in-Memory (PIM) technology distinct from traditional DRAM architecture.

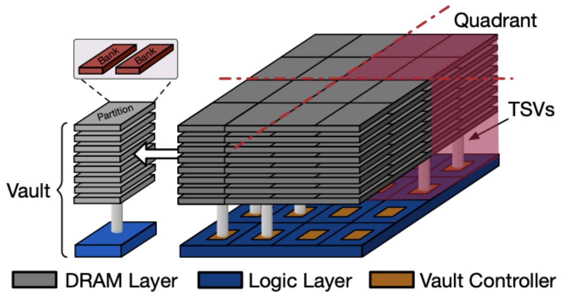


Figure 1 - Schematic of HMC Architecture

However, this innovative architecture introduces new challenges, primarily due to the structure of the 3D stacking. This tightly packed structure results in high temperatures, directly causing performance degradation within the overall system [11-12]. The thermal issues continue to worsen if the circuit implementations become more complex, as the heat dissipation from the densely packed layers become insufficient causing significant thermal problems. Also, the increase in power density due to the high amount of data throughput can exacerbate heat generation, which calls for more necessary effective cooling solutions.

C. HBM

In response to the increasing bandwidth demands of modern applications, High-Bandwidth Memory (HBM) presents an attractive solution for high performance system designers due to its scalability, compact footprint, and low power consumption [13]. The fundamental structure of HBM consists of a base logic die and vertically stacked DRAM dies, all interconnected using through-silicon vias (TSVs).

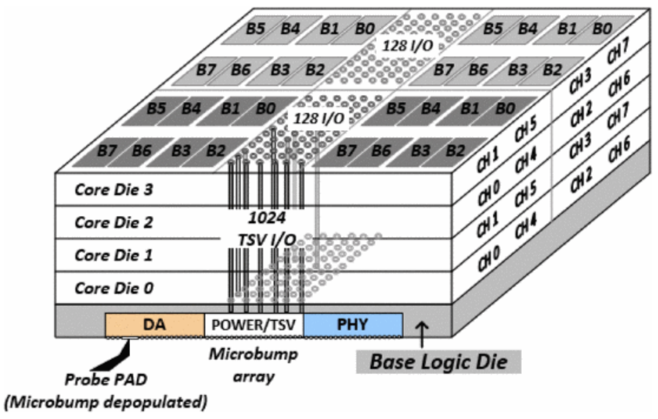


Figure 2 - Schematic of HBM Architecture

Figure 2 depicts the 3D stacked architecture of an HBM module, where multiple core DRAM dies (Core Die 0 to Core Die 3) are stacked atop a baes logic die. Each DRAM die is divided into banks (B0 to B7) and organized into channels (CH0 to CH7). The through-silicon vias (TSVs) connect the DRAM dies vertically to the bae logic die, providing 1,024 input/output (I/O) channels for high-bandwidth data transfers [13]. Although HBM builds upon traditional DRAM devices, its architectural adaptation allows for significantly higher bandwidth and memory capacity by implementing multiple stacks. The TSVs enable faster communication through the layers while also reducing latency compared to the planar structure of traditional DRAM architectures [13]. The performance of HBM is enhanced by its innovative interface that enables a wide data bus, which is a huge improvement in bandwidth compared to traditional memory technologies. In terms of reducing data movement, HBM aligns well with the goals of PIM technologies for an alternative solution to mitigate the Von Neumann bottleneck.

D. ReRAM

Resistive random access memory (ReRAM), is a non-volatile memory technology that can be adopted as an alternative to alleviate the Von Neumann bottleneck associated with traditional DRAM architectures. Unlike charge-based memory, ReRAM stores its information by altering the resistance of the material. Its structure can be divided into two primary components: an insulating layer sandwiched between two metal electrodes. Conductive filaments that form and break within the insulating layer enable the memory cell to switch between a high-resistance state (HRS) and a low-resistance state (LRS) [14-15].

There are three main operations used in ReRAM cells: set, reset, and read. The set operation transitions the ReRAM cell from HRS to LRS, while the reset operation reverses this process, bringing the cell back to HRS. The read operation identifies the current resistance state by applying a sensing voltage without affecting the cell's overall resistance. Applying this external voltage pulse that causes the transition between HRS to LRS (logic '0') and LRS (logic '1') is a phenomenon known as resistive switching (RS) [14-15].

ReRAM's resistance-based data allows for direct computation within the memory array, aligning with the goals of PIM technologies by reducing data movement and enabling faster computational speeds. Figure 3 illustrates the MIM structure, displaying the conductive filaments formed within the insulating layer between the electrodes, displaying the resistive switching behavior mechanism that ReRAM embodies [15-16]. The schematic and the cross-sectional view of the ReRAM cell is seen in Fig. 3a and b, respectively.

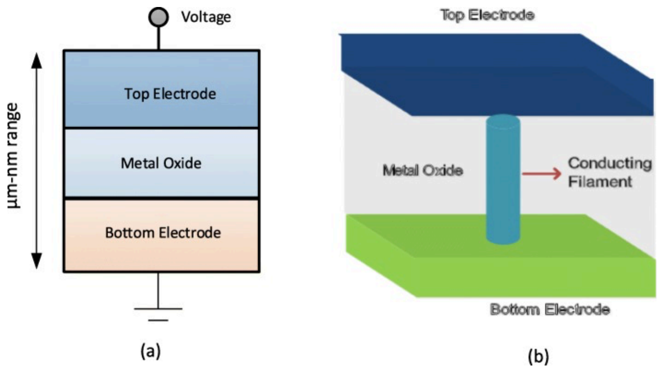


Figure 3 - (a) Schematic of metal-insulator-metal (MIM) structure for ReRAM (b) Cross-sectional view of ReRAM

E. PCM

Phase Change Memory (PCM), is an emerging non-volatile memory positioned as an alternative to traditional DRAM architecture [17]. Compared to DRAM, PCM offers several advantages including non-volatility, improved scalability, lower leakage power, and equal read latency in comparison. Similar to ReRAM, the design of a Single-Level Cell (SLC) PCM differentiates between two resistance levels of a phase change material in order to store logic bits (logic '0' or logic '1') [17-18]. PCM also utilizes set, read, and reset operations for its primary operations , much like ReRAM. The reset operation employs a high-amplitude, short-duration current pulse to switch the material to a high-resistance amorphous state, while the set operation uses a low-amplitude, longer-duration current pulse to switch the material to a low-resistance polycrystalline state. The amorphous high-resistance state is used to represent a binary 0, while the crystalline low-resistance state represents a binary 1. The PCM cell can be read by sensing the current flow through it, which can take only tens of nanoseconds [17-18]. However, PCM's set operation takes significantly longer than both the read and reset latency, resulting in higher average write latency and causing up to 60% performance degradation [19].

This inherent issue poses a significant challenge to the adoption of PCM for the next generation of memory systems, particularly in PIM technologies. It is vital for researchers to explore solutions to reduce PCM's long write latency in order to make it a viable option. Current approaches that have been presented in literature are categorized into four main strategies: (1) optimizing DRAM-PCM architectures to reduce the amount of write operations directed to PCM [20-23], (2) scheduling PCM writes during idle bank cycles to maximize efficiency [19], [24-26], (3) using data coding schemes that are latency aware that lessen the demand for writing logic '1' operations [27-28], and (4) developing solutions specifically aimed at reducing write latency in Multi-Level Cell (MLC) PCMs [29-33]. However, these are still limiting approaches in terms of improving the average write latency for PCM.

