SCIENCE CHINA
Information Sciences

• REVIEW •

October 2025, Vol. 68, Iss. 10, 201401:1–201401:21
https://doi.org/10.1007/s11432-024-4354-y

Expansion of the memory pyramid in the era of large models: compute-intensive compute-in-memory and memory-intensive compute-in-memory

Zhichao LIU[1], Yanqi ZHANG[1], Zhican ZHANG[1], Yi YANG[1], Zhaoyang ZHANG[1], Xing WANG[1], Jinwu CHEN[1], Xiaomin LI[2], Xin SI[1]* & Jun YANG[1]

[1] *National Application Specific Integrated Circuit Center, Southeast University, Nanjing 210096, China*
[2] *National Nanjing Starting Line Wearable Microelectronics Technology Co., Ltd., Nanjing 210032, China*

Received 27 September 2024/Revised 30 December 2024/Accepted 19 March 2025/Published online 2 September 2025

Abstract The rise of large-scale models has significantly increased the demand for high computational power and throughput in artificial intelligence (AI) chips, presenting challenges for existing architectures. Compute-in-memory (CIM) has emerged as a promising solution to address these bottlenecks. This study redefines the traditional computing architecture pyramid by introducing the CIM pyramid, structured around different storage media. CIM architectures are classified into two categories: compute-intensive CIM, which leverages static random-access memory (SRAM) and embedded dynamic random access memory (eDRAM), and memory-intensive CIM, which utilizes DRAM and non-volatile memory (NVM). The research reviews and analyzes recent advancements in both compute-intensive and memory-intensive CIM, highlighting their development trends and challenges. Furthermore, the study suggests a hybrid, heterogeneous architecture that integrates both CIM types with traditional computing systems, aiming to address the diverse computational needs of large models in the future.

Keywords compute-in-memory (CIM), artificial intelligence (AI), memory pyramid, compute-intensive, memory-intensive

Citation Liu Z C, Zhang Y Q, Zhang Z C, et al. Expansion of the memory pyramid in the era of large models: compute-intensive compute-in-memory and memory-intensive compute-in-memory. Sci China Inf Sci, 2025, 68(10): 201401, https://doi.org/10.1007/s11432-024-4354-y

1 Introduction

In recent years, the field of artificial intelligence (AI) has experienced rapid development, with its applications expanding from image recognition to e-commerce recommendation systems, object detection in autonomous vehicles, and generative models (such as creating realistic images and coherent text). As the scale of training data and parameters continues to increase, AI has made significant progress in areas such as natural language processing (NLP) and generative AI, establishing large-scale AI models as a key milestone in the evolution of the field. The rise of these large AI models brings both new opportunities and significant challenges to hardware design.

The traditional von Neumann architecture presents a significant bottleneck in further enhancing AI computational performance due to the memory wall issue [1]. In conventional methods, data are continuously moved from memory to the cache for processing during computation, leading to substantial energy consumption from data transfer. However, memory technology has struggled to keep pace with advancements in processor technology. In traditional architectures, transferring data from memory units to processing units consume a significant amount of power, resulting in a low proportion of energy and time being actually dedicated to computation. The development of memory performance lags far behind that of processors, thereby limiting the improvement of processor performance. Compute-in-memory (CIM) is considered one of the potential solutions to effectively address the current bottlenecks in chip design [2]. After years of rapid development, both academia and industry have conducted extensive research on overall CIM architectures and computational paradigms, demonstrating the feasibility of CIM designs and their immense potential in AI applications [3–7].

* Corresponding author (email: xinsi@seu.edu.cn)

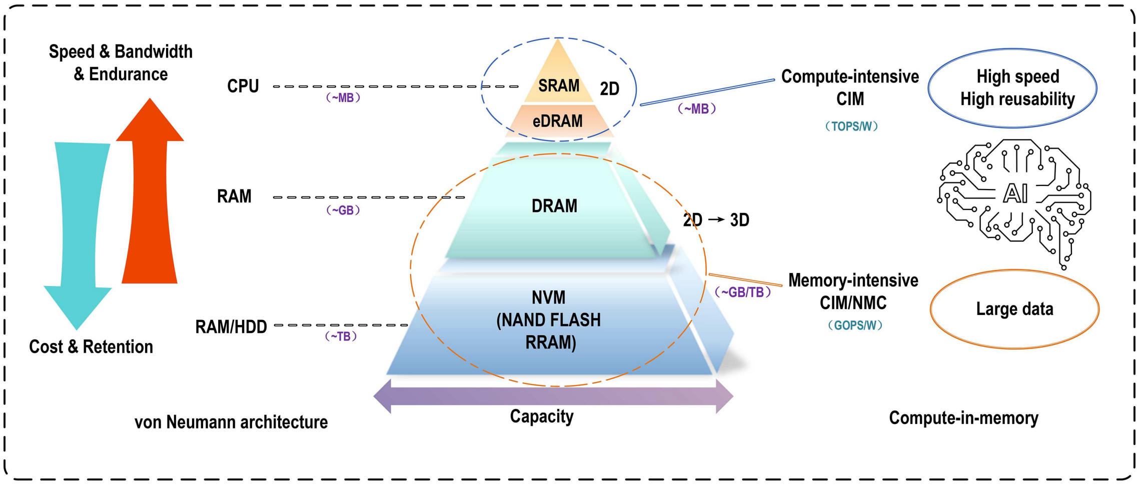


Figure 1 Memory pyramid of the von Neumann architecture vs. the memory pyramid of compute-in-memory architecture by medium.

The hierarchical structure of the modern computer memory pyramid is illustrated in Figure 1. Each type of memory has specific advantages and disadvantages compared to other types of memory, with distinct roles within computer architecture. For example, parts of static random-access memory (SRAM) and dynamic random-access memory (DRAM), such as embedded dynamic random access memory (eDRAM), have fast read/write speeds, no limits on read/write cycles, and low power consumption, but exhibit low storage density and high unit costs, making them primarily used in cache systems. On the other hand, DRAM and non-volatile memory (NVM) have relatively slower read/write speeds but offer lower costs and higher storage capacities, making them suitable for use as memory and hard disk components within computer architectures.

As shown in Figure 1, if CIM becomes involved in the data computation and processing pipeline, the memory hierarchy would also adapt accordingly. Taking AI models as an example, CIM can be categorized into two main types: compute-intensive CIM for computation-heavy operators and memory-intensive CIM for memory-heavy operators. Compute-intensive CIM focuses on enhancing computation density, primarily through in-memory computing (IMC), while memory-intensive CIM targets improvements in memory bandwidth, mainly through processor-in-memory (PIM) and near-memory computing (NMC).

In compute-intensive CIM, the memory architecture corresponds to the cache in the von Neumann architecture, primarily utilizing SRAM and eDRAM. Memory-intensive CIM aligns with the traditional architecture's memory and storage components, predominantly relying on DRAM and NVM. Extensive research and exploration have already been conducted across various levels of the memory hierarchy concerning CIM [8, 9], and selecting the appropriate memory types and structures is crucial for effective CIM implementations. In the CIM hierarchy, the storage capacity and computational energy efficiency of various memory technologies differ significantly. SRAM typically provides storage capacity at the megabyte (MB) scale, DRAM at the gigabyte (GB) scale, and NVM at the terabyte (TB) scale. Similarly, the energy efficiency of CIM architectures varies across different storage media. Compute-intensive CIM systems generally achieve energy efficiencies on the order of tera-operations per second per watt (TOPS/W), whereas memory-intensive CIM systems typically exhibit energy efficiencies on the order of giga-operations per second per watt (GOPS/W). Different types of CIM should leverage different storage media to optimize their respective strengths.

With the advent of the era of large models, existing AI chip architectures face new challenges. Large neural networks require substantial computational power for training [10]. Over the span of seven years, the number of parameters in neural networks has surged from 60 million in AlexNet to 175 billion in OpenAI GPT-3, a 2917-fold increase, while the computational demands for training neural networks have grown by 300000 times. These computational requirements far exceed the pace of semiconductor development predicted by Moore's law. At the same time, the Transformer model imposes higher memory access demands during the generation process. Currently, the computational speed of the world's most advanced AI chips significantly outpaces the memory bandwidth, yet the acceleration of memory-intensive, data-heavy operators remains limited.

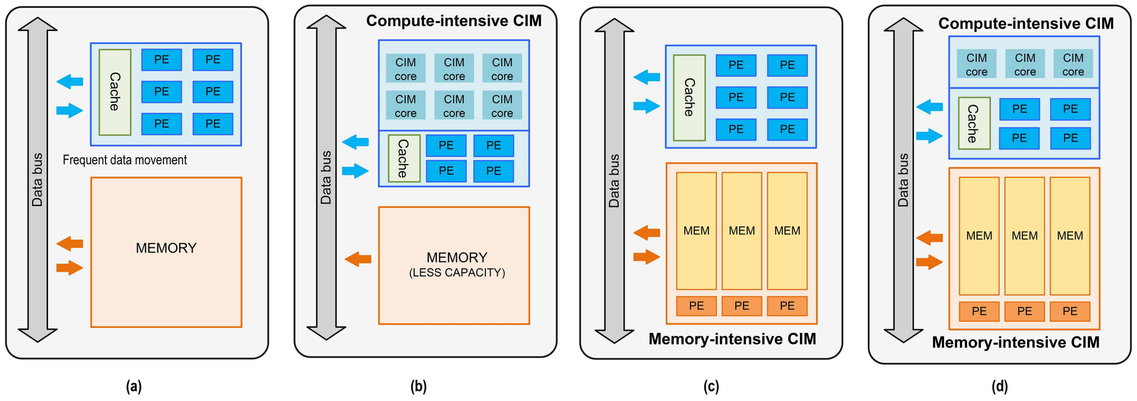


Figure 2 Different computing architectures categorized based on their computational and memory characteristics. (a) The traditional von Neumann architecture; (b) computation-intensive CIM architecture; (c) memory-intensive CIM architecture; (d) heterogeneous CIM architecture.

As shown in Figure 2(a), in traditional computing architectures, when large-scale data operations are performed, the movement of data between on-chip/off-chip memory and processing units significantly increases latency and energy consumption. As shown in Figures 2(b) and (c), to address the challenges posed by different operators, CIM has evolved into two distinct architectures: compute-intensive CIM and memory-intensive CIM, forming an important extension of the memory hierarchy. In the era of large models, it is essential to balance high parallelism and computational density while also meeting the high memory access demands during decoder operations. This indicates that, as application scenarios evolve, CIM should adopt different architectures (Figure 2(d)) for different applications, with compute-intensive and memory-intensive CIM complementing each other, driving the development toward heterogeneous architecture integration.

The remainder of the study is organized as follows. Section 2 introduces the challenges posed by compute-intensive and memory-intensive operators as well as those introduced by large AI models. Section 3 analyzes recent developments in compute-intensive CIM, with a focus on the design of SRAM-CIM macros and eDRAM-CIM macro units. Section 4 examines memory-intensive CIM. Section 5 discusses the challenges and development trends of CIM in the era of large models. The conclusion is presented in Section 6.

2 Challenges posed by AI and large models to hardware design

With the rise of AI large models, balancing computation and memory has become increasingly crucial. Developing efficient operators, optimizing memory usage, and designing specialized hardware will be key to addressing these challenges [11]. The design of hardware accelerators (such as GPUs and TPUs) and dedicated AI chips (such as ASICs) must strike a balance between computational power and memory bandwidth. This section focuses on comparing different types of operators and analyzing the distinct characteristics of large models to identify the requirements they impose on hardware design.

