Proceedings of CONF-FMCE 2026 Symposium: Smart City and Infrastructure Engineering
DOI: 10.54254/2755-2721/2026.CH38550

In-Memory Computing Architectures for Accelerating Large-Scale Models: Principles, Devices, and System Challenges

Tianhao Gui

Adcote School Shanghai, Shanghai, China
tianhaodeyouxiang@hotmail.com

Abstract. Memory bandwidth is the binding constraint for large-model inference today. In-memory computing (IMC) addresses this limitation. This paper reviews where the technology stands. It compares analog and digital IMC implementations and find that analog arrays currently lead on energy efficiency, while digital designs maintain a substantial edge in precision and programming flexibility. Then it discusses three device candidates: Resistive Random-Access Memory (ReRAM), Ferroelectric Field-Effect Transistor (FeFET), and Spin-Transfer-Torque Magnetic Random-Access Memory (STT-MRAM). ReRAM offers the highest density but suffers from stochastic filament growth and conductance drift. FeFETs integrate cleanly with CMOS but are limited by ferroelectric switching variability. STT-MRAM is the most durable and stable, yet its modest tunnel-magnetoresistance ratio restricts analog precision. Finally, three limitations are pointed out: device variation under realistic workloads, peripheral-circuit overhead that can erase array-level energy gains, and scalable multi-chip orchestration for trillion-parameter models. IMC will succeed or fail based on cross-layer co-design—not on any single device breakthrough. Materials, circuits, compilers, and algorithms must advance together, and the compiler layer remains underdeveloped relative to device research. This paper provides development ideas for higher performance for IMC.

Keywords: In-memory computing, Processing-in-memory, Neural network accelerator

1. Introduction

Large language models now contain trillions of parameters, and serving them at reasonable latency requires hundreds of tera-operations per second [1, 2]. These numbers are often cited as evidence that compute is the bottleneck. The bottleneck is data movement. In a von Neumann machine, every weight must travel from memory to the arithmetic unit before it can be used, and in contemporary GPU deployments this traffic accounts for roughly 70% of total energy and latency. A single byte moved across the memory hierarchy costs orders of magnitude more than the arithmetic operation that consumes it. As models grow, the gap widens. Compute units spend most of their time idle, waiting for weights to arrive, and this under-utilization has become the central obstacle to deploying larger models at acceptable cost.

Processing-in-memory (PIM) takes the opposite approach. Instead of bringing data to the computation, it brings computation to the data [3]. Early PIM work used Static Random-Access

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.CH38550

Memory (SRAM) and Dynamic Random Access Memory (DRAM) cells for simple bitwise operations in binary neural networks, but these designs had clear limits: SRAM is too dense-expensive, and DRAM is volatile. The field changed direction when researchers realized that emerging non-volatile memories could serve two purposes at once—storing multi-bit weights and participating directly in multiply-accumulate operations [4, 5]. This was more than a circuit trick. It turned IMC into a full-stack research problem, spanning device physics, array organization, peripheral circuits, and software mapping. On specific inference kernels, the energy gains are already substantial—published ReRAM chips have exceeded conventional GPU efficiency by a wide margin [6].

No memory technology has yet demonstrated the numerical precision that large models require under realistic process variation. Scheduling and operator-mapping interfaces are ad hoc at best. Multi-chip systems do not scale gracefully to the parameter counts of frontier models. And the software stack for IMC is years behind the device research, which most compilers still assume GPU or CPU execution models. This paper reviews the IMC technology in details. Section 2 explains how IMC works at the array level and contrasts digital with analog implementations. Section 3 examines Resistive Random-Access Memory (ReRAM), Ferroelectric Field-Effect Transistor (FeFET), and Spin-Transfer-Torque Magnetic Random-Access Memory (STT-MRAM) in detail. Section 4 focuses on system-level challenges: device non-ideality, peripheral overhead, and scalability. The final is the conclusion of this paper.

2. Computational principles of IMC

IMC relocates arithmetic activity into the memory array so weights need not leave their storage locations for every operation [7]. Consider what happens in a conventional pipeline: each matrix-vector multiplication requires the processor to fetch both activations and weights, compute a partial product, and write intermediate results back to memory. Modern networks contain hundreds of millions or billions of parameters, so this traffic dominates both energy and time. An IMC array eliminates most of this movement. The weight matrix is stored as conductance, resistance, or threshold-voltage states. Input activations are driven onto array rows as voltages, currents, or pulse trains. Column currents—or digital bit-line results—accumulate the weighted sums in parallel. The resulting massively concurrent multiply-accumulate behavior matches the dominant computational pattern of deep networks.