F. eDRAM

Over recent years, embedded DRAM (eDRAM) has gained prominence as a cost-effective alternative for enhancing DRAM memory operations [34]. eDRAM integrates DRAM directly inside the chip, thereby improving the efficiency and functionality of electronic devices by reducing power consumption and increasing processing speeds. The primary objective of eDRAM is to overcome the scalability and performance constraints of conventional DRAM architecture. It offers a high-bandwidth, low-latency memory solution, improved data transmission, and can be applied to devices such as CPUS, high-end servers, and networking equipment [34-35]. eDRAM also reduces the overall size and implementation of the overall system by minimizing the surface space required on the motherboard. A prominent application of eDRAM is accelerating data processing for machine learning tasks, including deep neural networks (DNNs) [34-35], which require massive amounts of data movement between off-chip memory to on-chip processing cores. Furthermore, it is valuable in IoT sensor nodes, which both require the use of leveraging PIM technologies in order to alleviate bottlenecks and reduce power consumption. However, the implementation of eDRAM poses many significant challenges, since data is stored as electrical energy. This way of storing data introduces the potential to degrade over time due to environmental factors such as temperature and noise, necessitating periodic refresh operations in order to preserve data integrity.

Compared to other embedded memory technologies, eDRAM provides the densest bitcell size, highest on-die capacity, high endurance, high performance, high bandwidth, low energy per bit access, and low soft error rate [36-39]. Despite its challenges, its compelling features make it a viable option for PIM technologies, as it mitigates data movement bottlenecks while maintaining data integrity.

G. FeRAM

Ferroelectric random access memory (FeRAM), is a novel and innovative memory technology known for its simple structure, ease of implementation, high integration and low power consumption, and has been positioned as one of the founding memory technologies in the scope of PIM. Recent research has observed the challenges associated with ReRAM, including high write power, long read latency, and the power overhead required to drive large arrays [40]. In response, researchers have developed a ferroelectric field-effect transistor (FeFET), an alternative memory solution for PIM architecture [40-41]. The FeFET incorporates a ferroelectric oxide layer into the gate dielectric stack. A ferroelectric oxide is an insulator that exhibits spontaneous electric polarization in the absence of an electric field. The direction can be reversed by applying a larger voltage than the coercive voltage on the gate terminal of the FeFET [41-42]. When the polarization is pointing downwards, the transistor remains in the 'on' state, and when it points upwards, the channel goes through accumulation and enters the 'off' state. Studies indicate that hafnium oxide-based FeFETs demonstrate excellent temperature stability, write endurance, data retention, and switching speed and energy [42-44]. The ultralow writing energy resulting from its unique electrical field effect switching mechanism is the most prominent feature that distinguishes the FeFET from other emerging memory technologies. Unlike ReRAM, FeFET is gate-driven, which eliminates the large RC delay in ReRAM and reduces the overall read latency [41].

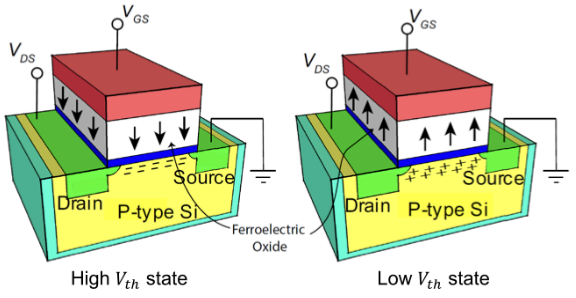
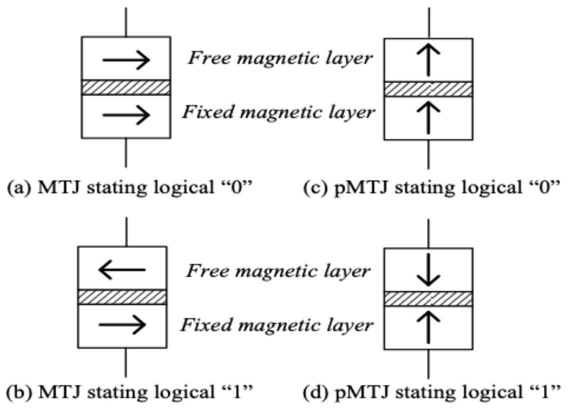


Figure 4 - Diagram of FeFET in ON/OFF state

Figure 4 illustrates the operating principle of the FeFET and highlights how changes in the polarization within the ferroelectric oxide layer influence the state of the transistors. The 'on' state is depicted on the left side of the figure, while the 'off' state is shown on the right side of the figure.

H. STT-MRAM

Spin-transfer torque random access memory (STT-MRAM) is an emerging non-volatile memory technology based on magnetic RAM (MRAM). It is characterized by its non-volatility, rapid read and write speeds (less than 10 nanoseconds), high programming endurance (greater than 1,015 cycles), and zero standby power consumption [45]. The formation of the storage capability is based on a magnetic tunneling junction (MTJ), where a thin tunneling dielectric layer made with materials such as magnesium oxide (MgO) is sandwiched between two ferromagnetic layers. One of the ferromagnetic layers, known as the pinned layer, has its magnetization fixed, while the other layer, known as the free layer, is able to switch its magnetization during a write event. When both layers have the same polarity, the MTJ has a low resistance, which represents logical 0. Conversely, if the magnetizations have opposite polarities, the MTJ displays high resistance, which represents logical 1 [45-46].



Free magnetic layer
Fixed magnetic layer
(a) MTJ stating logical "0"
(c) pMTJ stating logical "0"
Free magnetic layer
Fixed magnetic layer
(b) MTJ stating logical "1"
(d) pMTJ stating logical "1"

Figure 5 - STT-MRAM cell

Figure 5 illustrates the construction and functionality of the STT-MRAM, showcasing the high-resistance and low-resistance states of the MTJ. A new variation of the MTJ, known as perpendicular MTJ (pMTJ) [47]. In this variation, the poles of the pMTJ magnetic layers are perpendicularly aligned with the plane of the wafer, as seen in Figures 5c and 5d. This arrangement results in a significant reduction in write current compared to conventional MTJ designs. A benefit of using pMTJ is that it requires much lower write current than the conventional MTJ [48].

Research shows that replacing traditional on-chip cache with STT-MRAM can improve overall system performance and reduce power consumption due to higher off-chip cache capacity and increased density, which together help lower cache miss rates. Furthermore, the zero standby leakage of STT-MRAM also contributes to a reduction in power consumption [48-49].