2.1 Compute-intensive operators and memory-intensive operators

The commonly used metrics for evaluating the performance of neural network models include computational load, parameter count, memory access volume, and memory footprint. These metrics assess the model's size from different dimensions. Computational load refers to the number of operations required by the model and reflects the demand on hardware processing units, making it the most commonly used metric for assessing model size. Memory access volume, on the other hand, refers to the number of bytes the model needs to access during computation, which indicates the demand on memory bandwidth. Although often overlooked, memory access volume can significantly impact overall system performance.

Correspondingly, the inference speed of a model on hardware is not solely influenced by computational load but is also affected by factors such as memory access volume, hardware characteristics, software implementation, and system environment. These factors collectively determine the overall efficiency and performance of the model on a given hardware platform.

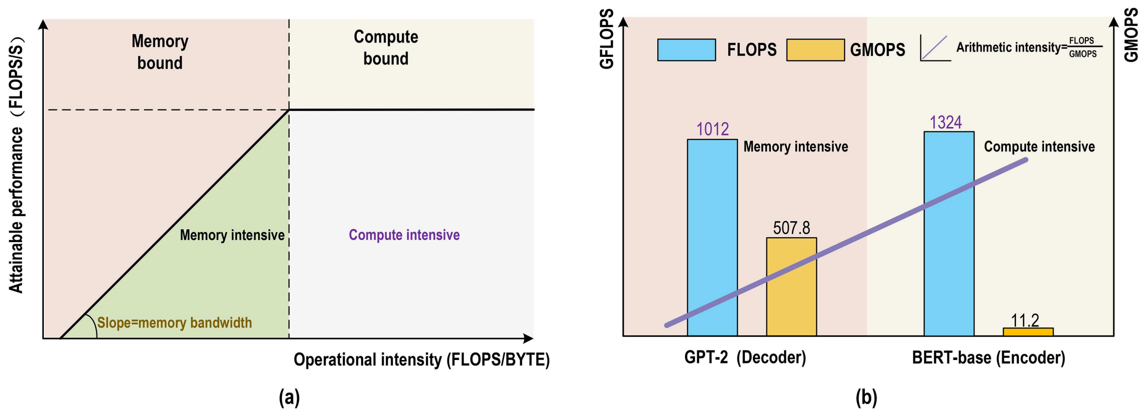


Figure 3 Key concepts in computational performance and model operations. (a) Roofline performance model; (b) two different types of operators in large models.

To evaluate the performance of models on processors, Williams et al. [12] proposed a throughput-oriented performance model called Roofline, which combines computational intensity with computational performance. Computational intensity refers to the amount of computation required by a program per unit of memory access, also known as the compute-to-memory ratio. This model provides a framework for understanding the balance between a system's computational power and memory bandwidth, helping to identify the limiting factors in a program's performance and optimize its execution on hardware.

As shown in Figure 3(a), when computational intensity is low, the program involves frequent memory accesses with fewer computations, and its performance is constrained by memory bandwidth. This type of program is referred to as memory-bound or memory-intensive. In this region, the performance upper bound is equal to the product of computational intensity and memory bandwidth, represented by the sloped line in Figure 3(a), where the slope corresponds to the size of the memory bandwidth. Conversely, when computational intensity is high, the program's performance is limited by the hardware's peak computational capacity, making it compute-bound or compute-intensive. This region is depicted by the horizontal blue line, where the performance upper bound is determined by the hardware's computational power. In this scenario, computational speed is independent of computational intensity, but as computational intensity increases, the required memory bandwidth decreases.

According to the Roofline model, in the compute-bound region, inference time scales linearly with computational load, whereas for memory-bound operators, the primary determinant of performance is memory access speed.

Traditional deep learning operators can be classified as either computation-intensive or memory-intensive based on the criteria above. Compute-intensive operators rely heavily on the computational power of processors or compute units. Their performance is typically limited by the processing speed and parallel computing capabilities of the processor, rather than the speed of data transfer or storage. These operators require significant amounts of floating-point or integer operations during execution. Memory-intensive operators, on the other hand, are constrained by data storage and transfer speeds. During their execution, they frequently access memory or storage, with their performance often limited by memory bandwidth or latency rather than the computational speed of the processing units.

The key metric for evaluating compute-intensive programs is usually computational performance, as they tend to have good data reuse and locality, for example, convolution and fully connected layers. In contrast, memory-intensive operators spend the majority of their time accessing memory. These include element-wise operators such as rectified linear unit (ReLU), element-wise summation, and, in some cases, depth-wise convolution.

2.2 Demands and challenges of large models

In the fields of computer science and artificial intelligence, large models have garnered significant attention due to their superior performance and wide range of applications. These models often contain billions or even trillions of parameters, enabling them to tackle complex tasks such as NLP, image recognition and generation, and speech recognition. However, the training and inference processes of large models are highly energy-intensive, making the improvement of computational and memory efficiency a critical area of research. The high performance of large models is accompanied by enormous data storage demands, as these models require vast amounts of storage for parameters and intermediate computation results during task processing. This creates challenges for traditional memory and computing architectures.

Transformers and large language models (LLMs), used to solve various NLP tasks, pose new challenges for hardware design. The Transformer architecture was initially introduced as an encoder-decoder model for machine translation tasks. The encoder focuses on the preliminary processing and feature extraction of the entire input sequence, which demands significant computational resources to perform complex matrix operations and neural network transformations, categorizing it as a compute-intensive operator. In contrast, the decoding phase is primarily concerned with the step-by-step generation of the output sequence, requiring frequent access to and updating of previously generated elements and intermediate representations, which involves substantial storage and data transfer operations, making it a memory-intensive operator.

During the training and inference of Transformer models, the significant data movement between memory and computational units becomes a critical factor in limiting bandwidth and computational power. In subsequent developments, architectures with encoder-only and decoder-only components were introduced, separating the original encoder-decoder structure into individual components. Encoder-only large models can process input sequences in parallel, with the entire sequence passing through repeated encoder modules in one go. However, decoder-only large models must generate outputs sequentially, as they are inherently autoregressive, meaning they generate one token at a time based on previously generated tokens. This fundamental difference in processing places distinct demands on hardware, further emphasizing the challenges posed by large models.

An analysis of floating-point operations and memory access for the bidirectional encoder representations from transformers (BERT)-base encoder and the GPT-2 decoder [13], as shown in Figure 3(b), reveals distinct differences in computational and memory access patterns. BERT requires significantly fewer memory accesses while performing a large number of computation operations, making it more reliant on computational resources, with memory bandwidth being a relatively minor bottleneck. In contrast, GPT-2 frequently accesses memory, exhibiting much lower arithmetic intensity compared to BERT's encoder. This clearly illustrates that the encoder is compute-intensive, whereas the decoder is memory-intensive.

This comparison highlights the challenges that large models pose to underlying inference hardware. As encoder-based models, such as BERT, are compute-intensive, the computational demands on chips grow rapidly with the increasing scale of large models. On the other hand, since the decoding phase in models like GPT-2 is memory-intensive, improving chip bandwidth is crucial to reduce latency. The development of large models places increasing pressure on hardware design to simultaneously boost computational power and memory bandwidth to meet the distinct needs of both encoder and decoder operations.

In this context, the two architectural paradigms of CIM could prove highly effective. The two distinct types of operators in large models can be accelerated using either compute-intensive or memory-intensive CIM architectures. Compute-intensive CIM is better suited for operators like encoders, which feature high data reuse, while memory-intensive CIM is more advantageous for improving memory speed and bandwidth, making it well-suited for decoders and certain activation operators.

Compute-intensive CIM typically performs calculations directly within memory units, addressing the needs of high arithmetic intensity and operators with high data reuse. This approach is particularly effective for complex computational tasks such as matrix operations and convolutions. By executing computations within memory, this paradigm not only enhances computational efficiency but also reduces power consumption, meeting the massive computational resource demands of large models. The primary focus of compute-intensive CIM is to efficiently parallelize large-scale computational operations within memory, reducing the reliance on external processing units. Additionally, the computational operations within memory units must be flexible enough to accommodate different types of tasks. This design strategy can significantly improve the computational efficiency of large models during the inference process.

Memory-intensive CIM, on the other hand, addresses the demands of large-scale data access and processing. The decoding phase is often memory-intensive, requiring frequent access to memory to retrieve model parameters and intermediate results. In such scenarios, the bandwidth and memory access efficiency of the chip become performance bottlenecks. The design of memory-intensive CIM emphasizes high-bandwidth memory access capabilities and efficient data transfer mechanisms. By significantly increasing the bandwidth between memory units and external processing units, this architecture supports rapid data exchange. For large models, memory-intensive CIM enables more efficient processing of vast amounts of data during the decoding phase, thus reducing overall system latency.

By adopting these two complementary CIM paradigms, hardware can be better tailored to the differing computational and memory demands of large AI models, optimizing both computation and data transfer efficiency.

The development of large models presents significant challenges to underlying inference chips. The encoder phase is compute-intensive, demanding substantial computational power, whereas the decoder phase is data-intensive, relying on frequent memory access. To enhance the performance of large models, compute-intensive CIM and memory-intensive CIM offer effective solutions for these two types of operators. Compute-intensive CIM reduces the reliance on external processing units by performing large-scale computations directly within memory units, thereby improving computational efficiency. On the other hand, memory-intensive CIM enhances memory bandwidth and access efficiency, optimizing data processing during the decoding phase.

In Sections 3 and 4, we will summarize and discuss the current advancements in these two CIM architectural paradigms.

3 Compute-intensive CIM

Compute-intensive CIM corresponds to the high data reuse and computational density of computation-heavy operators, primarily aiming to achieve high computational power and energy efficiency. Its main application scenarios include AI operator acceleration, such as convolution operations in convolutional neural networks (CNNs) and the encoder part of transformer models. CNNs and fully convolutional networks (FCNs) require a large number of pixel-wise and channel-wise multiply-accumulate (MAC) operations, performing extensive MAC calculations per memory access, which consumes significant energy. In the encoder part of transformers, the entire input sequence is fed into the model for extensive matrix-matrix multiplications and element-wise additions. To achieve high computational power, computation-intensive CIM integrates computing units into memory arrays, enabling large-scale parallel processing.

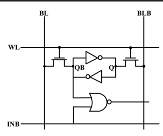
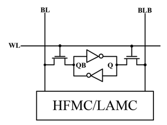
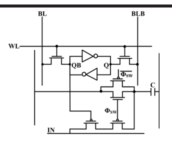
Compute-intensive CIM is mainly classified into SRAM-CIM, eDRAM-CIM, and some non-volatile memory-based NVM-CIM. Among them, SRAM and eDRAM-based CIM primarily rely on in-memory computing. SRAM, with its stable storage through a dual-cross inverter logic structure, has relatively low power consumption during read and write operations and can implement CIM calculations using various methods. eDRAM, which stores data using capacitors, has a higher storage density compared to SRAM but suffers from lower stability. Both methods typically store weight values in memory arrays and perform bitwise multiplication with input feature values, followed by accumulation along dedicated paths to execute rapid MAC operations.

This section reviews the silicon validation of compute-intensive CIM from both academia and industry over the past few years, categorizing these developments based on the storage medium used, specifically into SRAM-CIM and eDRAM-CIM.