A practical IMC subsystem comprises more than just the memory array. It needs row drivers, column accumulation or sensing circuits, analog-to-digital converters when required, digital post-processing logic, and a control interface. In SRAM-based digital designs, weights remain binary or low-bit digital values and computation proceeds through XNOR, AND, or population-count logic. In non-volatile designs, the same physical quantities encoding stored data also participate in the arithmetic transformation: conductance multiplies input voltage according to Ohm's law, and Kirchhoff's current law performs the summation. Positive and negative weights are commonly realized by differential cell pairs, which also help cancel common-mode errors affecting both cells equally. Large layers are tiled across multiple smaller arrays to mitigate IR drop and parasitic resistance [8].

Non-ideal device behavior introduces errors that are especially pronounced in analog realizations: process variation, conductance drift, nonlinear switching, limited endurance, and read noise. Counter-measures include low-bit quantization, bit-slicing, closed-loop write-and-verify programming, calibration circuits, and hardware-aware training. Analog systems further incur the area and energy cost of data converters. Overall efficiency depends as much on peripheral circuitry,

DOI: 10.54254/2755-2721/2026.CH38550

mapping strategy, and quantization policy as on the intrinsic properties of the memory cell. For transformer-based large models, additional requirements arise: efficient handling of key-value caches, attention scores, and intermediate activations that do not map cleanly onto pure matrix-vector multiplication.

3. Device technologies for IMC

3.1. ReRAM

ReRAM is the most widely studied IMC device. A crossbar array can execute thousands of multiply-accumulate operations simultaneously—parallelism that no digital architecture matches at this density. The operating principle is straightforward. A metal-insulator-metal stack changes resistance when nanometer-scale conductive filaments form or rupture inside the insulator. Apply voltages to the rows, and the resulting column currents are exactly the weighted sums of a matrix-vector multiplication, courtesy of Ohm's and Kirchhoff's laws. Binary states encode single bits. Intermediate conductance levels can encode multi-bit weights. The problem is that filament growth is stochastic. Individual ions move unpredictably, so two cells programmed to the same target conductance rarely end up at the same value. This variability is manageable for binary states but becomes a serious obstacle for multi-bit precision, which is what large models need.

PRIME, an early architectural proposal, mapped entire neural-network layers onto ReRAM crossbars with peripheral converters and digital logic, reporting meaningful energy-efficiency gains over conventional processors [9]. Wan et al. later demonstrated a ReRAM compute-in-memory chip achieving 11.5 TOPS/W for 8-bit inference [6]. Fully hardware-implemented memristor convolutional neural networks have further shown that the approach can scale beyond small test circuits [10]. Hardware-aware training, which injects device variation into the learning loop, has recovered a good deal of the accuracy lost to imperfect programming. These are real advances. But the fundamental problems have not gone away. Conductance drift over time. Cycle-to-cycle variability. Nonlinear weight updates. Limited endurance. IR drop in large arrays. Each of these gets worse as arrays grow, and analog-to-digital conversion adds its own area and power cost on top. The field's current response—low-bit quantization, tiled sub-arrays, mixed-signal calibration, algorithm-device co-optimization—is reasonable, but it remains to be seen whether these techniques can scale to the precision and array sizes that large-model inference demands.

3.2. FeFETs

FeFETs stand apart from ReRAM and STT-MRAM in one important respect: they are transistors. A ferroelectric layer—usually hafnium-zirconium oxide—is built into the gate stack. When polarization switches, it changes the threshold voltage, which changes the channel conductance. The result is a non-volatile, multi-level weight storage element that is also an active device. First, the access transistor suppresses sneak-path currents, which are a major scalability problem for passive ReRAM crossbars. Second, FeFETs are compatible with standard CMOS processes, so memory and logic can be integrated on the same die without exotic fabrication steps. Both advantages are significant.