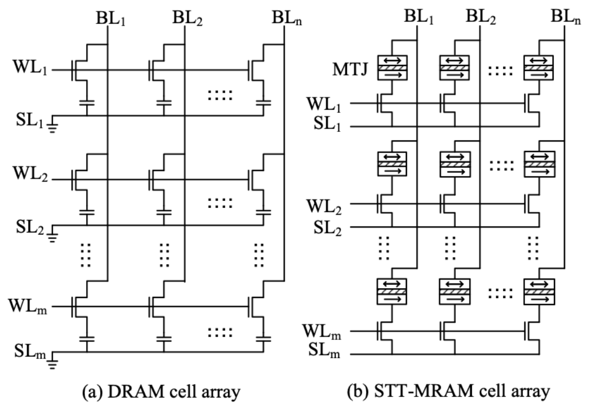


Figure 6 - STT-MRAM and DRAM cell-array

In terms of the cell design, we can see the conventional DRAM cell array in Figure 6a and the STT-MRAM cell array in Figure 6b, where we can see that both use transistors to access a selected set of cells. The fundamental difference comes from the cell type; while the DRAM uses a capacitor, STT-MRAM cells use the MTJ. Despite the fundamental difference in storage mechanism, both architectures share a similar design in terms of rows, columns, buffers, and banks. This structural compatibility allows STT-MRAM to integrate seamlessly with existing DRAM-based architectures. The high-density, energy-efficient, and high-speed characteristics of STT-MRAM, combined with its compatibility in existing architectures, make it a promising memory technology for PIM systems [47].

III. Analysis of PIM Literature

Since the late 1960s, memory architecture has evolved significantly, driven by the high demand for data processing due to technological advancements in concepts such as machine learning (ML) and artificial intelligence (AI). Since the primary bottleneck dilemma lies in the motherboard connections for most computing systems, many researchers have been looking at ways to alleviate this bottleneck by integrating processing-in-memory (PIM) into existing memory architectures, with an objective to reduce the bottleneck constraints and enhance computational performance.

In this study, we will classify PIM implementations into three categories: design architecture, memory technologies, and dedicated PIM technologies at the module level.

A. Design Architecture

Several studies [50-51, 54, 56] have found ways to implement PIM by adjusting the internal circuitry of memory arrays to perform computations on the module itself, with the computed output subsequently transferred to the CPU.

Seshandri et al. [50] proposed Ambit, which is an in-memory accelerator to perform bulk bitwise operations by exploiting the analog operations of DRAM by using 1T1C circuitry to perform AND, OR, and NOT functions. Further studies [51] have extended its capabilities by implementing multiplication operations within the DRAM sub-array.

Ambit uses the charge sharing phase to perform bulk bitwise operations. Since the sense amplifier is shared by many DRAM cells on the same bitline, it is able to perform 512 or 1024 of these operations simultaneously. This simultaneous operation process is known as triple-row activation (TRA), which requires at least two cells to be utilized for the input while the third cell is used to store the output.

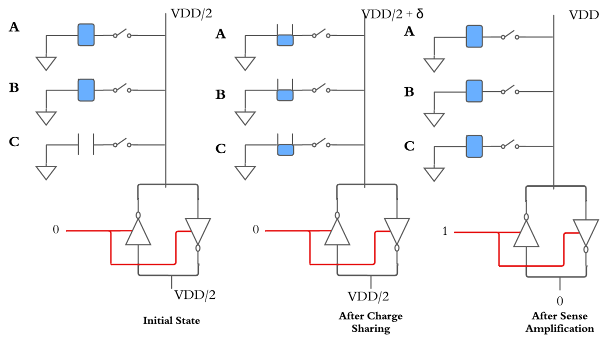


Figure 7 - Triple Row Activation

Figure 6 illustrates the process of TRA in a 1T1C DRAM array. Initially, two of the three cells are fully charged with all transistors modeled as open switches and the bitline supplied with a voltage of VDD/2. Once the transistors close, charge sharing occurs across all three cells causing a voltage deviation (δ) on the bitline. Once the sense amplifier is enabled in the row buffer, the bitline is driven to either VDD or 0, representing a logical '1' or '0'. After the sense amplifier is activated, all three cells are now fully charged. The logical states of cells A, B, and C can then be represented as follows:

$$AB + BC + CA$$

Or it can be rewritten as the expression:

$$C(A + B) + \sim C(AB)$$

With row-level activation in DRAM, TRA operates across an entire row of DRAM cells and sense amplifiers, enabling multi-kilobyte-wide bitwise AND/OR operations between two rows. Once integrated into a DRAM module, it minimally impacts the module's overall layout size, increasing it by less than 1%. Given that a single stick of RAM can contain over eight chips, this would cause a prominent effect on PIM.

As previously mentioned, there are many different approaches to memory design, one of which is Resistive RAM (ReRAM). Leitersdorf et al. [52] utilized ReRAM by proposing a way to speed up in-memory multiplication by using a novel partition-based computing technique for dta broadcasting and shifting. They also designed an in-memory multiplication algorithm based on the carry-save add-shift (CSAS) technique, which was made possible by exploiting the unique properties of memristors. Building upon the memristor design of Kvatinsky et al. [53], Leitersdorf produced this method using a crossbar array architecture. Similar to the TRA technique in DRAM, the resistance of memristors can be controlled via an applied voltage, enabling stateful logic to be performed within the crossbar array.

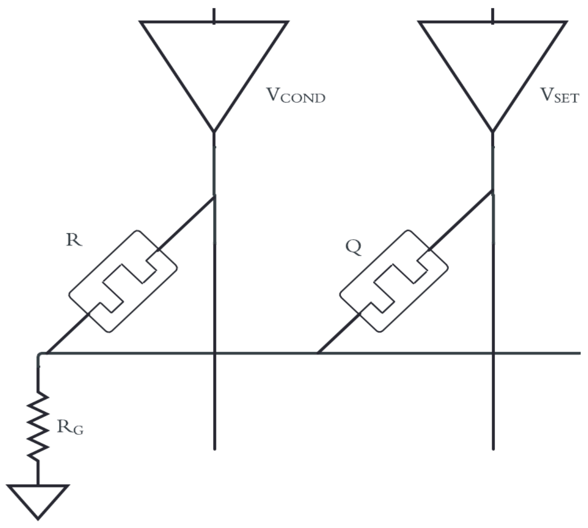


Figure 8 - Memristor Crossbar

Long et al. [54] proposed a ReRAM design to enhance the acceleration of recurrent neural network (RNN) computation. The authors proposed a PIM processing engine into three main subarrays: ReRAM crossbar subarrays for matrix-vector multiplication (MVM), special function unit (SFU) subarrays for nonlinear function, and multiplier subarrays for element-wise operations, as shown in Figure 9.

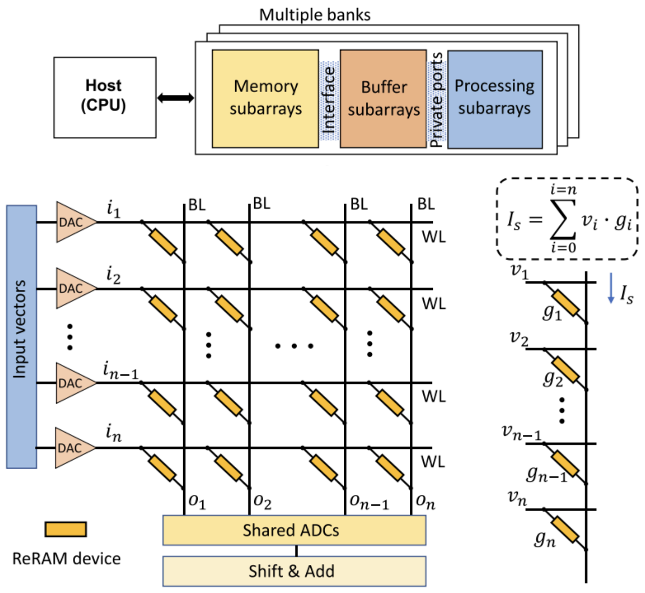


Figure 9: PIM architecture and ReRAM crossbar for matrix-vector multiplication proposed by Long et al.

