SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2026, Vol. 69, Iss. 9, 192401:1–192401:7
https://doi.org/10.1007/s11432-025-4826-6

A 64-bit high-precision flash computing-in-memory architecture with a parallel input scheme

Yang FENG[1], Xinyi GUO[1], Chang CHEN[1], Yixuan FAN[1], Xiaopeng LI[1], Bo CHEN[1], Junyu ZHANG[2], Jing LIU[3], Xuepeng ZHAN[1], Jixuan WU[1*] & Jiezhi CHEN[1*]

[1]*School of Information Science and Engineering, Shandong University, Qingdao 266200, China*
[2]*Neumem Co., Ltd., Hefei 230093, China*
[3]*Key Laboratory of Microelectronic Devices and Integrated Technology, Institute of Microelectronics of Chinese Academy of Sciences, Beijing 100084, China*

Received 13 October 2025/Revised 23 December 2025/Accepted 2 March 2026/Published online 17 August 2026

Abstract Computing-in-memory (CIM) architecture is a promising solution for enhancing energy efficiency and reducing latency by minimizing data movement. With the rapid expansion of the big data era, high-performance computing (HPC) demands not only computing abilities but also storage capacities. Thereby, to utilize CIM for future high-performance applications, further advancements in computing precision are essential. To address this concern, we present a novel parallel input scheme for high-precision computing in flash-based CIM with high bit-density (4 bits/cell). The proposed scheme leverages the array and operational design to achieve a 64-bit precision computing system. By implementing the parallel input scheme, pulse time, latency, and power consumption that are primarily dominated by peripheral circuits could be effectively reduced. Our work provides a feasible approach to constructing a high-precision CIM architecture.

Keywords computing-in-memory, general-purpose computing, precision extension, flash memory

Citation Feng Y, Guo X Y, Chen C, et al. A 64-bit high-precision flash computing-in-memory architecture with a parallel input scheme. Sci China Inf Sci, 2026, 69(9): 192401, https://doi.org/10.1007/s11432-025-4826-6

1 Introduction

Vector-matrix multiplication (VMM), consisting of a series of multiply-accumulate operations (MACs), serves as a fundamental computational block in modern computing applications [1]. A prominent example is machine learning, which heavily relies on VMM operations and demands frequent memory access. This intensive computational load in VMM operations is significantly constrained by the von Neumann bottleneck [2], where continuous data transfer between memory and processing elements (PEs) results in high energy consumption and latency. To overcome this challenge, computing-in-memory (CIM) has gained increasing attention as it integrates computation directly within memory, leveraging high parallelism to minimize data movement and improve efficiency.

Many emerging memory technologies have been explored for CIM applications, and various neural network (NN) architectures and signal processing tasks have been implemented using CIM-based designs [3, 4]. The inherent error tolerance of neural networks reduces the dependence on high-precision computation while still achieving high recognition accuracy. However, certain applications impose stricter demands on computational precision, making it a critical challenge for CIM. In fields such as scientific computing, financial engineering, and cryptography [5, 6], maintaining high computational precision is crucial. High precision can reduce error accumulation and improve the computation range, which ensures the computational accuracy.

Recently, 32-bit computing architectures were demonstrated using various memory technologies, including resistive random access memory (RRAM) [7], static random access memory (SRAM) [8], and flash memory [9]. However, the 32-bit precision currently achievable in CIM applications remains insufficient to meet the computational demands of modern high-performance computing (HPC) [10]. Higher precision architectures, such as 64-bit systems, are required [11], which can enable extended bit-width operations and offer improved performance. Meanwhile, as data processing demands continue to grow, large-scale CIM architectures impose increasingly stringent requirements on memory technologies. Advancing CIM architecture chips necessitates improvements in memory performance and

* Corresponding author (email: jixuanwu@sdu.edu.cn, chen.jiezhi@sdu.edu.cn)

stability, particularly for applications requiring superior accuracy. When targeting large-scale and high-precision computing, RRAM faces limitations due to its maximum array sizes [12]. When tackling large-scale problems, SRAM-based accelerators require additional computation cycles because SRAM's volatility forces frequent weight movement and rewriting, whereas flash-based CIM stores weights non-volatile in the compute array and eliminates this overhead [8]. Flash memory—with its mature technology, high bit density, and robust reliability, even in large arrays [13,14]—emerges as a promising candidate among emerging memories for designing high-precision computing architectures.