3.1 SRAM-CIM

In recent years, SRAM-CIM has seen rapid development, with continuous improvements in supported MAC precision and an expanding range of operator types. Many designs are based on SRAM for several key reasons. First, SRAM technology is mature, offering high data read/write speeds, and its performance has significantly improved with advances in semiconductor processes. Second, SRAM is compatible with advanced CMOS (complementary metal-oxide-semiconductor) processes, providing faster operation speeds and greater durability, which are essential for supporting high computational power. Third, SRAM can achieve computational precision consistent with traditional digital computing methods, eliminating the need for complex retraining processes, which enhances the applicability of AI models. This capability also lays the foundation for high computational efficiency and density, making SRAM-CIM highly suitable for compute-intensive operators. Additionally, the inherent stability of SRAM and its energy efficiency during operations further strengthen its role in enabling high-performance, large-scale parallel computations directly within memory. These characteristics make SRAM-CIM an ideal choice for accelerating AI tasks that require intensive MAC operations, such as convolution and matrix multiplications in deep learning models. Table 1 presents several recent studies on SRAM-CIM.

Table 1 Comparison table of compute-intensive SRAM-CIM studies.

			
Reference	JSSC2023 [14]	JSSC2024 [15]	ISSCC2024 [16]
Process	28 nm	28 nm	28 nm
Cell structure	6T+NOR	DB6T+HFMC/LAMC	10T CR-CIM cell+C
Domain	Digital	Digital	Charge
Precision	INT8/INT16	INT8/BF16	INT4/INT8
Model	Transformer	CNN, Yolo, U-Net	CNN/Transformer
Area efficiency	0.22 TOPS/mm^2	0.61–2.05 TFLOPS/mm^2	0.1–0.45 TOPS/mm^2
Energy efficiency	16.22–70.2 TOPS/W	14.04–31.6 TFLOPS/W	29.6–148.5 TOPS/W

3.1.1 *TranCIM: full-digital bitline-transpose CIM-based sparse transformer accelerator*

Tu et al. [14] proposed TranCIM, a full-digital bitline-transpose CIM-based sparse transformer accelerator with both pipeline and parallel reconfigurable modes. The pipeline mode is designed to reduce off-chip memory access for attention layers, while the parallel mode enables high parallelism in fully connected (FC) layers. This design is based on full-digital SRAM-CIM, supporting INT8 and INT16 precision, and further optimizes computational resource usage through a sparse attention scheduler (SAS).

Fabricated using 28 nm CMOS technology, the TranCIM chip operates at a supply voltage of 0.6–1.0 V and a frequency of 80–240 MHz. In terms of energy consumption, the chip consumed 15.59 μJ per token for the BERT-base model and achieved 1.48–20.5 TOPS/W at INT8 precision and 0.37–5.1 TOPS/W at INT16 precision. These results demonstrate TranCIM's efficiency in handling the computational demands of transformer-based models, especially in sparse attention and fully connected layers.

3.1.2 *A 28 nm 64-kb digital-domain floating-point SRAM-CIM macro*

Guo et al. [15] proposed a floating-point digital CIM macro based on a double-bit 6T SRAM structure. The double-6T structure, utilizing split word lines on the bit-cell, allows two-bit weights to be read out in the same cycle for calculation. This design implements the new ShareFloatv2 floating-point data type within the CIM array and uses a high-bit full-precision multiply cell (HFMC) alongside a low-bit approximate-calculation multiply cell (LAMC) to reduce internal bandwidth and area costs.

By employing a hybrid partial approximation and partial full-precision computation, the design approach strikes an optimal balance between energy efficiency, spatial efficiency, and computational accuracy. This work achieved an energy efficiency of 31.6 TFLOPS/W and an area efficiency of 2.05 TFLOPS/mm^2 for floating-point MAC operations, demonstrating its potential for high-performance floating-point computations in energy-constrained environments.

3.1.3 *CR-CIM: a capacitor-reconfigured CIM macro for CNNs and transformers*

Yoshioka [16] proposed a capacitor-reconfigured CIM macro (CR-CIM) designed for the unified acceleration of both CNNs and Transformers. In CNN mode, the design utilized bit-parallel computing and an 8-bit analog-to-digital converter (ADC), focusing on energy efficiency but with lower computational accuracy. In Transformer mode, it switched to bit-serial computing and a 10-bit ADC to improve the carrier-to-noise ratio (CSNR), thus enhancing computational accuracy.

The key innovation in CR-CIM was the reconfiguration of capacitors in the array for dual purposes: computation and ADC implementation, which helped reduce the area overhead in the charge domain. This dual-mode operation allowed for optimization across different tasks, balancing energy efficiency and accuracy according to the application requirements. CR-CIM achieved an energy efficiency of 29.6–48.5 TOPS/W (normalized to 28 nm INT8), demonstrating its capability to efficiently accelerate both CNN and Transformer-based models.

3.2 eDRAM-CIM

In recent years, with the rapid development of deep learning technologies, the scale of networks and the volume of computations have experienced explosive growth, which, from an application perspective, has placed higher demands on computation-intensive CIM for both memory density and computational power density. However, due to the inherent 6T storage structure of SRAM cells, the potential for improving the memory density and computational power density in SRAM-based CIM designs is limited. eDRAM offers several advantages, including higher memory density, lower memory access power consumption, and fast read/write speeds. As a result, eDRAM-based CIM designs have emerged as a promising solution to address the density bottlenecks of SRAM-CIM, attracting significant attention from researchers.

This section provides a review of several eDRAM CIM macro designs and their contributions to overcoming these challenges.

3.2.1 *eDRAM CIM: a 1T1C eDRAM CIM macro*

Xie et al. [17] proposed the first charge-domain CIM macro based on a 1T1C eDRAM cell, supporting 8-bit inputs and 8-bit weights. In this design, the storage node capacitance of some eDRAM cells is reconfigured into a weight capacitor array, enabling operations such as data conversion, MAC, pooling, and ReLU activation. This reconfiguration reduces the computational overhead at the accelerator level and enhances the reconfigurability of the macro unit.

Implemented on a 65 nm process node, this study achieved a throughput of 4.71 GOPS, an energy efficiency of 4.76 TOPS/W, and an area efficiency of 8.26 GOPS/mm^2, demonstrating the potential of eDRAM CIM designs to offer both high energy and area efficiency in deep learning accelerators.

3.2.2 *Calibration-free 15-level/cell eDRAM CIM macro*

Song et al. [18] proposed a 3T1C eDRAM cell that supports 8-level current programming. This cell employs a dynamic cascaded structure to perform weight writing and MAC computations in the current domain, reducing the impact of transistor threshold voltage variations and nonlinear I-V characteristics on computational accuracy. As a result, a single memory cell can store 15 levels of weight without the need for calibration.

The macro supports operations with 4-bit inputs and 4-bit weights. Fabricated using a 28 nm process technology, it achieved an energy efficiency of 233–304 TOPS/W for 4-bit computations and an area efficiency of 4.74 TOPS/mm^2. With a refresh interval of 0.4 ms, the macro achieved over 90% inference accuracy on the CIFAR-10 dataset.

3.2.3 *eDRAM-LUT-based DCIM macro*

He et al. [19] proposed a full-precision digital CIM macro based on eDRAM-LUT (look-up table) technology, supporting inputs ranging from 1 to 8 bits and 8-bit weights. This macro consists of 256 × 160 3T eDRAM cells, which are configured as a look-up table in CIM mode to perform MAC operations. In storage mode, it functions as a 40 kb memory array. The design incorporates in-memory encoding and refresh circuitry to reduce external read/write overhead.

This LUT-based in-memory computation approach offers a novel solution for balancing the trade-off between memory density and computational density. The study achieved an energy efficiency of 18.1 TOPS/W in 8-bit CIM mode, an area efficiency of 11.8 TOPS/mm^2, and a storage density of 2.4 Mb/mm^2, demonstrating its potential for high-density and efficient computation in deep learning applications.

As the aforementioned research demonstrates, recent work in this field typically involves structural modifications or additional specialized circuitry to support computation, closely linked to memory cells, memory arrays, and peripheral circuits for computation. These modified cell designs often result in a significant reduction in memory density. However, for applications that achieve high performance, the reduction in area efficiency can be justified when the applications exhibit high data reuse or high GOPs per byte, as the area cost can be amortized over time through repeated data reuse.

The primary goal of in-memory computing (IMC) is to enhance both energy and area efficiency. From the perspective of macro-level designs, improving precision and supporting a broader range of operator types are key research focuses. Optimizing analog readout circuits and digital adder trees is also critical for boosting energy efficiency. From an accelerator perspective, exploring more efficient dataflows is an important area of investigation. For compute-intensive CIM, the specific operators used in different compute-intensive tasks (e.g., matrix multiplication in the self-attention mechanism of large model encoders and tasks in edge-based BERT NLP networks) are essentially the same, primarily consisting

Table 2 Comparison table of different metrics for DRAM-PIM (memory-intensive CIM).

	AMD MI100 GPU [23]	GDDR6-AiM [24]	UPMEM PIM [25]
Year	2023	2022	2018
Memory	DRAM (HBM)	DRAM (GDDR6)	DRAM (DDR4)
Type	NMC	NMC (single core)	NMC
Memory bandwidth	1.2 TB/s	16 GB/s	2 TB/s
Capacity	2.25 TB	8 GB	160 GB
Application	AI	AI	AI/Genomics

of MAC operations for matrix multiplication. Compute-intensive CIM macro can achieve scalability for matrix multiplications of different sizes by extending the length of MAC operations and the depth of memory.

In addition, NVM technologies [20, 21] have been increasingly employed for compute-intensive CIM in recent years. Choosing the appropriate storage medium is crucial for improving computational efficiency. These topics will be further discussed in Section 5.

4 Memory-intensive CIM/NMC

Memory-intensive CIM/NMC aims to address the critical challenges of LLMs access and bandwidth constraints. This paradigm integrates storage and computing units. The computing units are placed close to the memory to maximize memory access bandwidth and reduce data movement, thereby enhancing overall computing efficiency.

Memory-intensive CIM is mainly based on DRAM and NVM, categorized into storage up-shift and computation down-shift. Storage up-shift utilizes advanced packaging technologies to position memory near the processor, increasing the number of links between computing and storage to improve memory access bandwidth. Typical products include high bandwidth memory (HBM) and hybrid memory cube (HMC), which achieve high bandwidth and low power consumption through vertically stacking multiple chips. Computation down-shift refers to moving computational tasks from the main processor (such as the CPU) to dedicated computing units or accelerators within the memory subsystem for execution. Through computation down-shift, compute-intensive tasks can be processed in specialized hardware accelerators or computational units within the memory, alleviating the burden on the main processor and improving computational efficiency and energy efficiency. A typical product is computational storage devices (CSD), which incorporate computing units within or near the controller of solid-state drives (SSD) to enhance overall energy efficiency.

Emerging NVM also shows immense potential in data-intensive near-memory computing. Compared to traditional NVMs like Flash, emerging NVMs such as ReRAM, MRAM, PCRAM, and FeRAM provide better read/write performance, improved scalability, and distinct characteristics.

This section reviews representative achievements in data-intensive near-memory computing from both academia and industry in recent years. Table 2 presents several related examples.

4.1 DRAM NMC

DRAM has advanced storage density, performance, power efficiency, and reliability. By utilizing smaller process nodes and multi-layer stacking techniques, DRAM chips now provide higher storage capacities, making them ideal for applications like data centers that handle LLMs. Due to the mature manufacturing processes of DRAM, memory-intensive compute-in-memory technology based on DRAM has already found applications in specific commercial fields.