Kazemi et al. [55] proposed a different type of PIM architecture, utilizing Hyperdimensional Computing (HDC) to perform multi-bit operations by using ferroelectric field-effect transistor (FeFET) crossbar arrays for multiply-and-add (MAC) operations and FeFET multi-bit content addressable memories for associative searches.

Lee et. al [4] proposed an HBM2-based experimental platform to include 16-cycle MAC units and 8-cycle reducers for matrix-vector multiplication. Their study shows a performance increase of 406% and 35.2% with all-bank and per-bank scheduling, respectively. Other studies that revolved around accelerating neural networks have also shown innovative techniques of implementing PIM techniques [57-59].

B. Memory Technologies

Beyond modifying transistor-level circuitry in memory arrays, few studies have shown that processing-in-memory (PIM) can be achieved in or around either the row buffer or a sense amplifier.

In their study, Long et al. [40] proposed a FeFET based PIM architecture to accelerate the inference of deep neural networks (DNN). The design proposed by Long et al. is a FeFET crossbar for digital in-memory vector-matrix multiplication (VMM) to enable bit-parallel computation and eliminate analog-to-digital conversion (ADC), which was used in prior PIM designs. Simulations were done in a 28nm CMOS process, which showed higher computing efficiency over commercial desktop graphic processing unit (GPU) and ReRAM-based designs. There design is built on three core concepts: using FeFET as the memory cell due to its lower read latency and ultra-low programming energy compared to ReRAM, exploiting the FeFETs operation to design an all digital VMM engine which eliminates the need for an ADC/DAC while ensuring high throughput, and achieving microarchitecture scalability by connecting multiple VMM engines using a hierarchical network-on-chip (H-NoC) with in-router accumulators. The study reported 115× higher computing efficiency compared to the Nvidia GTX 1080Ti desktop GPU and 6.3× higher computing efficiency over ReRAM-based PIM architectures.

Roy et al. proposed a novel multiplication scheme inside the DRAM subarray level, with minimal changes in the DRAM architecture subarrays. Compared to Ambit, Roy incorporates a quintuple-row activation to expand operational capabilities. To perform ADD operations, nine additional ‘compute rows’ are required for copying the original data and storing the carry-out and carry-in values. The operation comprises four main steps: (1) copy the first vector bit (A) to the compute rows, (2) copy the second vector bit (B) to the computer rows, (3) calculate Cout using the multiple-row activation scheme as shown in the equation below, and (4) calculate the sum using the multiple-row activation as seen in the equation below.

$$Cout = Majority(A, B, Cin)$$

$$Sum = Majority(A, B, Cin, \sim Cout, \sim Cout)$$

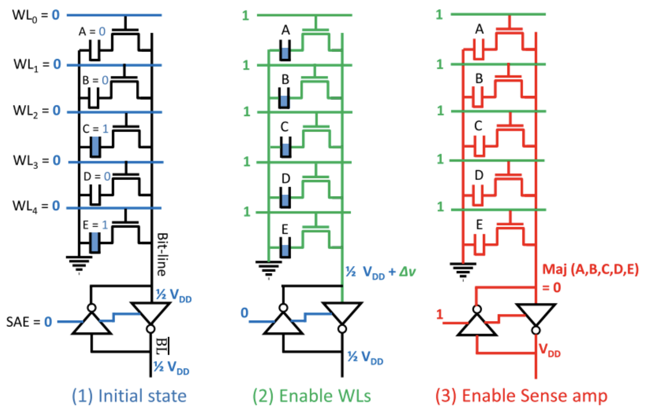
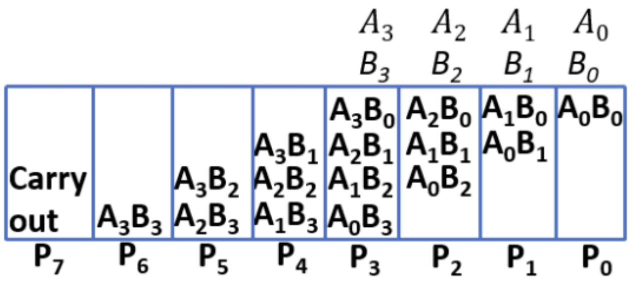


Figure 10: Quintuple-row activation proposed by Roy



				A_3	A_2	A_1	A_0
				B_3	B_2	B_1	B_0
				A_3B_0	A_2B_0	A_1B_0	A_0B_0
			A_3B_1	A_2B_1	A_1B_1	A_0B_1	
		A_3B_2	A_2B_2	A_1B_2	A_0B_2		
Carry out	A_3B_3	A_2B_3	A_1B_3	A_0B_3			
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

Figure 11 - Multiplication of two 4-bit operands broken down into AND and ADD operations

Similar to [50], the capacitances store charge to represent a logical ‘1’ or ‘0’. Once the access MOSFETs are activated by the worldline, charge is shared among all the capacitors, and there is some voltage deviation on the bitline. Once the sense amplifiers are enabled, the bitline voltage is cleared, and the output of the sense amplifiers is a logical ‘1’. The multiplication operation can be broken down into AND and ADD operations. Figure 11 illustrates a 4-bit multiplication, where A_n and B_n refer to the nth bit of the two operands and P_n refers to the nth bit of the product.

The study also incorporates a novel PIM-DRAM bank architecture shown in Figure 11, including adder trees and nonlinear activation function units in each DRAM bank for efficient ML acceleration.

C. Dedicated PIM Technologies

Instead of making modifications to the memory array architecture, researchers have proposed adding a dedicated process-in-memory processing unit onto the memory module.

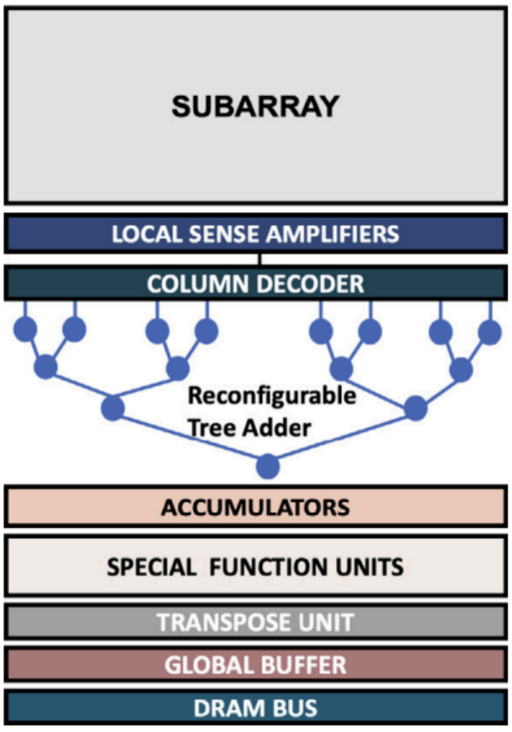


Figure 12 - PIM-DRAM bank architecture proposed by Roy et al.