In conventional systems (e.g., CPUs and GPUs), transitioning from 32-bit to 64-bit operations typically increases computational complexity and memory overhead, often reducing processing speed by up to 50% [15]. This trade-off affects performance in areas like scientific simulation and graphics rendering [16], where algorithmic optimization or mixed-precision strategies are often necessary to maintain efficiency [17].

Although floating-point computation is widely adopted in these systems [18–20], floating-point arithmetic requires exponent alignment, normalization, and rounding logic, which increases hardware complexity and introduces nondeterministic numerical behavior. These challenges become even more pronounced in CIM architectures, where peripheral overhead and analog variability amplify the cost of implementing floating-point operations. As a result, for high-precision CIM, integer-based computation offers a more practical and deterministic alternative with lower hardware complexity.

Similarly, in CIM systems, the benefits of massive parallelism are challenged by the increased complexity of higher precision computing. Since a single memory cell cannot support 64-bit precision directly, computations must be decomposed into lower-bit components, leading to increased processing complexity and latency. Consequently, precision extension scheme optimizations are essential in CIM design to achieve a balance between device capabilities and system performance requirements. These optimizations play a pivotal role in enhancing the overall performance of high-precision computing systems.

In this work, we propose a flash-based 64-bit integer precision CIM system that offers high speed and energy efficiency. The system is built using flash memory CIM arrays with a large memory window (>6.5 V) and ultra-high bit density (4–5 bits per cell), ensuring high storage capacity and robust reliability. The stability of the large memory window is thoroughly evaluated to ensure consistent computing performance. Importantly, we optimize the precision extension scheme tailored for 64-bit computing, incorporating a novel parallel input scheme to enhance processing speed. By enabling all bit slices of a wide input word to be processed in parallel, the proposed input-mapping scheme removes sequential bit-slice operations and substantially improves latency and energy efficiency. The system's performance is comprehensively assessed through simulations, considering the impact of peripheral circuit design to validate its overall efficiency and scalability.

2 Precision extension strategy

Achieving higher precision, such as 64-bit precision throughout computations, is essential for various applications, including scientific computing [21–23], to ensure accurate and reliable results. However, achieving 64-bit precision with high accuracy using a single flash memory cell is impractical due to the limited number of storage states. Each memory state typically exhibits a distribution in threshold voltage ($V_{\rm th}$), and the voltage margin must be sufficiently large to mitigate the impact of device-to-device and cycle-to-cycle variations on storage accuracy. Consequently, the number of reliable memory states within a given memory window is restricted. To attain higher computing precision in CIM applications within this constrained memory window, precision bits must be segmented into smaller bit units.

The single flash memory cell can generally support 16-level storage (4-bit) for CIM applications. Therefore, 64-bit computing precision can be achieved by dividing it into sixteen 4-bit arrays using an equal precision extension scheme, as illustrated in Figure 1(a). For example, if X and Y are 16-bit data, they can be represented as

$$X = \begin{pmatrix} (a_3, a_2, a_1, a_0) \\ (b_3, b_2, b_1, b_0) \\ \vdots \\ (c_3, c_2, c_1, c_0) \end{pmatrix} = (X_3, X_2, X_1, X_0)\,, \tag{1}$$

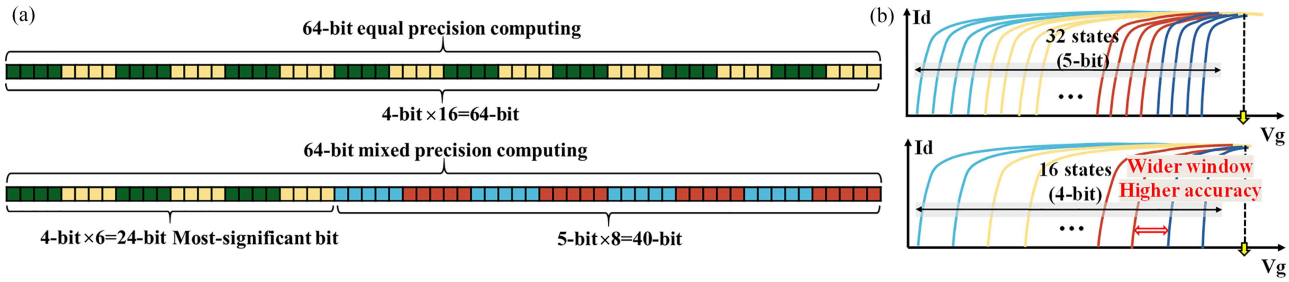

Figure 1 (Color online) (a) Comparison of 64-bit equal precision computing and 64-bit mixed precision computing. (b) 5-bit/cell and 4-bit/cell operations require 32 states and 16 states per cell, respectively. For the same cell, the 4-bit/cell operation provides a wider memory window margin per bit, reducing the error rate and improving computing accuracy.