4.1.1 *HBM-PIM*

At the 2021 Hot Chips conference [22], Samsung proposed the Aquabolt-XL HBM2-PIM solution for machine learning accelerators. HBM2 employs 3D stacking of up to eight DRAM chips, each HBM2 stack consisting of eight DRAM chips (8-Hi). Each chip features two 64-bit channels, resulting in 16 pseudo-channels with a combined width of 1024 bits. This architecture offers a high memory bandwidth exceeding 256 GB/s. The Aquabolt-XL HBM2-PIM architecture integrates PIM technology within HBM2, achieving over a 2× improvement in system performance and reducing energy consumption by more than 70%.

Aquabolt-XL achieved an external PIM compute bandwidth of 1.23 TB/s and an internal PIM compute bandwidth of 4.92 TB/s on unmodified GPU and Xilinx FPGA.

At the 2023 Hot Chips conference [23], Samsung introduced the HBM-PIM cluster architecture based on PIM and PNM technologies. The HBM-PIM cluster demonstrated over 3× higher energy efficiency and more than 2× performance improvement compared to conventional GPU clusters. This architecture features 96 AMD MI100 GPUs integrated with HBM-PIM technology, designed to accelerate large-scale workloads, delivering 471.9 TFLOPS of PIM performance and 17.7 PFLOPS of FP16 GPU performance, with a memory capacity of 2.25 TB. The GPUs are interconnected via a 200G InfiniBand network, providing a bandwidth of 1.2 TB/s.

4.1.2 *GDDR6-AiM*

At the 2022 ISSCC conference [24], SK Hynix proposed the first accelerator memory based on PIM technology, GDDR6-AiM, designed to accelerate memory-intensive machine learning applications. GDDR6-AiM integrates 16 processing units per chip and is responsible for performing MAC and other deep-learning tasks. The chip employs an entire library base-width mantissa shifting (BWMS) scheme to handle floating-point operations efficiently, reducing computation time, power consumption, and circuit area. A dedicated instruction set is also introduced to minimize latency during operation mode transitions and efficiently support deep learning operations. During automated test equipment testing, GDDR6-AiM demonstrated up to 16 Gbps with a supply voltage of 1.10 V, lower than the standard GDDR6 voltage of 1.35 V. In FPGA-based system evaluations, GDDR6-AiM exhibited a 7.5–10.5-fold performance improvement in GEMV and MNIST applications compared to HBM2 or conventional GDDR6 systems, showcasing significant performance and energy efficiency advantages in memory-intensive machine learning applications.

4.1.3 *UPMEM PIM*

UPMEM [25] is the first company to integrate processors into DDR4 DRAM chips, achieving general-purpose in-memory computing and commercialization. UPMEM modules embed 8 PIM chips on each side, totaling 16 per module, with 128 DPUs in each DIMM module. These modules adopt the standard DDR4-2400 DIMM form factor, allowing easy insertion into standard servers. Standard DIMMs and UPMEM DIMMs can coexist, supporting both conventional and PIM processing. The additional cost of manufacturing PIM-DRAM chips is minimal compared to regular DRAM. UPMEM PIM can accelerate applications that utilize highly parallel operations and fine-grained partitioning, handle data volumes of tens or hundreds of gigabytes, exhibit a high compute/data ratio, and involve processing large datasets, implicit data movement, and executing critical operations on these applications.

DRAM-PIM technology integrates computational capabilities into DRAM chips, significantly enhancing performance and energy efficiency for data-intensive applications. Samsung's HBM-PIM and SK Hynix's GDDR6-AiM have substantially improved system performance and power efficiency, particularly in machine learning and large-scale workloads. UPMEM has pioneered the commercial use of general-purpose in-memory computing with DDR4 DRAM. With technology maturing, DRAM-PIM is expected to see widespread adoption in fields like AI and big data analytics, driving innovations in computing architecture.

4.2 NVM CIM/NMC

NVM is pivotal in enabling data-intensive computing. NVM CIM/NMC technology provides an efficient approach to near-memory computing by significantly increasing storage density, reducing power consumption, and improving read/write speeds. NAND Flash is the fundamental storage medium in SSD. It provides high-density, non-volatile storage that allows SSD to access large volumes of data rapidly and reliably, significantly reducing data transfer overhead between memory and remote processors. Numerous academic studies have underscored the promising future of emerging NVM.

This section will introduce representative products of data-intensive near-memory computing based on NVM.

4.2.1 *Samsung smartSSD*

Lee et al. [26] proposed the SmartSSD, an SSD equipped with an onboard FPGA, allowing for the offloading of computation directly within the SSD. The SmartSSD consists of an SSD controller, NAND array, FPGA accelerator, FPGA DRAM, and a PCIe switch. Utilizing the built-in FPGA supports high-speed computation close to where the data is stored, enhancing data processing speed and efficiency. A comparison between the SmartSSD approach and various other methods, including those utilizing external FPGAs, reveals that SmartSSD achieves a 3.04× speedup over traditional CPU-based techniques. In contrast, another method utilizing two external FPGAs demonstrates up to a 2.64× speedup.

Regarding commercial application, the first-generation SmartSSD, co-developed by Samsung and AMD in 2020 with a capacity of 3.84 TB, has been instrumental in supporting global IT service providers, including video communication platforms. Notably, it was recognized as a CES 2021 Innovation Awards Honoree for its exceptional performance and energy efficiency. In 2022, Samsung introduced the second-generation SmartSSD, which offers up to 97% CPU utilization, a 50% reduction in processing time, and a 70% decrease in power consumption compared to traditional Samsung SSDs. The second-generation SmartSSD utilizes Arm cores, custom-developed intellectual property (IP), and software, enabling more efficient data processing. It effectively integrates computation and storage functions in data centers, significantly enhancing CPU efficiency while dramatically reducing power consumption. With data-intensive applications proliferate, Samsung second-generation SmartSSD offers enhanced performance and efficiency, meeting the growing demands of this expanding market.

4.2.2 *StreamPIM: streaming matrix computation in racetrack memory*

An et al. proposed StreamPIM [27], a novel PIM architecture that integrates the memory core with computation units. StreamPIM directly constructs a matrix processor using domain-wall nanowires, eliminating the need for CMOS-based computation units. Additionally, it incorporates a domain-wall nanowire-based bus, which removes the requirement for electromagnetic conversion. StreamPIM enhances performance by exploiting internal parallelism. Evaluation results demonstrate that StreamPIM delivers a 39.1× increase in performance and a 58.4× reduction in energy consumption compared to conventional computing platforms.

4.2.3 *X-Former: in-memory acceleration of transformers*

Sridharan et al. [28] proposed X-Former, a hybrid in-memory hardware accelerator integrating NVM and CMOS processing elements to handle transformer workloads efficiently. To enhance hardware utilization of X-Former, they also introduced a sequence blocking dataflow that allows concurrent computations between the two processing elements, thereby reducing execution time. Through various benchmarks, X-Former demonstrated significant performance gains, achieving up to 69.8× and 13× reductions in latency and energy consumption compared to an NVIDIA GeForce GTX 1060 GPU and up to 24.1× and 7.95× improvements in latency and energy compared to a state-of-the-art in-memory NVM accelerator.

The high storage density, low power consumption, and high bandwidth of NVM technology enable faster data access and processing by the processor, accelerating computation and enhancing overall performance and energy efficiency. However, this technological approach presents drawbacks, including slow read/write speeds and limited write endurance. While it offers benefits in reducing data transfer energy consumption and improving data processing efficiency, these limitations must be considered in practical applications.

With the development of LLMs, the application scenarios for data-intensive operators are becoming increasingly widespread. Data-intensive in-memory computing is undoubtedly an essential approach to addressing this challenge. Numerous efforts have significantly improved the performance of data-intensive applications by enhancing memory bandwidth and processing capabilities, combined with optimizing the integration of storage and computation. The technology for data-intensive storage circuits has gradually matured. However, to further enhance the efficiency of large-scale data processing, additional efforts are required to optimize the overall circuit architecture and the algorithmic and software frameworks. These aspects will be discussed in detail in Section 5.

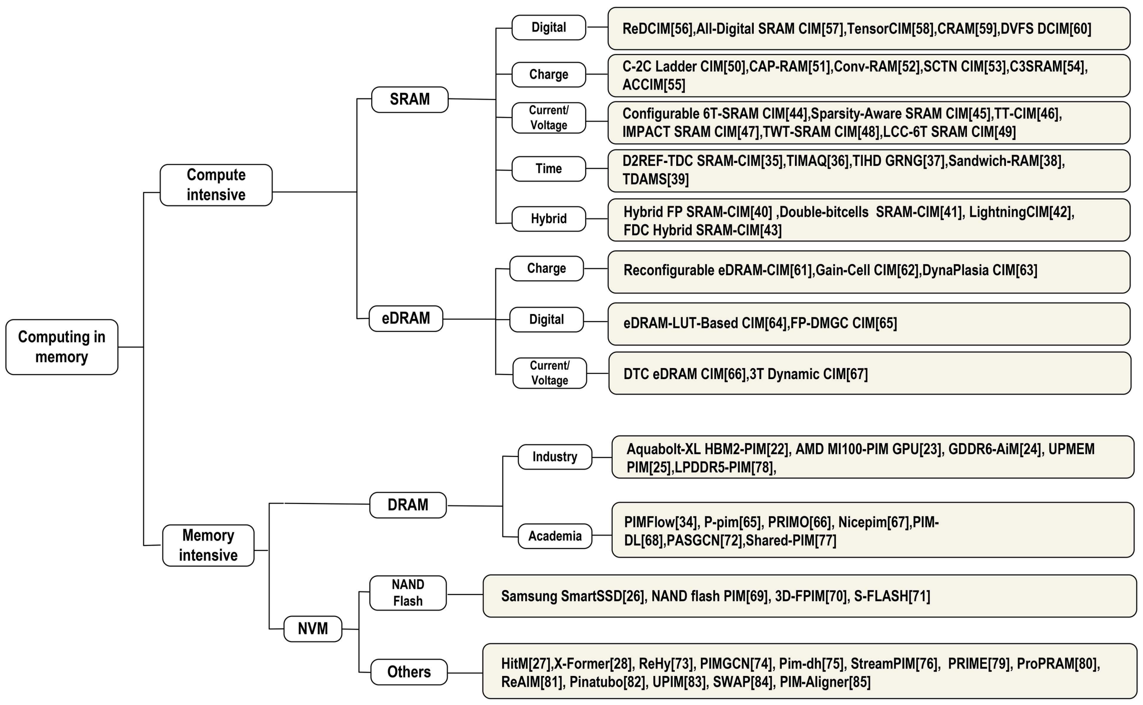

Computing in memory
Compute intensive
SRAM
Digital
ReDCIM[56],All-Digital SRAM CIM[57],TensorCIM[58],CRAM[59],DVFS DCIM[60]
Charge
C-2C Ladder CIM[50],CAP-RAM[51],Conv-RAM[52],SCTN CIM[53],C3SRAM[54], ACCIM[55]
Current/ Voltage
Configurable 6T-SRAM CIM[44],Sparsity-Aware SRAM CIM[45],TT-CIM[46], IMPACT SRAM CIM[47],TWT-SRAM CIM[48],LCC-6T SRAM CIM[49]
Time
D2REF-TDC SRAM-CIM[35],TIMAQ[36],TIHD GRNG[37],Sandwich-RAM[38], TDAMS[39]
Hybrid
Hybrid FP SRAM-CIM[40] ,Double-bitcells SRAM-CIM[41], LightningCIM[42], FDC Hybrid SRAM-CIM[43]
eDRAM
Charge
Reconfigurable eDRAM-CIM[61],Gain-Cell CIM[62],DynaPlasia CIM[63]
Digital
eDRAM-LUT-Based CIM[64],FP-DMGC CIM[65]
Current/ Voltage
DTC eDRAM CIM[66],3T Dynamic CIM[67]
Memory intensive
DRAM
Industry
Aquabolt-XL HBM2-PIM[22], AMD MI100-PIM GPU[23], GDDR6-AiM[24], UPMEM PIM[25],LPDDR5-PIM[78],
Academia
PIMFlow[34], P-pim[65], PRIMO[66], Nicepim[67],PIM-DL[68],PASGCN[72],Shared-PIM[77]
NVM
NAND Flash
Samsung SmartSSD[26], NAND flash PIM[69], 3D-FPIM[70], S-FLASH[71]
Others
HitM[27],X-Former[28], ReHy[73], PIMGCN[74], Pim-dh[75], StreamPIM[76], PRIME[79], ProPRAM[80], ReAIM[81], Pinatubo[82], UPIM[83], SWAP[84], PIM-Aligner[85]

Figure 4 Taxonomy of compute-intensive CIM and memory-intensive CIM.