Farahani et al. [61] proposed a near-DRAM acceleration architecture which processes data using accelerators 3D-stacked on DRAM devices composed of off-chip memory modules. This implementation of near-memory processing (NMP) shows improvement, with the NDA-based system consuming 46% (68% less energy) for data transfer and delivering 1.67× higher performance than a comparable system with the same logic accelerator integrated into the processor.

Kwon et al. [62] proposed a custom, tensor based, DIMM module enhanced with near memory processing cores tailored for Deep Learning (DL) applications. The DIMM module is located inside a GPU-centric system interconnect as a remote memory pool, which allows for scalable memory bandwidth and capacity expansion. The prototype proposed shows an average 6.2-17.6× performance improvement on DNN-based recommender systems.

Ke et al. [63] proposed a DRAM compliant, near-memory processing (NMP) solution to accelerate personalized recommendation inference. This solution looks into the Deep Learning Recommendation Model (DLRM) to provide accurate recommendations for content to users based on previous interactions. The proposed NMP accelerates the execution of a broad class of recommendation models, achieving a 9.8× memory latency speedup and 45.9% memory savings. The design proposed consists of a buffer chip on the DIMM; this buffer chip bridges the memory channel interface from the host and the standard DRAM device interface. Each buffer contains a NMP processing unit made up of a DIMM-NMP module and multiple rank-NMP modules. This approach has no effect on the DRAM chip itself, but it provides a scalable solution to DLRM applications to improve system throughput.

Zhou et al. [64] proposes a high performance inter-DIMM communication (IDC) in DIMM-NMP architectures, which supports seamless integration with existing host memory systems. DIMM-Link allows multiple DIMMs in a system with high-speed external data links, where both point-to-point communication and broadcast are effectively supported in a packet routing way. As seen in Figure 13, DIMM-Link consists of two main components: a DIMM-LINK Bridge (DL-Bridge) which connects multiple DIMM modules via customized interfaces, and multiple DIMM-Link Controllers (DL-Controller) which drives the DL-Bridge and handle the packet generation, routing and receiving.

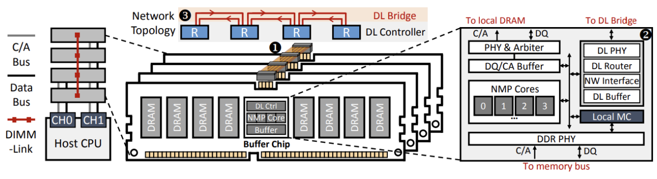


Figure 13 - DIMM-Link Architecture proposed by Zhou et al. [64]

Samsung [65] introduced the first silicon implementation of a PIM-HBM in a 20nm DRAM process with an unmodified processor. This innovation demonstrated PIM reduced the end-to-end execution time of memory bound neural network kernels and applications by up to 11.2× and 3.5× respectively, while also improving the energy efficiency up to 3.2× of the system running the applications.

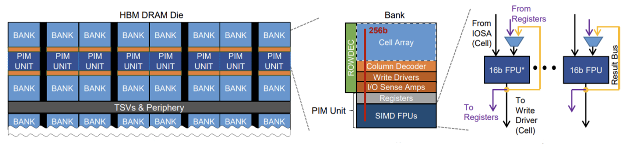


Figure 14 - Overview of PIM architecture proposed by Samsung [65]

Upon further research into the topic of commercial PIM memory technologies, Samsung [66] has introduced the world-first RNN-Transducer inference accelerator using Field Programmable Gate Array (FPGA) with the previously mentioned PIM-HBM technology. From the study, Samsung evaluated that their accelerator with PIM-HBM reduces execution time of RNN-T by 2.5× on average with 11.09% reduced LUT size and improved energy efficiency up to 2.6× compared to normal HBM cubes.

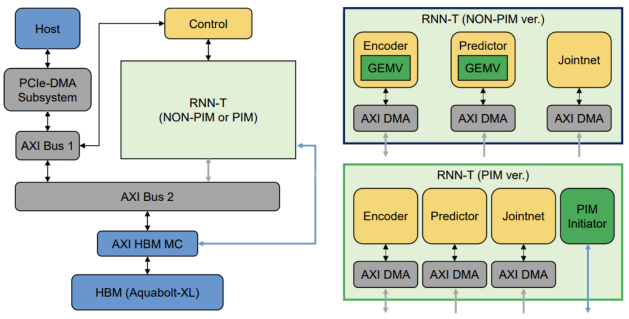
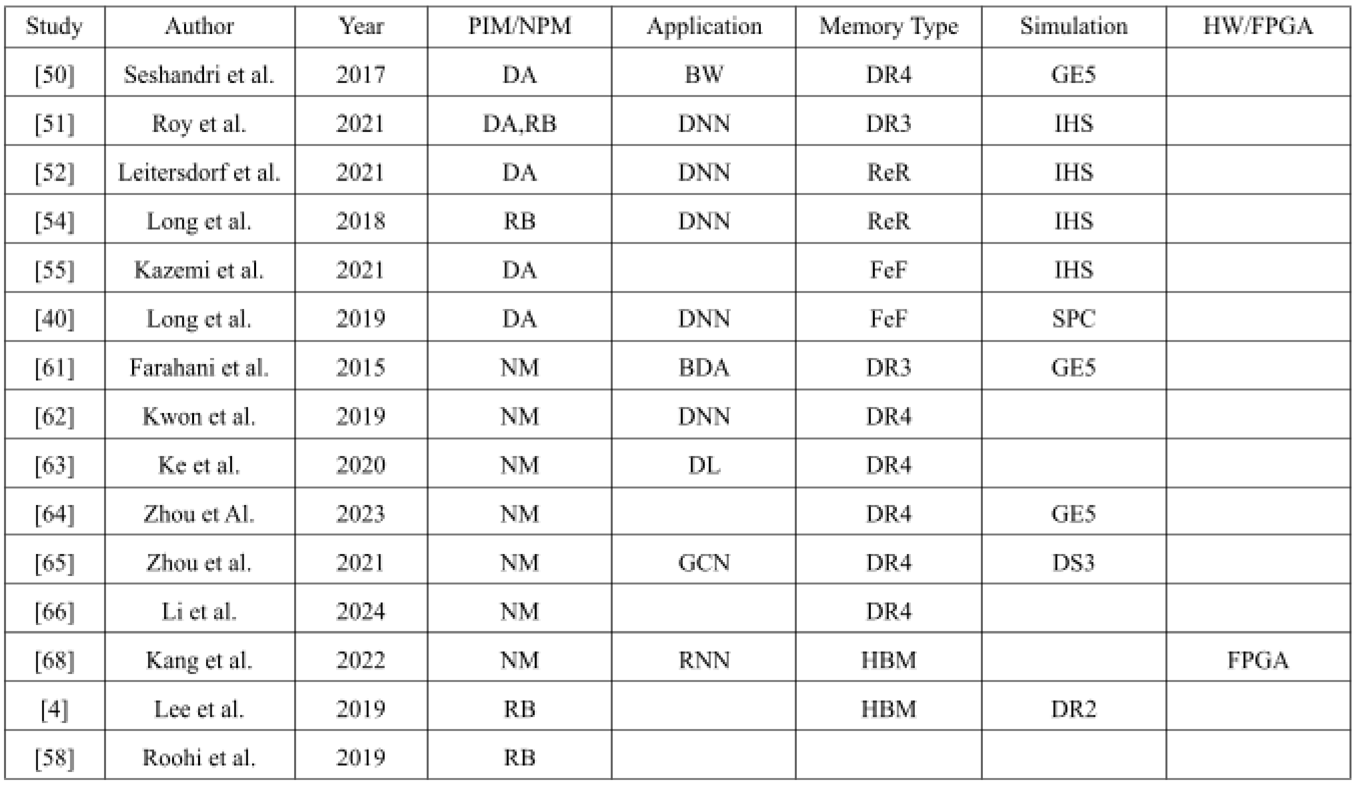


Figure 15 - FPGA PIM-HBM System proposed by Samsung [67]

IV. Comparative Analysis

With developments in machine learning, neural networks, and artificial intelligence, PIM has become a hot topic in research and discussion. This survey explores various methods for implementing PIM technologies, ranging from modifications at the several different data hierarchies. Table 1 summarizes the different PIM implementations studied by researchers, providing information on the publication year, whether it was developed within the data array or near-memory, the applications that it utilized, the memory type modified for compatibility, and the simulation software used for testing.