$$Y = \begin{pmatrix} (d_3, d_2, d_1, d_0) \\ (e_3, e_2, e_1, e_0) \\ \vdots \\ (f_3, f_2, f_1, f_0) \end{pmatrix} = (Y_3, Y_2, Y_1, Y_0)\,, \tag{2}$$

where a_3, a_2, a_1, a_0 are 4-bit vectors. The multiplication results can be represented by

$$\begin{aligned} X \cdot Y = {} & l^6 (X_3 \cdot Y_3) + l^5 (X_3 \cdot Y_2) + l^4 (X_3 \cdot Y_1) + l^3 (X_3 \cdot Y_0) + l^5 (X_2 \cdot Y_3) + l^4 (X_2 \cdot Y_2) + l^3 (X_2 \cdot Y_1) \\ & + l^2 (X_2 \cdot Y_0) + l^4 (X_1 \cdot Y_3) + l^3 (X_1 \cdot Y_2) + l^2 (X_1 \cdot Y_1) + l^1 (X_1 \cdot Y_0) + l^3 (X_0 \cdot Y_3) + l^2 (X_0 \cdot Y_2) \\ & + l^1 (X_0 \cdot Y_1) + (X_0 \cdot Y_0)\,, \end{aligned} \tag{3}$$

where l denotes the digit shifts (4-bit). Besides, $(X_i \cdot Y_j)$ should be computed as normal vector matrix multiplication, which can be processed by the flash memory. The 16-bit data division is shown as an example, and this precision extension method can also be extended to achieve 64-bit precision.

In equal precision computing, all bits are treated with equal significance, which ensures consistent accuracy across each bit. However, in certain practical computing, the most significant bit (MSB) has a greater influence on overall precision. Therefore, mixed-precision computing can be employed, where the MSBs are stored with a larger memory window margin per bit while the least significant bits (LSBs) are stored using a smaller memory window margin per bit. This approach allows for maintaining computing precision while reducing the number of required memory arrays and lowering energy consumption. As shown in Figure 1(b), decreased bit density within the same memory window effectively helps reduce the data error rate, thereby improving computing accuracy. This mixed-precision strategy enables dynamic adjustment of bit density according to the precision requirements of specific computing tasks.

3 Device characterizations to extend precision

In this work, NOR flash memory is employed to validate a 64-bit high-precision CIM system. Figure 2(a) presents the array structure and TEM image of a 55 nm NOR flash memory, highlighting its scalability for large-scale applications with reliable fabrication. To accommodate the requirements of large-scale CIM for flash memory, the programming scheme must be optimized for both efficiency and precision, which are influenced by programming speed and a sufficient memory window. This work employs channel hot electron injection (CHEI), Fowler-Nordheim (FN) tunneling, and hot hole injection (HHI) to achieve precise $V_{\rm th}$ adjustment within a wide memory window.

When comparing the two erasing methods, FN tunneling and HHI, FN tunneling is more effective for achieving a large memory window under an erase gate voltage of -8 to -9.5 V and a substrate voltage of 7.7 V. In contrast, HHI enables finer and faster $V_{\rm th}$ tuning [24] using a drain voltage of 3 to 5 V and a gate voltage of -7 to -9.5 V. The choice of erasing method depends on the required memory window for different bit densities. Both CHEI and HHI enable individual cell tuning, mitigating the effects of over-programming and enhancing modulation efficiency, thereby improving programming accuracy. Additionally, CHEI and HHI are highly effective for ultra-fast $V_{\rm th}$ adjustment in NOR flash cells [25]. In contrast, FN tunneling enables slower erasing while providing a larger memory window. Moreover, HHI performs at lower voltages than FN tunneling, which could help minimize performance degradation caused by cycling and reduce power consumption during programming.