Prototype work has been encouraging. Analog IMC macros with programmable threshold voltages have demonstrated in-memory multiply-accumulate operations at low energy [11]. Ni et al. built a ferroelectric ternary content-addressable memory that achieves one-shot learning with ultra-low energy consumption [12]. Fast switching, modest operating voltages, and non-volatile storage

DOI: 10.54254/2755-2721/2026.CH38550

make FeFET arrays particularly interesting for edge devices that power-cycle frequently. Density continues to improve through multi-bit storage and capacitor-free cell designs. The downside is precision. Ferroelectric switching is inherently stochastic, and device-to-device variation, polarization fatigue, and retention loss all make accurate analog programming difficult. Practical systems compensate with write-and-verify sequences, calibration, quantization-aware training, and variation-aware mapping—techniques that add complexity and overhead. Whether FeFETs can achieve the multi-bit precision needed for large-model inference remains an open question.

3.3. STT-MRAM

STT-MRAM offers excellent endurance and stable resistance states but limited analog capability. The device is a magnetic tunnel junction: a fixed layer, a tunnel barrier, and a free layer. When the magnetizations of the two magnetic layers are parallel, resistance is low; when antiparallel, resistance is high. The tunnel-magnetoresistance ratio—typically 150–200%—determines how clearly these two states can be distinguished. Non-volatile storage with near-zero standby leakage, nanosecond-scale switching, and high endurance make STT-MRAM attractive for on-chip caches and frequently accessed weight matrices. These are genuinely useful properties.

For IMC, the robust binary switching behavior is well suited to binary, ternary, and heavily quantized networks. MTJ resistances can encode quantized weights, and sensing the resulting column currents implements logic or low-precision multiply-accumulate operations [13]. Jain et al. demonstrated energy-efficient binary neural network inference using this approach [14]. But analog precision is hard to come by. Write currents are relatively large. Neighboring cells can suffer write disturbance. The modest TMR ratio means multi-level representations are constrained. Stochastic magnetic switching and sensing errors add further uncertainty. For these reasons, STT-MRAM is most likely role not as a standalone IMC technology, but as a component in hybrid systems, such as providing non-volatile weight storage alongside ReRAM compute arrays or digital processing elements. Emerging spin-orbit-torque and voltage-controlled magnetic devices could eventually lower write energy while preserving endurance, but these are still at a relatively early stage.

4. System-level limitations and design pathways

4.1. Device non-ideality versus numerical fidelity

Analog IMC stores weights as continuous conductance or resistance values. In theory, this enables dense, efficient matrix multiplication. In practice, the values are not stable. Process variation means two cells manufactured identically will have different conductance distributions. Cycle-to-cycle fluctuations mean the same cell programmed twice will not end up at the same value. Conductance drift means values change over time. Nonlinear programming, read noise, and finite endurance compound the problem. These errors accumulate across large crossbars and deep networks, and the effect is most severe when multi-bit weights are used—precisely the regime that large models require. Parasitic line resistance causes voltage drops that further distort results. Leakage and noise inject additional uncertainty.

At the device level, better materials and tighter process control reduce conductance spread, and closed-loop write-and-verify programming ensures each cell reaches its target. But variation cannot be eliminated entirely—physics sets a floor. At the circuit level, reference cells, differential sensing, adaptive readout, and on-array calibration compensate for what device-level control cannot fix. Each of these techniques adds area and power, however, so there is a real cost. At the algorithm level,

DOI: 10.54254/2755-2721/2026.CH38550

hardware-aware training has become essential. By injecting measured device noise and variation into the training loop, models learn to tolerate the specific imperfections of a given technology rather than assuming ideal arithmetic. Low-bit quantization, error-correction codes, and noise-aware mapping provide additional robustness. The accuracy ceiling of analog IMC will be determined not by device variation alone, but by how well these three layers compensate for one another.

4.2. Peripheral overhead and scalability to ultra-large models

In some published designs, peripherals occupy more than half the total chip area. High-resolution converters are particularly problematic. A single analog-to-digital conversion can consume more energy than the entire array operation it digitizes. When this happens, the energy advantage of in-array computation is largely erased. This is not a minor implementation detail—it is a fundamental constraint on analog IMC efficiency.