Table 1: A summary of the survey of different studies viewing the year the article was published, the type of PIM implemented or Near-Processing Memory Implemented, the type of memory that was used or modified, the simulation that the authors used and any additional hardware modifications to the memory module.



Study	Author	Year	PIM/NPM	Application	Memory Type	Simulation	HW/FPGA
[50]	Seshandri et al.	2017	DA	BW	DR4	GE5	
[51]	Roy et al.	2021	DA,RB	DNN	DR3	IHS	
[52]	Leitersdorf et al.	2021	DA	DNN	ReR	IHS	
[54]	Long et al.	2018	RB	DNN	ReR	IHS	
[55]	Kazemi et al.	2021	DA		FeF	IHS	
[40]	Long et al.	2019	DA	DNN	FeF	SPC	
[61]	Farahani et al.	2015	NM	BDA	DR3	GE5	
[62]	Kwon et al.	2019	NM	DNN	DR4		
[63]	Ke et al.	2020	NM	DL	DR4		
[64]	Zhou et Al.	2023	NM		DR4	GE5	
[65]	Zhou et al.	2021	NM	GCN	DR4	DS3	
[66]	Li et al.	2024	NM		DR4		
[68]	Kang et al.	2022	NM	RNN	HBM		FPGA
[4]	Lee et al.	2019	RB		HBM	DR2	
[58]	Roohi et al.	2019	RB				

. V. Challenges and Limitations

Despite significant developments and research in processing-in-memory (PIM) technology, there are still several challenges that remain before it can become a universal alternative to conventional architectures seen in modern computing systems today. Current, only a handful of industrial implementations, such as those from Samsung, are being manufactured in the mainstream market. Challenges such as limited processing versatility, energy efficiency constraints, and compatibility issues with the current frameworks of computer architectures are what is hindering the adoption of PIM technologies. While PIM devices are able to perform computational operations, most of the processing still occurs at the CPU level. As seen in Table 1, only a few existing PIM implementations have been successful, and it is crucial that further research must be done for improvement.

VI. Conclusions

In this survey, we reviewed various methods for designing memory across different architectures. With processing power at an all-time high due to cutting-edge advancements in machine learning and neural networks, the main bottleneck in the conventional Von Neumann architecture is becoming increasingly problematic. In this survey, we introduce and explore the latest research conducted on PIM technologies. We summarize the different methods and techniques that come with implementing PIM, ranging from modifying conventional memory to employing alternative memory technologies such as PCM, ReRAM, STT-MRAM, HMC, HBM, and FeRAM. While challenges still persist today with PIM technologies, such as the limited processing power of PIM compared to traditional CPUS and the complexity of the tasks involved, we believe that this survey is a valuable reference for future studies exploring PIM techniques. Addressing the challenges of the current state of PIM could help unlock the future potential that it can have to mitigate the limitations of the bottleneck dilemma once and for all.

1. DA = Data Array, RB = Row Buffer, NM = Near Memory, BW = BitWeaving, DNN = Deep Neural Networks, BDA = Big Data Applications, GCN = GCNear, DL = Deep Learning, DRx = DDRx, ReR = ReRAM, FeF = FeFET, GE5 = Gem5, IHS = In-House Software, SPC = SPICE, DS3 = DramSim3

References

[1] J. Gómez-Luna, I. N. 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 (Cornell University)*, May 2021, doi: https://doi.org/10.48550/arxiv.2105.03814.

[2] X. Zou, S. Xu, X. Chen, L. Yan, and Y. Han, "Breaking the von Neumann bottleneck: architecture-level processing-in-memory technology," *Science China Information Sciences*, vol. 64, no. 6, Apr. 2021, doi: https://doi.org/10.1007/s11432-020-3227-1.

[3] S. Raman, S. Teja, and J. P. Kulkarni, "Enabling In-Memory Computations in Non-Volatile SRAM Designs," *IEEE journal on emerging and selected topics in circuits and systems*, vol. 12, no. 2, pp. 557–568, Jun. 2022, doi: https://doi.org/10.1109/jetcas.2022.3174148.

[4] S. Lee *et al.*, "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology : Industrial Product," *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2021, doi: https://doi.org/10.1109/isca52012.2021.00013.

[5] K. Asifuzzaman, N. R. Miniskar, A. R. Young, F. Liu, and J. S. Vetter, "A survey on processing-in-memory techniques: Advances and challenges," *Memories - Materials, Devices, Circuits and Systems*, vol. 4, p. 100022, Jul. 2023, doi: https://doi.org/10.1016/j.memori.2022.100022.

[6] V. Sridharan and D. A. Liberty, "A study of DRAM failures in the field," Nov. 2012, doi: https://doi.org/10.1109/sc.2012.13.

[7] S. K. Kim and M. Popovici, "Future of dynamic random-access memory as main memory," *MRS Bulletin*, vol. 43, no. 5, pp. 334–339, May 2018, doi: https://doi.org/10.1557/mrs.2018.95.

[8] B. R. Childers, J. Yang, and Y. Zhang, "Achieving Yield, Density and Performance Effective DRAM at Extreme Technology Sizes," Oct. 2015, doi: https://doi.org/10.1145/2818950.2818963.

[9] J. Kim *et al.*, "Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques." Accessed: May 07, 2024. [Online]. Available: https://arxiv.org/pdf/2005.13121

[10] D. Kim *et al.*, "Rowhammer Attacks in Dynamic Random-Access Memory and Defense Methods," *Sensors*, vol. 24, no. 2, p. 592, Jan. 2024, doi: https://doi.org/10.3390/s24020592.

[11] D.-I. Jeon, K.-B. Park, and K.-S. Chung, "HMC-MAC: Processing-in Memory Architecture for Multiply-Accumulate Operations with Hybrid Memory Cube," *IEEE Computer Architecture Letters*, vol. 17, no. 1, pp. 5–8, Jan. 2018, doi: https://doi.org/10.1109/lca.2017.2700298.

[12] R. Hadidi, B. Asgari, A. Mudassar, S. Mukhopadhyay, S. Yalamanchili, and H. Kim, "Demystifying the Characteristics of 3D-Stacked Memories: A Case Study for Hybrid Memory Cube." Accessed: May 07, 2024. [Online]. Available: https://arxiv.org/pdf/1706.02725

[13] H. Jun *et al.*, "HBM (High Bandwidth Memory) DRAM Technology and Architecture," *IEEE Xplore*, May 01, 2017. https://ieeexplore.ieee.org/document/7939084 (accessed Jul. 20, 2022).