5 Challenges and development trends of CIM/NMC in large language model

In the era of LLMs, the rapid progress in AI and deep learning has exposed the computational and bandwidth limitations of traditional architectures [29, 30], especially in addressing compute-intensive and data-intensive tasks. To overcome these challenges, compute-intensive CIM and data-intensive CIM/NMC have emerged as critical approaches for accelerating LLMs [31–33]. The former enhances efficiency in tasks with high data reuse and computational demands, while the latter focuses on optimizing storage and processing in scenarios limited by bandwidth and data intensity. Figure 4 [22–28, 34–85] summarizes numerous existing studies on compute-intensive and memory-intensive CIM architectures. Current research is actively exploring these two computing paradigms. With the development of LLMs, the efficient implementation and integration of both compute-intensive CIM and memory-intensive CIM will be crucial for enhancing the performance and energy efficiency of future high-performance computing systems.

5.1 Compute-intensive CIM

In compute-intensive CIM design, maximizing energy efficiency and computational power is crucial, particularly for operators with high reusability and computational density in LLMs. Many recent studies have focused on optimizing energy efficiency while balancing other design aspects. Moreover, a complete AI computing deployment requires more than just macro cell design; it also demands an optimized memory hierarchy, data flow, and instruction set design. A rational approach to data flow and control is essential. The use of advanced materials could further enhance energy efficiency and computational power. This section focuses on the design trade-offs in compute-intensive CIM and explores future development trends.

5.1.1 *Energy efficiency*

Compute-intensive CIM confronts a tradeoff between energy efficiency, throughput, and computational accuracy. While high reuse and parallel computation boost performance in compute-intensive tasks, throughput often comes at the expense of energy efficiency. Analog computation employs digital-to-analog converters (DACs) for simultaneous multi-bit input, shortens the computation cycle, and improves throughput while significantly enhancing energy efficiency. However, accuracy errors introduced by ana-

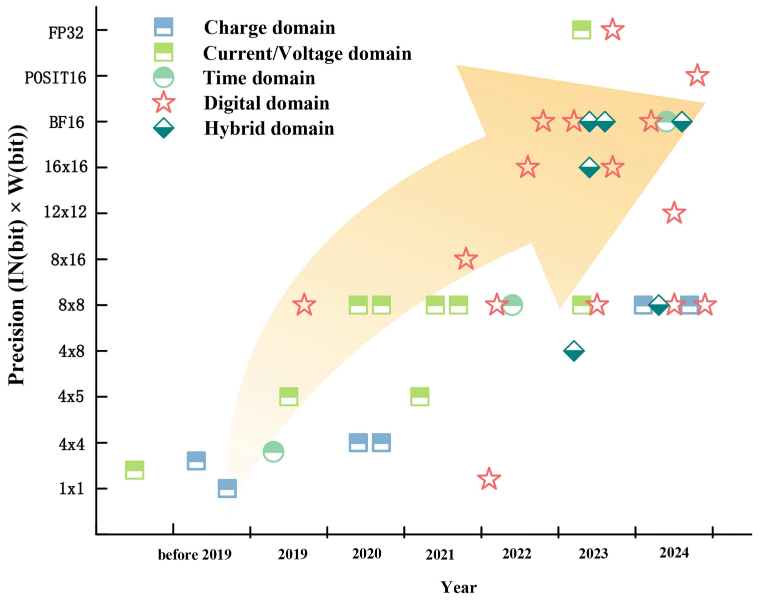


Figure 5 Compute-intensive CIM precision development trends.

log computation must be addressed, making the balance between energy efficiency and computational accuracy a key focus for future CIM research.

A promising approach to improving energy efficiency is hybrid CIM, which combines digital and analog domains. Hybrid CIM ensures accuracy in the digital domain while utilizing the analog domain for greater area and energy efficiency. Current hybrid methods include product decomposition, where lower bits of multi-bit multiplication are computed in the analog domain and higher bits in the digital domain, and decomposition of calculations, assigning analog computation to less significant portions and digital computation to critical parts.

Low power technology is also very helpful for improving energy efficiency [86–88]. Additionally, utilizing network characteristics such as sparsity [89] offers a promising avenue for circuit optimization and energy efficiency improvement. In analog computation, sparsity affects quantization circuit utilization, while in digital circuits, it influences the switching rate. Despite efforts to integrate software and hardware to exploit network characteristics, effectively harnessing circuit properties arising from sparsity to reduce energy consumption remains a significant challenge.

5.1.2 *Operator types*

Compute-intensive CIM supports only specific layers in certain networks, leaving other layers reliant on traditional digital circuits, limiting system-level energy efficiency. As network complexity increases and more operators are required, advancing CIM will depend on expanding its support for a broader range of operators. This would reduce data transfer between CIM macros, external memory, and logic circuits, improving overall generalization.

For more complex operators, the precision demands for CIM are continuously increasing. Figure 5 shows the compute-intensive CIM precision development trends in recent years. Some networks can operate at 8-bit fixed-point precision through quantization during inference, but floating-point operations are essential for training and specific inference tasks. The challenge with floating-point CIM is the tight coupling between exponent alignment and integer mantissa MAC (multiply-accumulate). Implementing re-exponent alignment in memory disrupts the direct accumulation structure of CIM, reducing efficiency. The critical issue is how the exponent is managed during computation.

As shown in Figure 6, there are three main approaches for implementing floating-point operations in CIM. The first method pre-aligns inputs and weights to their local maximum exponents before computation, resembling a global floating-point and local fixed-point approach. This method has minimal circuit overhead but suffers from significant precision loss. The second method calculates the exponents of inputs and weights and then shifts the inputs before multiplication. This approach also has low circuit overhead with comparatively less precision loss. The third method adheres to the standard floating-point

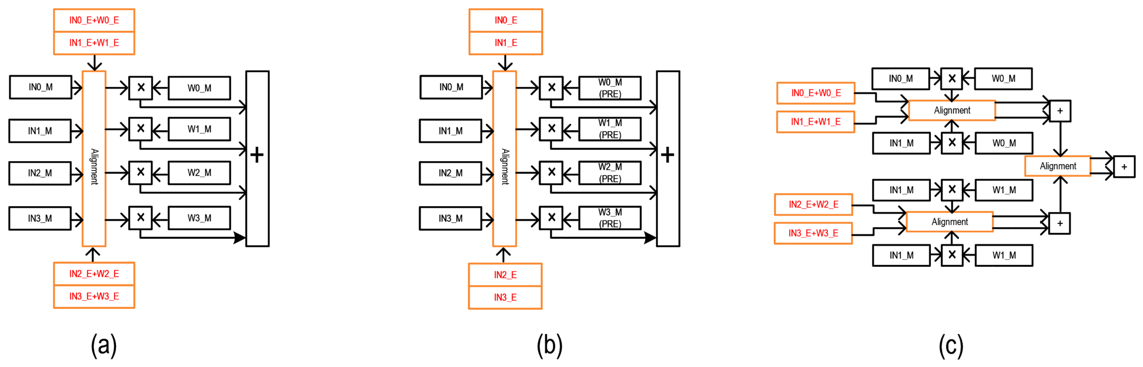


Figure 6 (Color online) Three methods for implementing floating-point operation shift in CIM. (a) Only input alignment: aligning input data before performing floating-point operations; (b) sharefloat: utilizing a shared floating-point format to optimize computational efficiency; (c) product alignment (normal): aligning the product during computation to ensure accuracy.

process, including shift calculations, which increases area and energy consumption but offers much higher precision. Achieving greater energy and area efficiency is essential for compute-intensive CIM, but the appropriate precision must be selected based on task requirements. Further research is necessary to develop better floating-point solutions for in-memory computing.

Supporting multiple operators and operator fusion in compute-intensive CIM is a significant challenge. Multiple operators are essential for handling complex AI scenarios, while operator fusion enhances efficiency by performing more operations on data during each read. However, most current architectures are limited to MAC or simple logical operations and need more flexibility of digital architectures to support configurable operators and fusion. Achieving this flexibility is crucial for improving overall system energy efficiency.

5.1.3 *System performance*

In compute-intensive CIM/NMC, system performance is heavily influenced by architectural design. Beyond the CIM macro, a complete AI acceleration core requires memory hierarchy, data flow, and instruction design support. Performance hinges on the design of the external memory hierarchy and overall data flow. More significant external memory facilitates data reuse and reduces energy for data updates, enhancing performance, though it increases area overhead.

Optimizing system efficiency involves selecting data flow methods tailored to the operational characteristics and data features of different network operators. Lightweight neural network architectures are relatively simple and have fewer parameters [90]. The computational bottleneck in these networks lies in how to efficiently handle input data, emphasizing operations such as data shifting and transmission. In contrast, deep convolutional neural networks have a larger depth and more parameters [91]. For hardware, the focus is on reducing the scheduling of weight data or efficiently updating weights. In multi-reuse convolution operations, the choice of computation dimension affects system design and power consumption. Channel-based accumulation operations allow data accumulation but demand more input data, straining input bandwidth. Kernel-based operations reuse input data but incur storage overhead for partial sums. Effective data flow scheduling is crucial for multi-task scheduling in multi-operator operation reconstruction, ensuring high computational power and energy efficiency in data-intensive tasks. Additionally, in-memory computing requires well-aligned software and hardware interfaces to maximize efficiency.

5.1.4 *Advanced material*

For compute-intensive CIM, the predominant technologies are SRAM-CIM and eDRAM-CIM, owing to their mature storage media, which offer high durability, low write energy consumption, and fast read/write speeds. However, these media also suffer from data loss upon power failure and low storage density. Though less efficient in read/write time and energy, emerging storage media like PCM, RRAM, and MRAM provide higher storage density and non-volatility, making them better suited for high-density computing [92]. Figure 7 shows the comparison of various features of different memory technologies.