Scalability is the second major problem, and it is largely independent of the first. Frontier models have billions to trillions of parameters, and weight matrices must be partitioned across multiple arrays or multiple chips, which introduces inter-chip communication, synchronization overhead, and data-transfer energy. Transformer architectures make this harder. Attention requires frequent, irregular access to key-value caches and intermediate activations—access patterns that do not map naturally onto the regular matrix-vector engines that IMC does best. On top of this, existing compilers, programming models, and software stacks are designed for GPUs and CPUs. Mapping workloads efficiently onto heterogeneous IMC hardware remains a largely manual, ad hoc process.

Hybrid heterogeneous architectures are the most promising response to these problems. Mixed-signal designs use analog arrays for low-precision matrix multiplication—the workload they handle best—while digital units manage control flow, nonlinearities, and high-precision stages. Peripheral overhead can be reduced by lowering converter resolution, adopting time- or charge-domain computation, and sharing conversion resources across arrays. For scalability, chiplet-based organizations with high-bandwidth-memory integration offer a path to larger effective capacity and better inter-module bandwidth. HBM-PIM prototypes already demonstrate the benefits of placing compute closer to dense memory, and multiple IMC chiplets interconnected through advanced packaging could achieve the weight capacity needed for billion-parameter models. Future compilers must automatically partition layers, select quantization levels, map weights onto physical arrays, and compensate for device variation.

5. Conclusion

This paper comprehensively reviews the current state and trends of in-memory computing technologies for accelerating large-scale models. First, the core computational principles of IMC and the evolution from digital to analog implementations are systematically introduced, highlighting how IMC addresses the memory wall bottleneck in von Neumann architectures. Second, three representative emerging non-volatile memory technologies: ReRAM, FeFET, and STT-MRAM, are analyzed in detail, examining their device structures, computational mechanisms, and performance characteristics. Third, system-level challenges are discussed, including device non-ideality, peripheral overhead, and scalability limitations for ultra-large models, with emphasis on cross-layer co-design strategies spanning devices, circuits, and algorithms.

DOI: 10.54254/2755-2721/2026.CH38550

References

[1] Guo, D., Zhu, Q., Yang, D., et al. (2024). Deepseek-coder: When the large language model meets programming–the rise of code intelligence. arXiv.

[2] Liu, A., Feng, B., Xue, B., et al. (2024). Deepseek-v3 technical report. arXiv.

[3] Yang, T., Li, D., Ma, F., et al. (2022). Pasgcn: An ReRAM-based PIM design for GCN with adaptively sparsified graphs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 42*, 150-163.

[4] Wang, J., Ge, M., Ding, B., et al. (2023). NicePIM: Design space exploration for processing-in-memory DNN accelerators with 3-D stacked-DRAM. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 43*, 1456-1469.

[5] Sudarshan, C., Soliman, T., Delaparra, C., et al. (2021). ANovelDRAM-BasedProcess-in-Memory architecture and its implementation for CNNs. *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, 35-42.

[6] Wan, W., Kubendran, R., Schaefer, C., et al. (2022). A compute-in-memory chip based on resistive random-access memory. *Nature, 608*, 504-512.

[7] Zhu, Z. H., Zhu, Y., Wang, Y., et al. (2021). Simulator design for memristor-based computing-in-memory architectures: A review and perspective. *Micro-nano Electronics and Intelligent Manufacturing, 3*, 42-49.

[8] Xia, Q., & Yang, J. J. (2019). Memristive crossbar arrays for brain-inspired computing. *Nature Materials, 18*, 309-323.

[9] Chi, P., Li, S., Xu, C., et al. (2016). PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 27-39.

[10] Yao, P., Wu, H., Gao, B., et al. (2020). Fully hardware-implemented memristor convolutional neural network. *Nature, 577*, 641-646.

[11] Dünkel, S., Trentzsch, M., Richter, R., et al. (2017). A FeFET based super-low-power ultra-fast embedded NVM technology for 22nm FDSOI and beyond. *2017 IEEE International Electron Devices Meeting (IEDM)*, 19.7.1–19.7.4.

[12] Ni, K., Yin, X., Laguna, A. F. B., et al. (2019). Ferroelectric ternary content-addressable memory for one-shot learning. *Nature Electronics, 2*, 521–529.

[13] Apalkov, D., Dieny, B., & Slaughter, J. M. (2016). Magnetoresistive random access memory. *Proceedings of the IEEE, 104*, 1796–1830.

[14] Jain, S., Ranjan, A., Roy, K., et al. (2018). Computing in memory with spin-transfer torque magnetic RAM. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 26*, 470–483.