[14] H.-S. . P. Wong *et al.*, "Metal–Oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012, doi: https://doi.org/10.1109/jproc.2012.2190369.

[15] F. Zahoor, T. Z. Azni Zulkifli, and F. A. Khanday, "Resistive Random Access Memory (RRAM): an Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (mlc) Storage, Modeling, and Applications," *Nanoscale Research Letters*,

vol. 15, no. 1, Apr. 2020, doi: https://doi.org/10.1186/s11671-020-03299-9.

[16] Furqan Zahoor *et al.*, “Resistive random access memory: introduction to device mechanism, materials and application to neuromorphic computing,” *Discover Nano*, vol. 18, no. 1, Mar. 2023, doi: https://doi.org/10.1186/s11671-023-03775-y.

[17] B. R. Lee, Engin Ipek, O. Mutlu, and D. Burger, “Architecting phase change memory as a scalable dram alternative,” Jun. 2009, doi: https://doi.org/10.1145/1555754.1555758.

[18] I. Thakkar and S. Pasricha, “DyPhase: A Dynamic Phase Change Memory Architecture With Symmetric Write Latency and Restorable Endurance,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1760–1773, Sep. 2018, doi: https://doi.org/10.1109/tcad.2017.2762921.

[19] M. K. Qureshi, M. M. Franceschini, and L. A. Lastras-Montano, “Improving read performance of Phase Change Memories via Write Cancellation and Write Pausing,” *High-Performance Computer Architecture*, Apr. 2010, doi: https://doi.org/10.1109/hpca.2010.5416645.

[20] Hyung Gyu Lee, S. Baek, Chrysostomos Nicopoulos, and J. Kim, “An energy- and performance-aware DRAM cache architecture for hybrid DRAM/PCM main memory systems,” Oct. 2011, doi: https://doi.org/10.1109/iccd.2011.6081427.

[21] H. Aghaei Khouzani, F. S. Hosseini, and C. Yang, “Segment and Conflict Aware Page Allocation and Migration in DRAM-PCM Hybrid Main Memory,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, pp. 1458–1470, Sep. 2017, doi: https://doi.org/10.1109/tcad.2016.2615845.

[22] S. Lee, H. Bahn, and S. H. Noh, “CLOCK-DWF: A Write-History-Aware Page Replacement Algorithm for Hybrid PCM and DRAM Memory Architectures,” *IEEE Transactions on Computers*, vol. 63, no. 9, pp. 2187–2200, Sep. 2014, doi: https://doi.org/10.1109/tc.2013.98.

[23] D. Zhang, L. Ju, M. Zhao, X. Gao, and Z. Jia, “Write-back aware shared last-level cache management for hybrid main memory,” Jun. 2016, doi: https://doi.org/10.1145/2897937.2898110.

[24] Qureshi, Franceschini, Jagmohan, and Lastras, “PreSET: Improving performance of phase change memories by exploiting asymmetry in write times,” Jun. 2012, doi: https://doi.org/10.1109/isca.2012.6237033.

[25] Y. Kim, S. Yoo, and S. Lee, “Write performance improvement by hiding R drift latency in phase-change RAM,” Jun. 2012, doi: https://doi.org/10.1145/2228360.2228520.

[26] S. Kwon, S. Yoo, S. Lee, and J. Park, “Optimizing Video Application Design for Phase-Change RAM-Based Main Memory,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 11, pp. 2011–2019, Nov. 2012, doi: https://doi.org/10.1109/tvlsi.2011.2165974.

[27] J. Yue and Y. Zhu, “Accelerating write by exploiting PCM asymmetries,” *CiteSeer X (The Pennsylvania State University)*, Feb. 2013, doi: https://doi.org/10.1109/hpca.2013.6522326.

[28] J. Li and Kartik Mohanram, “Write-once-memory-code phase change memory,” *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, Jan. 2014, doi: https://doi.org/10.7873/date2014.194.

[29] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, “Improving write operations in MLC phase change memory,” Feb. 2012, doi: https://doi.org/10.1109/hpca.2012.6169027.

[30] J. Yue and Y. Zhu, “Making Write Less Blocking for Read Accesses in Phase Change Memory,” *CiteSeer X (The Pennsylvania State University)*, Aug. 2012, doi: https://doi.org/10.1109/mascots.2012.39.

[31] C. Pan, M. Xie, J. Hu, Y. Chen, and C. Yang, “3M-PCM,” Oct. 2014, doi: https://doi.org/10.1145/2656075.2656076.

[32] L. Jiang, Y. Zhang, B. R. Childers, and J. Yang, “FPB: Fine-grained Power Budgeting to Improve Write Throughput of Multi-level Cell Phase Change Memory,” Dec. 2012, doi: https://doi.org/10.1109/micro.2012.10.

[33] L. Jiang, B. Zhao, J. Yang, and Y. Zhang, “A low power and reliable charge pump design for Phase Change Memories,” Jun. 2014, doi: https://doi.org/10.1109/isca.2014.6853194.

[34] anysilicon, “Introduction to eDRAM,” *AnySilicon*, Feb. 11, 2024. https://anysilicon.com/introduction-to-edram/

[35] T. Yoo, H. Kim, Q. Chen, Tony Tae-Hyoung Kim, and B. Kim, “A Logic Compatible 4T Dual Embedded DRAM Array for In-Memory Computation of Deep Neural Networks,” Jul. 2019, doi: https://doi.org/10.1109/islped.2019.8824826.

[36] Fatih Hamzaoglu *et al.*, “A 1 Gb 2 GHz 128 GB/s Bandwidth Embedded DRAM in 22 nm Tri-Gate CMOS Technology,” *IEEE journal of solid-state circuits*, vol. 50, no. 1, pp. 150–157, Jan. 2015, doi: https://doi.org/10.1109/jssc.2014.2353793.

[37] G. Fredeman *et al.*, “17.4 A 14nm 1.1Mb embedded DRAM macro with 1ns access,” Feb. 2015, doi: https://doi.org/10.1109/isscc.2015.7063053.

[38] Y.-P. Fang, B. Vaidyanathan, and A. S. Oates, “Soft error rate cross-technology prediction on embedded DRAM,” Jan. 2009, doi: https://doi.org/10.1109/irps.2009.5173382.

[39] Nasser Kurd *et al.*, “Haswell: A Family of IA 22 nm Processors,” *IEEE Journal of Solid-state Circuits*, vol. 50, no. 1, pp. 49–58, Jan. 2015, doi: https://doi.org/10.1109/jssc.2014.2368126.

[40] Y. Long *et al.*, “A ferroelectric FET based power-efficient architecture for data-intensive computing,” Nov. 2018, doi: https://doi.org/10.1145/3240765.3240770.

[41] Y. Long *et al.*, “A Ferroelectric FET-Based Processing-in-Memory Architecture for DNN Acceleration,” vol. 5, no. 2, pp. 113–122, Jun. 2019, doi: https://doi.org/10.1109/jxcdc.2019.2923745.