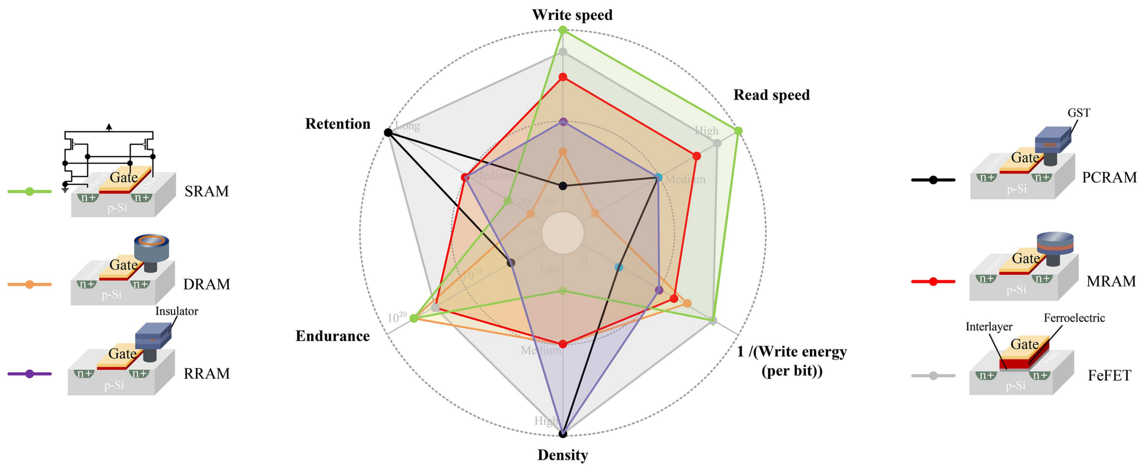


Figure 7 Key feature comparison of different memory technologies.

The high storage density and low power consumption of ReRAM reduce data transfer bottlenecks, enhancing computational efficiency. With its phase-change properties, PCM enables high-speed access and persistence, ideal for rapid data access and large-scale storage. The 3D XPoint technology based on PCM is already used as non-volatile memory, offering storage density several times higher than DRAM. MRAM, utilizing magnetic materials, combines fast read/write speeds with high durability and low power consumption, making it suitable for frequent data access scenarios. FeFET also supports efficient data access with low power consumption, while STT-MRAM, leveraging spin-transfer torque, offers high speed and durability for high-frequency data access applications.

Selecting the appropriate storage media depends on the specific AI application. Emerging storage media are less favorable for training, which involves frequent weight updates, due to their limited durability and expensive write operations. In contrast, inference operations are mostly read-based, allowing new storage media to store more data and enhance computational power. Given the complexity of processing large AI models, loading the entire model into memory at once is impractical, so data are processed layer by layer. As volatile memory, SRAM avoids the memory management challenges of non-volatile media due to its unlimited write endurance. However, for inference, where data retention upon power loss is critical, new storage media like PCM and RRAM offer more promise than SRAM and DRAM.

5.2 Memory-intensive CIM/NMC

Memory-intensive computing is primarily near-memory computing, characterized by high storage density and sufficient bandwidth, making it highly compatible with data-intensive operators. In previous neural networks, most data-intensive operators, such as activation functions, were optimized through operator fusion, which reduced the demand for high storage density computing. However, with the explosive growth of LLMs, the decoder section has introduced significant data-intensive computing requirements. For data-intensive in-memory computing, architectural design and storage medium technology breakthroughs are crucial for development. This section analyzes the architectural-level challenges of data-intensive in-memory computing and provides an outlook on future development trends.

5.2.1 *Circuit architecture application level*

Numerous memory-intensive CIM circuits have reached a level of maturity in both design and fabrication processes. Nonetheless, the effective integration of such data-intensive methodologies into existing computer architectures while maintaining high performance continues to present substantial challenges. This issue has garnered attention within the academic community, with several studies investigating potential solutions. As shown in Figure 8(a), Zhao et al. [93] explored the design of DRAM-based PIM systems with a unified memory space alongside the CPU. Typically, CPUs optimize memory bandwidth by reading and writing data across different banks to exploit the increased bandwidth offered by multiple banks.

In contrast, PIM relies heavily on data centralization to enhance computational efficiency. However, if separate DRAM spaces are employed to satisfy the distinct requirements of the CPU and PIM, data

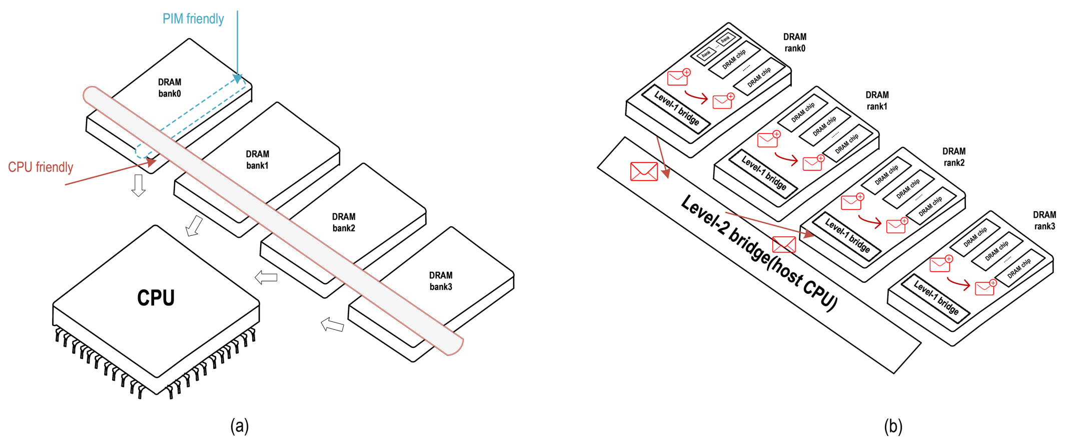


Figure 8 (Color online) Two studies for memory-intensive methodologies into existing computer architectures. (a) Conflict between CPU data read/write operations and the utilization of data in PIM; (b) DRAM-based PIM system to address the challenges associated with cross-bank data interactions.

transfers between these spaces would still be necessary, thereby undermining the primary objective of PIM. Consequently, developing a unified memory space that can accommodate both divergent demands is paramount. Moreover, as mentioned in Figure 8(b), Tian et al. [94] have investigated optimizations to DRAM-based PIM systems to address the challenges associated with cross-bank data interactions. These studies contribute substantially to the advancement of knowledge regarding the effective integration of storage-intensive in-memory computing into modern computer architectures and the management of complex applications within these systems. Further architectural research is imperative to identify optimal solutions for managing data and control flow within storage-intensive in-memory computing circuits.

5.2.2 *Dedicated software architecture*

To effectively leverage computational memory, several unique challenges and opportunities emerge for software architecture, particularly in mapping computational demands to the execution units within memory. Realizing the potential of memory-integrated computing capabilities requires adaptive adjustments and optimizations in critical areas such as parallelization, algorithm design, memory management, and programming models. Traditional centralized computing methods must be better suited for harnessing the parallel processing capabilities of PIM systems. Therefore, developing innovative parallel processing strategies that facilitate the equitable distribution of computational tasks across the diverse memory-resident processing units is imperative.

Moreover, traditional algorithms are generally designed under the assumption of the von Neumann architecture, where computing and storage resources are separate. In contrast, PIM architectures integrate computing and storage resources, opening up new opportunities for algorithm optimization. To fully capitalize on the performance benefits of PIM, it is essential to optimize algorithms to minimize data movement and enable the execution of most computational tasks directly within the memory itself.

Regarding programming models, new frameworks and tools are necessary to simplify the utilization of PIM hardware for developers. Existing artificial intelligence frameworks, such as Pytorch and Tensorflow, must be extended to support PIM functionalities, including providing runtime libraries that optimize task execution during PIM operations. Developing specialized PIM compilers can also enable graph-level optimizations during end-to-end execution, effectively mapping complex computational demands to the memory's processing units.

Standardization plays a critical role in the future development of PIM technology. As this technology matures, establishing unified standards and interfaces and developing a shared software ecosystem across different hardware platforms will be essential for promoting the widespread adoption of PIM. Figure 9 proposes an improved CIM-based software and hardware architecture based on traditional architecture. The current technological landscape remains dominated by traditional computing models, with a significant gap in comprehensive software ecosystem support for PIM. Therefore, advancing the standardization of PIM technology is imperative for enabling developers to fully exploit this innovative memory architecture, ultimately leading to more efficient computing.

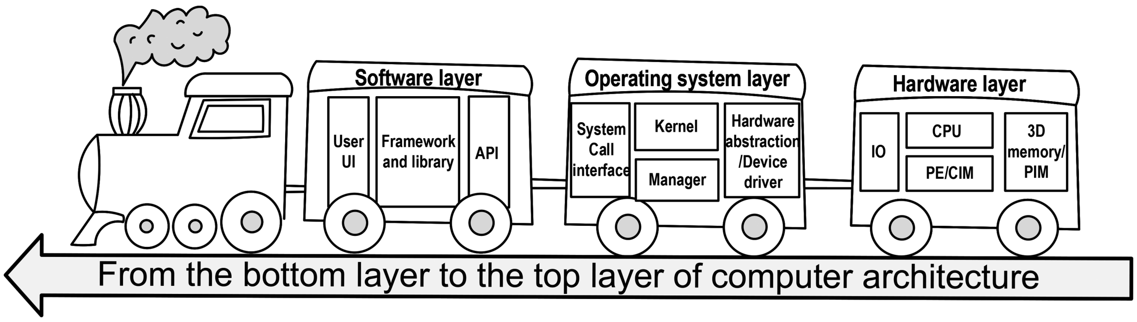


Figure 9 Computer architecture from CIM AI core to dedicated software.

5.3 Development trend of CIM/NMC in LLMs

The rapid development of LLMs has substantially increased the demand for computational power, presenting significant challenges in data-intensive and compute-intensive application scenarios. This has led to the necessity for in-memory computing chips that deliver higher energy efficiency and greater versatility. In-memory computing solutions currently demonstrate substantial efficiency advantages in these scenarios. However, it is crucial to consider how to enhance computational efficiency from the kernel level to the system level and improve chip performance through coordinated design across upstream and downstream processes for LLM applications. This section focuses on analyzing in-memory computing in the context of LLMs, with particular emphasis on heterogeneous architecture and heterogeneous integration, while also exploring future development trends.

The heterogeneous architecture integrates diverse computing architectures and hardware units with distinct functionalities, allowing the system to leverage the strengths and compensate for the weaknesses of each component to achieve enhanced overall performance. Compute-intensive CIM offers notable precision and energy efficiency advantages, whereas memory-intensive CIM excels in storage density and bandwidth.

For current LLM applications, relying on a single architecture may only adequately satisfy some performance requirements. A heterogeneous architecture can facilitate a balanced trade-off among precision, energy efficiency, and area metrics. Several studies have already explored heterogeneous integration in hybrid architectures, demonstrating its potential for LLM. Cambricon-LLM [95] introduces a hybrid architecture based on chip-let technology, integrating high-performance NPUs with NAND flash chips that offer limited in-die computation capabilities to achieve efficient hardware resource utilization for large model acceleration. Cambricon-LLM demonstrates 22× to 45× speedups over existing flash offloading techniques, showcasing its potential for deploying powerful LLMs on edge devices. NeuPIMs [96] address the challenge of accelerating both memory-intensive GEMV and compute-intensive GEMM operators in LLM, which traditional NPU and PIM methods struggle to handle simultaneously. Compared to GPU-only, NPU-only, and a naïve NPU+PIM integrated acceleration approach, NeuPIMs achieve 3×, 2.4×, and 1.6× throughput improvements, respectively. SpecPIM [97] proposes a heterogeneous accelerator architecture with software-hardware co-design based on a near-DRAM computing architecture to address the difficulty of simultaneously accelerating compute-intensive and memory-intensive operators in large models. Compared to speculative inference on GPUs and existing PIM-based LLM accelerators, SpecPIM achieves 1.52×/2.02× geomean speedup and 6.67×/2.68× geomean higher energy efficiency. Additionally, the implementation of heterogeneous architecture necessitates support from heterogeneous integration technologies. With advancements in advanced packaging technologies, CIM chips integrated with 2.5D/3D/Chiplet packaging techniques can achieve superior performance. Furthermore, it is imperative to ensure the compatibility of new memory device processes with advanced integrated packaging processes to enable the integration of in-memory computing chips across different media, thereby fully harnessing the strengths of various media in multiple aspects.

The relationship among different memory-based CIM solutions is more complementary than competitive. These solutions possess distinct characteristics and realize functions with corresponding features at different levels within the memory hierarchy. Different CIM solutions can be employed at various levels of the memory hierarchy, akin to the harmony achieved by SRAM, DRAM, and Flash in the evolution of modern computing. By coordinating task allocation during computation, general tasks with lower reuse and computational intensity can be managed by traditional architectures. In contrast, tasks characterized by high data reuse and computational intensity can be assigned to compute-intensive CIM. Large-scale data access and processing tasks, which are data-intensive, can be delegated to in-memory computing units, thereby achieving a balance and optimizing overall performance. The heterogeneous integration of compute-intensive and memory-intensive CIM architectures within traditional architectures for LLM computing applications represents a promising direction for future development.

6 Conclusion

The advent of the era of large models has imposed new demands for high computational power and throughput on existing AI chips. CIM architecture is one of the potential solutions to address current chip bottlenecks effectively. This study redefines the traditional computer architecture pyramid based on storage media, creating the CIM pyramid. The study categorizes current CIM works into two types: compute-intensive CIM, which is primarily based on SRAM and eDRAM, and memory-intensive CIM, which is mainly based on DRAM and NVM. This research summarizes the related studies of these two types of in-memory computing, analyzing and discussing the development trends and challenges faced by both compute-intensive and memory-intensive CIM. Additionally, the study proposes a hybrid heterogeneous future development trend that integrates compute-intensive CIM, memory-intensive CIM, and traditional computer architectures to meet the demands of both types of operators in the era of large models.

Acknowledgements This work was supported by National Key R&D Program of China (Grant Nos. 2020YFA0711900, 2020YFA0711902, 2022ZD0118901), National Natural Science Foundation of China (Grant Nos. 92264203, 62204036), Key R&D Program of Jiangsu Province (Grant No. BE2023020-1), and Fundamental Research Funds for the Central Universities (Grant No. 2242022k60009).

References

1 Gholami A, Yao Z, Kim S, et al. AI and memory wall. IEEE Micro, 2024, 44: 33–39
2 Jhang C J, Xue C X, Hung J M, et al. Challenges and trends of SRAM-based computing-in-memory for AI edge devices. IEEE Trans Circuits Syst I, 2021, 68: 1773–1786
3 Su Y, Kim H, Kim B. CIM-Spin: a scalable CMOS annealing processor with digital in-memory spin operators and register spins for combinatorial optimization problems. IEEE J Solid-State Circuits, 2022, 57: 2263–2273
4 Yue Z, Wang Y, Wang H, et al. CV-CIM: a hybrid domain xor-derived similarity-aware computation-in-memory supporting cost-volume construction. IEEE J Solid-State Circuits, 2025, 60: 719–733
5 Bankman D, Yang L, Moons B, et al. An always-on 3.8 μJ/86% CIFAR-10 mixed-signal binary CNN processor with All memory on chip in 28-nm CMOS. IEEE J Solid-State Circuits, 2019, 54: 158–172
6 Biswas A, Chandrakasan A P. Conv-RAM: an energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2018. 488–490
7 Zhang J, Wang Z, Verma N. In-memory computation of a machine-learning classifier in a standard 6T SRAM array. IEEE J Solid-State Circuits, 2017, 52: 915–924
8 Ling Y T, Wang Z W, Yang Y H, et al. An isolated symmetrical 2T2R cell enabling high precision and high density for RRAM-based in-memory computing. Sci China Inf Sci, 2024, 67: 152402
9 Zhang Z Y, Chen J W, Chen X, et al. From macro to microarchitecture: reviews and trends of SRAM-based compute-in-memory circuits. Sci China Inf Sci, 2023, 66: 200403
10 Liang B-S. AI computing in large-scale era: pre-trillion-scale neural network models and exa-scale supercomputing. In: Proceedings of International VLSI Symposium on Technology, Systems and Applications (VLSI-TSA/VLSI-DAT), Taiwan, 2023. 1–3
11 Wolters C, Yang X, Schlichtmann U, et al. Memory is all you need: an overview of compute-in-memory architectures for accelerating large language model inference. 2024. ArXiv:2406.08413
12 Williams S, Waterman A, Patterson D. Roofline: an insightful visual performance model for multicore architectures. Commun ACM, 2009, 52: 65–76
13 Kim S, Hooper C, Wattanawong T, et al. Full stack optimization of transformer inference: a survey. 2023. ArXiv:2302.14017
14 Tu F, Wu Z, Wang Y, et al. TranCIM: full-digital bitline-transpose CIM-based sparse transformer accelerator with pipeline/parallel reconfigurable modes. IEEE J Solid-State Circuits, 2023, 58: 1798–1809
15 Guo A, Si X, Chen X, et al. A 28nm 64-kb 31.6-TFLOPS/W digital-domain floating-point-computing-unit and double-bit 6T-SRAM computing-in-memory macro for floating-point CNNs. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2023. 128–130
16 Yoshioka K. 34.5 A 818-4094TOPS/W capacitor-reconfigured CIM macro for unified acceleration of CNNs and transformers. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2024. 574–576
17 Xie S, Ni C, Sayal A, et al. eDRAM-CIM: compute-in-memory design with reconfigurable embedded-dynamic-memory array realizing adaptive data converters and charge-domain computing. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2021. 248–250

18 Song J, Tang X, Luo H, et al. A calibration-free 15-level/cell eDRAM computing-in-memory macro with 3T1C current-programmed dynamic-cascoded MLC achieving 233-to-304-TOPS/W 4b MAC. In: Proceedings of IEEE Custom Integrated Circuits Conference (CICC), San Antonio, 2023. 1–2
19 He Y, Fan S, Li X, et al. A 28 nm 2.4 Mb/mm^2 6.9–16.3 TOPS/mm^2 eDRAM-LUT-based digital-computing-in-memory macro with in-memory encoding and refreshing. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2024. 578–580
20 Park S O, Hong S, Sung S J, et al. Phase-change memory via a phase-changeable self-confined nano-filament. Nature, 2024, 628: 293–298
21 Ikegawa S, Mancoff F B, Janesky J, et al. Magnetoresistive random access memory: present and future. IEEE Trans Electron Devices, 2020, 67: 1407–1419
22 Kim J H, Kang S, Lee S, et al. Aquabolt-XL: Samsung HBM2-PIM with in-memory processing for ML accelerators and beyond. In: Proceedings of IEEE Hot Chips 33 Symposium (HCS), 2021. 1–26
23 Kim J H, Ro Y, So J, et al. Samsung PIM/PNM for transformer based AI: energy efficiency on PIM/PNM cluster. In: Proceedings of IEEE Hot Chips 35 Symposium (HCS), 2023. 1–31
24 Kwon Y, Vladimir V K, Kim N, et al. System architecture and software stack for GDDR6-AiM. In: Proceedings of IEEE Hot Chips 34 Symposium (HCS), 2022. 1–25
25 Gómez-Luna J, Hajj I E, Fernandez I, et al. Benchmarking memory-centric computing systems: analysis of real processing-in-memory hardware. In: Proceedings of IEEE 12th International Green and Sustainable Computing Conference (IGSC), 2021. 1–7
26 Lee J H, Zhang H, Lagrange V, et al. SmartSSD: FPGA accelerated near-storage data analytics on SSD. IEEE Comput Arch Lett, 2020, 19: 110–113
27 An Y, Tang Y, Yi S, et al. StreamPIM: streaming matrix computation in racetrack memory. In: Proceedings of IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2024. 297–311
28 Sridharan S, Stevens J R, Roy K, et al. X-Former: in-memory acceleration of transformers. IEEE Trans VLSI Syst, 2023, 31: 1223–1233
29 Li W J, Lyu D X, Wang G, et al. Hardware-oriented algorithms for softmax and layer normalization of large language models. Sci China Inf Sci, 2024, 67: 200404
30 Liu Y F, Li X Y, Yin S Y. Review of chiplet-based design: system architecture and interconnection. Sci China Inf Sci, 2024, 67: 200401
31 Choi J, Park J, Kyung K, et al. Unleashing the potential of PIM: accelerating large batched inference of transformer-based generative models. IEEE Comput Arch Lett, 2023, 22: 113–116
32 Chen L, Chen Y Q, Chu Z F, et al. Large circuit models: opportunities and challenges. Sci China Inf Sci, 2024, 67: 200402
33 Han S H, Liu S S, Du S C, et al. CMN: a co-designed neural architecture search for efficient computing-in-memory-based mixture-of-experts. Sci China Inf Sci, 2024, 67: 200405
34 Shin Y, Park J, Cho S, et al. PIMFlow: compiler and runtime support for CNN models on processing-in-memory DRAM. In: Proceedings of the 21st ACM/IEEE International Symposium on Code Generation and Optimization, 2023. 249–262
35 Wu P-C, Su J-W, Chung Y-L, et al. A 28 nm 1 Mb time-domain computing-in-memory 6T-SRAM macro with a 6.6 ns latency, 1241 GOPS and 37.01 TOPS/W for 8b-MAC operations for edge-AI devices. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2022. 1–3
36 Yang J, Kong Y, Zhang Z, et al. TIMAQ: a time-domain computing-in-memory-based processor using predictable decomposed convolution for arbitrary quantized DNNs. IEEE J Solid-State Circuits, 2021, 56: 3021–3038
37 Dorrance R, Dasalukunte D, Wang H, et al. An energy-efficient Bayesian neural network accelerator with CiM and a time-interleaved Hadamard digital GRNG using 22-nm FinFET. IEEE J Solid-State Circuits, 2023, 58: 2826–2838
38 Yang J, Kong Y, Wang Z, et al. Sandwich-RAM: an energy-efficient in-memory BWN architecture with pulse-width modulation. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2019. 394–396
39 Miyashita D, Kousai S, Suzuki T, et al. A neuromorphic chip optimized for deep learning and CMOS technology with time-domain analog and digital mixed-signal processing. IEEE J Solid-State Circuits, 2017, 52: 2679–2689
40 Wu P-C, Su J-W, Hong L-Y, et al. A 22 nm 832 Kb hybrid-domain floating-point SRAM in-memory-compute macro with 16.2–70.2 TFLOPS/W for high-accuracy AI-edge devices. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2023. 126–128
41 Guo A, Xi C, Dong F, et al. A 28-nm 64-kb 31.6-TFLOPS/W digital-domain floating-point-computing-unit and double-bit 6T-SRAM computing-in-memory macro for floating-point CNNs. IEEE J Solid-State Circuits, 2024, 59: 3032–3044
42 Guo A, Chen X, Dong F, et al. A 22 nm 64 kb lightning-like hybrid computing-in-memory macro with a compressed adder tree and analog-storage quantizers for transformer and CNNs. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2024. 570–572
43 Jeong S, Oh J, Jeon D. A 28 nm 157TOPS/W 446.9 Kb/mm^2 compute-in-memory SRAM macro with analog-digital hybrid computing for deep neural network inference. In: Proceedings of IEEE Custom Integrated Circuits Conference (CICC), Denver, 2024. 1–2
44 Chiu Y C, Zhang Z, Chen J J, et al. A 4-Kb 1-to-8-bit configurable 6T SRAM-based computation-in-memory unit-macro for CNN-based AI edge processors. IEEE J Solid-State Circuits, 2020, 55: 2790–2801
45 Ali M, Chakraborty I, Saxena U, et al. A 35.5–127.2 TOPS/W dynamic sparsity-aware reconfigurable-precision compute-in-memory SRAM macro for machine learning. IEEE Solid-State Circuits Lett, 2021, 4: 129–132
46 Guo R, Yue Z, Si X, et al. A 5.99-to-691.1 TOPS/W tensor-train in-memory-computing processor using bit-level-sparsity-based optimization and variable-precision quantization. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), 2021. 242–244
47 Kneip A, Lefebvre M, Verecken J, et al. IMPACT: a 1-to-4b 813-TOPS/W 22-nm FD-SOI compute-in-memory CNN accelerator featuring a 4.2-POPS/W 146-TOPS/mm^2 CIM-SRAM with multi-bit analog batch-normalization. IEEE J Solid-State