[42] J.-P. Muller, T. S. Böscke, Uwe Schröder, R.-D. Hoffmann, T. Mikolajick, and L. Frey, “Nanosecond Polarization Switching and Long Retention in a Novel MFIS-FET Based on Ferroelectric $\hbox{HfO}_{2}$,” vol. 33, no. 2, pp. 185–187, Feb. 2012, doi: https://doi.org/10.1109/led.2011.2177435.

[43] M. Trentzsch *et al.*, “A 28nm HKMG super low power embedded NVM technology based on ferroelectric FETs,” *IEEE Xplore*, Dec. 01, 2016. https://ieeexplore.ieee.org/document/7838397 (accessed May 31, 2023).

[44] Halid Mulaosmanovic *et al.*, “Novel ferroelectric FET based synapse for neuromorphic systems,” Jun. 2017, doi: https://doi.org/10.23919/vlsit.2017.7998165.

[45] Y. Xie, “Modeling, Architecture, and Applications for Emerging Memory Technologies,” *IEEE Design & Test of Computers*, vol. 28, no. 1, pp. 44–51, Jan. 2011, doi: https://doi.org/10.1109/mdt.2011.20.

[46] Y. Kim and J. Park, “Energy-Efficient STT-MRAM based Digital PIM supporting Vertical Computations Using Sense Amplifier,” *2022 19th International SoC Design Conference (ISOCC)*, Oct. 2022, doi: https://doi.org/10.1109/isocc56007.2022.1003 1290

[47] Kazi Asifuzzaman, Rommel Sánchez Verdejo, and Petar Radojković, “Enabling a reliable STT-MRAM main memory simulation,” *Proceedings of the International Symposium on Memory Systems*, Oct. 2017, doi: https://doi.org/10.1145/3132402.3132416.

[48] V. B. Naik, H. Meng, and R. Sbiaa, “Thick CoFeB with perpendicular magnetic

anisotropy in CoFeB-MgO based magnetic tunnel junction," *AIP Advances*, vol. 2, no. 4, p. 042182, Dec. 2012, doi: https://doi.org/10.1063/1.4771996.

[49] Kazi Asifuzzaman *et al.*, "Performance Impact of a Slower Main Memory," *UPCommons institutional repository (Universitat Politècnica de Catalunya)*, Oct. 2016, doi: https://doi.org/10.1145/2989081.2989082.

[50] V. Seshadri et al., "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture, 2017, pp. 273–287

[51] S. Roy, M. Ali, A. Raghunathan, PIM-DRAM: Accelerating machine learning workloads using processing in commodity DRAM, 2021, arXiv:2105.03736

[52] O. Leitersdorf, R. Ronen, S. Kvatinsky, MultPIM: Fast stateful multiplication for processing-in-memory, 2021, arXiv:2108.13378.

[53] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "The desired memristor for circuit designers," IEEE Circuits Syst. Mag., vol. 13, no. 2, pp. 17–22, 2nd Quart., 2013.

[54] Y. Long, T. Na, S. Mukhopadhyay, ReRAM-based processing-in-memory architecture for recurrent neural network acceleration, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 26 (12) (2018) 2781–2794, http://dx.doi.org/10.1109/TVLSI.2018.2819190.

[55] A. Kazemi, M.M. Sharifi, Z. Zou, M.T. Niemier, X.S. Hu, M. Imani, MIMHD: Accurate and efficient hyperdimensional inference using multi-bit in-memory computing, in: IEEE/ACM International Symposium on Low Power Electronics and Design, ISLPED 2021, Boston, MA, USA, July 26-28, 2021, IEEE, 2021, pp. 1–6, http://dx.doi.org/10.1109/ISLPED52811.2021.9502498.

[56] W.J. Lee, C.H. Kim, Y. Paik, J. Park, I. Park, S.W. Kim, Design of processing-"inside"-memory optimized for DRAM behaviors, IEEE Access 7 (2019) 82633–82648, http://dx.doi.org/10.1109/ACCESS.2019.2924240

[57] S. Gupta, M. Imani, H. Kaur, T.S. Rosing, NNPIM: A processing in-memory architecture for neural network acceleration, IEEE Trans. Comput. 68 (9) (2019) 1325–1337, http://dx.doi.org/10.1109/TC.2019.2903055.

[58] A. Roohi, S. Angizi, D. Fan, R.F. DeMara, Processing-in-memory acceleration of convolutional neural networks for energy-efficiency, and power-intermittency resilience, 2019, arXiv:1904.07864.

[59] C.-C. Lin, C.-L. Lee, J.-K. Lee, H. Wang, M.-Y. Hung, Accelerate binarized neural networks with processing-in-memory enabled by RISC-V custom instructions, in: 50th International Conference on Parallel Processing Workshop, Association for Computing Machinery, 2021, doi: https://doi.org/10.1145/3458744.3473351.

[60] A. Farmahini-Farahani, J. H. Ahn, K. Morrow and N. S. Kim, "NDA: Near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules," 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), Burlingame, CA, USA, 2015, pp. 283-295, doi: 10.1109/HPCA.2015.7056040.

[61] Youngeun Kwon, Yunjae Lee, and Minsoo Rhu. Tensordimm: A practical near-memory processing architecture for embeddings and tensor operations in deep learning. In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, pages 740– 753, 2019.

[62] Liu Ke, Udit Gupta, Benjamin Youngjae Cho, David Brooks, Vikas Chandra, Utku Diril, Amin Firoozshahian, Kim Hazelwood, Bill Jia, Hsien-Hsin S Lee, et al. 2020. Recnmp: Accelerating personalized recommendation with nearmemory processing. In Proceedings of the 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA'20). IEEE.

[63] Z. Zhou, C. Li, F. Yang and G. Sun, "DIMM-Link: Enabling Efficient Inter-DIMM Communication for Near-Memory Processing," 2023 IEEE International

Symposium on High-Performance Computer Architecture (HPCA), Montreal, QC, Canada, 2023, pp. 302-316, doi: 10.1109/HPCA56546.2023.10071005.

[64] Cong Li, Zhe Zhou, Yang Wang, Fan Yang, Ting Cao, Mao, Yang, Yun Liang, Guangyu, Sun "PIM-DL: Expanding the Applicability of Commodity DRAM-PIMs for Deep Learning via Algorithm-System Co-Optimization," Apr. 2024, doi: https://doi.org/10.1145/3620665.3640376.

[65] X. W. X. W. G. S. Zhe Zhou, Cong Li, "Gnnear: Accelerating full-batch training of graph neural networks with near-memory processing," in Proceedings of the 31st International Conference on Parallel Architectures and Compilation Techniques (PACT), 2022.

[66] S. Lee, S.-h. Kang, J. Lee, H. Kim, E. Lee, S. Seo, H. Yoon, S. Lee, K. Lim, H. Shin, J. Kim, O. Seongil, A. Iyer, D. Wang, K. Sohn, N.S. Kim, 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, 2021, pp. 43–56, http://dx.doi.org/10.1109/ ISCA52012.2021.00013.

[67] Kang, Shinhaeng, Sukhan Lee, Byeongho Kim, Hweesoo Kim, Kyomin Sohn, Nam Sung Kim, and Eojin Lee. "An FPGA-based RNNT Inference Accelerator with PIM-HBM." In Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 146-152. 2022. doi: https://doi.org/10.1145/3490422.3502355.