Circuits, 2023, 58: 1871–1884

48 Su J W, Si X, Chou Y C, et al. Two-way transpose multibit 6T SRAM computing-in-memory macro for inference-training AI edge chips. IEEE J Solid-State Circuits, 2022, 57: 609–624

49 Si X, Tu Y N, Huang W H, et al. A local computing cell and 6T SRAM-based computing-in-memory macro with 8-b MAC operation for edge AI chips. IEEE J Solid-State Circuits, 2021, 56: 2817–2831

50 Wang H, Liu R, Dorrance R, et al. 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 J Solid-State Circuits, 2023, 58: 1037–1050

51 Chen Z, Yu Z, Jin Q, et al. CAP-RAM: a charge-domain in-memory computing 6T-SRAM for accurate and precision-programmable CNN inference. IEEE J Solid-State Circuits, 2021, 56: 1924–1935

52 Biswas A, Chandrakasan A P. CONV-SRAM: an energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks. IEEE J Solid-State Circuits, 2019, 54: 217–230

53 Yang X, Zhu K, Tang X, et al. An in-memory-computing charge-domain ternary CNN classifier. In: Proceedings of IEEE Custom Integrated Circuits Conference (CICC), Austin, 2021. 1–2

54 Jiang Z, Yin S, Seo J S, et al. C3SRAM: an in-memory-computing SRAM macro based on robust capacitive coupling computing mechanism. IEEE J Solid-State Circuits, 2020, 55: 1888–1897

55 Zhang Z, Liu Z, Liu F, et al. A 28 nm 16 kb aggregation and combination computing-in-memory macro with dual-level sparsity modulation and sparse-tracking ADCs for GCNs. In: Proceedings of IEEE Custom Integrated Circuits Conference (CICC), Denver, 2024. 1–2

56 Tu F, Wang Y, Wu Z, et al. ReDCIM: reconfigurable digital computing- in-memory processor with unified FP/INT pipeline for cloud AI acceleration. IEEE J Solid-State Circuits, 2023, 58: 243–255

57 Chih Y D, Lee P H, Fujiwara H, et al. An 89 TOPS/W and 16.3 TOPS/mm^2 all-digital SRAM-based full-precision compute-in-memory macro in 22 nm for machine-learning edge applications. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2021. 252–254

58 Tu F, Wang Y, Wu Z, et al. TensorCIM: a 28 nm 3.7 nJ/Gather and 8.3 TFLOPS/W FP32 digital-CIM tensor processor for MCM-CIM-based beyond-NN acceleration. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2023. 254–256

59 Wang J, Wang X, Eckert C, et al. A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing. IEEE J Solid-State Circuits, 2020, 55: 76–86

60 Fujiwara H, Mori H, Zhao W-C, et al. A 5-nm 254-TOPS/W 221-TOPS/mm^2 fully-digital computing-in-memory macro supporting wide-range dynamic-voltage-frequency scaling and simultaneous MAC and write operations. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2022. 1–3

61 Xie S, Ni C, Jain P, et al. Gain-cell CIM: leakage and bitline swing aware 2T1C gain-cell eDRAM compute in memory design with bitline precharge DACs and compact Schmitt trigger ADCs. In: Proceedings of IEEE Symposium on VLSI Technology and Circuits, Honolulu, 2022. 112–113

62 Kim S, Li Z, Um S, et al. DynaPlasia: an eDRAM in-memory-computing-based reconfigurable spatial accelerator with triple-mode cell for dynamic resource switching. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2023. 256–258

63 Khwa W S, Wu P C, Wu J J, et al. A 16 nm 96 Kb integer/floating-point dual-mode-gain-cell-computing-in-memory macro achieving 73.3–163.3 TOPS/W and 33.2–91.2 TFLOPS/W for AI-edge devices. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2024. 568–570

64 Chen Z, Chen X, Gu J. A 65 nm 3T dynamic analog RAM-based computing-in-memory macro and CNN accelerator with retention enhancement, adaptive analog sparsity and 44 TOPS/W system energy efficiency. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2021. 240–242

65 Zhou R, Tabrizchi S, Morsali M, et al. P-PIM: a parallel processing-in-DRAM framework enabling row hammer protection. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), 2023. 1–6

66 Heo J, Shin Y, Choi S, et al. PRIMO: a full-stack processing-in-DRAM emulation framework for machine learning workloads. In: Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2023. 1–9

67 Wang J, Ge M, Ding B, et al. NicePIM: design space exploration for processing-in-memory DNN accelerators with 3-D stacked-DRAM. IEEE Trans Comput-Aided Des Integr Circuits Syst, 2024, 43: 1456–1469

68 Li C, Zhou Z, Wang Y, et al. PIM-DL: expanding the applicability of commodity DRAM-PIMs for deep learning via algorithm-system co-optimization. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 2024. 879–896

69 Kim H, Lee H, Kim J, et al. Cache register sharing structure for channel-level near-memory processing in NAND flash memory. In: Proceedings of the 24th International Symposium on Quality Electronic Design (ISQED), 2023. 1–6

70 Lee H, Kim M, Min D, et al. 3D-FPIM: an extreme energy-efficient DNN acceleration system using 3D NAND flash-based in-situ PIM unit. In: Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture (MICRO), 2022. 1359–1376

71 Kang M, Kim H, Shin H, et al. S-FLASH: a NAND flash-based deep neural network accelerator exploiting bit-level sparsity. IEEE Trans Comput, 2021, 71: 1291–1304

72 Yang T, Li D, Ma F, et al. PASGCN: an ReRAM-based PIM design for GCN with adaptively sparsified graphs. IEEE Trans Comput-Aided Des Integr Circuits Syst, 2023, 42: 150–163

73 Li B, Wang Y, Chen Y. HitM: high-throughput ReRAM-based PIM for multi-modal neural networks. In: Proceedings of the 39th International Conference on Computer-Aided Design, 2020. 1–7

74 Jin H, Liu C, Liu H, et al. ReHy: a ReRAM-based digital/analog hybrid PIM architecture for accelerating CNN training. IEEE Trans Parallel Distrib Syst, 2021, 33: 2872–2884

75 Yang T, Li D, Han Y, et al. PIMGCN: a ReRAM-based PIM design for graph convolutional network acceleration. In: Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC), 2021. 583–588

76 Liu F, Zhao W, Chen Y, et al. PIM-DH: ReRAM-based processing-in-memory architecture for deep hashing acceleration. In: Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC), 2022. 1087–1092
77 Mamdouh A, Geng H, Niemier M, et al. Shared-PIM: enabling concurrent computation and data flow for faster processing-in-DRAM. 2024. ArXiv:2408.15489
78 Kim J H, Kang S H, Lee S, et al. Aquabolt-XL HBM2-PIM, LPDDR5-PIM with in-memory processing, and AXDIMM with acceleration buffer. IEEE Micro, 2022, 42: 20–30
79 Chi P, Li S, Xu C, et al. Prime: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. ACM SIGARCH Comput Archit News, 2016, 44: 27–39
80 Wang Y, Han Y, Zhang L, et al. ProPRAM: exploiting the transparent logic resources in non-volatile memory for near data computing. In: Proceedings of the 52nd IEEE Design Automation Conference (DAC), 2015. 1–6
81 Chiang H W, Nien C F, Cheng H Y, et al. ReAIM: a ReRAM-based adaptive ising machine for solving combinatorial optimization problems. In: Proceedings of the 51st ACM/IEEE Annual International Symposium on Computer Architecture (ISCA), 2024. 58–72
82 Li S, Xu C, Zou Q, et al. Pinatubo: a processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories. In: Proceedings of the 53rd IEEE Design Automation Conference (DAC), 2016. 1–6
83 Bavikadi S, Sutradhar P R, Ganguly A, et al. UPIM: performance-aware online learning capable processing-in-memory. In: Proceedings of IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2021. 1–4
84 Sharma H, Mandal S K, Doppa J R, et al. SWAP: a server-scale communication-aware chiplet-based manycore PIM accelerator. IEEE Trans Comput-Aided Des Integr Circuits Syst, 2022, 41: 4145–4156
85 Angizi S, Sun J, Zhang W, et al. PIM-Aligner: a processing-in-MRAM platform for biological sequence alignment. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020. 1265–1270
86 Shan W W, Cui Y Q, Dai W T, et al. An efficient path delay variability model for wide-voltage-range digital circuits. Sci China Inf Sci, 2023, 66: 129401
87 Luo X, Zhang C, Geng C B, et al. TSCompiler: efficient compilation framework for dynamic-shape models. Sci China Inf Sci, 2024, 67: 200403
88 Zhang J Y, Shen J R, Wang Z K, et al. SpikingMiniLM: energy-efficient spiking transformer for natural language understanding. Sci China Inf Sci, 2024, 67: 200406
89 Xia Z H, Wan R, Chen J N, et al. Reconfigurable spatial-parallel stochastic computing for accelerating sparse convolutional neural networks. Sci China Inf Sci, 2023, 66: 162404
90 Sandler M, Howard A, Zhu M, et al. MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 4510–4520
91 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 770–778
92 Tong W, Liu Y. Recent progress of layered memristors based on two-dimensional MoS_2. Sci China Inf Sci, 2023, 66: 160402
93 Zhao Y, Gao M, Liu F, et al. UM-PIM: DRAM-based PIM with uniform & shared memory space. In: Proceedings of ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA), 2024
94 Tian B, Li Y, Jiang L, et al. NDPBridge: enabling cross-bank coordination in near-DRAM-bank processing architectures. In: Proceedings of ACM/IEEE 51st International Symposium on Computer Architecture (ISCA), Buenos Aires, 2024. 628–643
95 Yu Z, Liang S, Ma T, et al. Cambricon-LLM: a chiplet-based hybrid architecture for on-device inference of 70B LLM. In: Proceedings of the 57th IEEE/ACM International Symposium on Microarchitecture (MICRO), 2024, 1474–1488
96 Heo G, Lee S, Cho J, et al. NeuPIMs: NPU-PIM heterogeneous acceleration for batched LLM inferencing. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating, 2024. 722–737
97 Li C, Zhou Z, Zheng S, et al. SpecPIM: accelerating speculative inference on PIM-enabled system via architecture-dataflow co-exploration. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating, 2024. 950–965