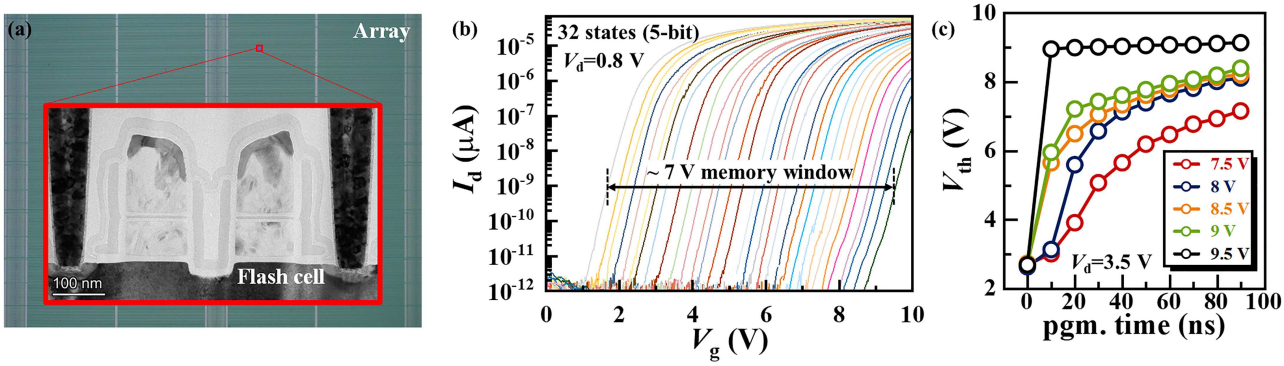

Figure 2 (Color online) (a) Cross-sectional TEM image of the fabricated 55 nm NOR flash memory array. (b) Measured *I*-*V* curves of flash memory in different memory states. The flash memory is capable of 32 states (5-bit) storage. (c) The programming speed with different V_g from 7.5 to 9.5 V at various programming pulses at V_d. High programming voltage can achieve higher memory states.

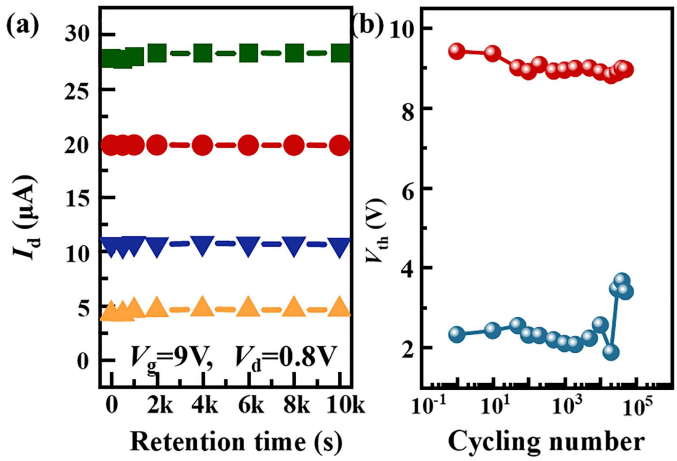

Figure 3 (Color online) (a) The retention characteristics of flash memory cells in different memory states; (b) the endurance of flash memory with a large memory window at room temperature.

The *I*-*V* curves of the flash memory using the combined programming scheme are shown in Figure 2(b), demonstrating its capability to achieve 5-bit (32 states) storage with an acceptable memory window. The large memory window and high on-off ratio enable stable computing performance, whereas other emerging memory technologies often require further optimization. Beyond its 5-bit storage capability, the programming ability to achieve a large memory window is also confirmed. Figure 2(c) illustrates the V_{th} adjustment range under varying programming pulses and gate voltages (V_g). To ensure consistent programming effects across different voltages, the initial V_{th} is set to the same value. As V_g increases from 7.5 to 9.5 V, the V_{th} adjustment range expands significantly. At 9.5 V, a memory window approaching 7 V (with the highest V_{th} surpassing 9 V) is achieved. Reliability characteristics, including retention and endurance, are rigorously tested to ensure computational accuracy. Figure 3(a) demonstrates that various current levels remain stable over 10000 s, highlighting the robust retention characteristics of the memory cells. Furthermore, the endurance results depicted in Figure 3(b) reveal that a large memory window is maintained even after 10^5 cycles, confirming the feasibility of implementing high bit-density schemes. The flash-based CIM arrays, with their wide memory window, enable high bit-density programming to implement complex computation tasks. These reliability results collectively demonstrate that the memory cells can execute computing tasks consistently and stably, without significant accuracy degradation over prolonged operation.

4 Parallel input scheme

Analog CIM operations offer significant advantages in performing VMM with exceptional energy efficiency. This advantage stems from their highly parallel computing architecture, which enables ultra-large-scale matrix-vector multiplications to be executed simultaneously, thereby circumventing limitations imposed by data transfer bandwidth and the restricted parallelism of registers.

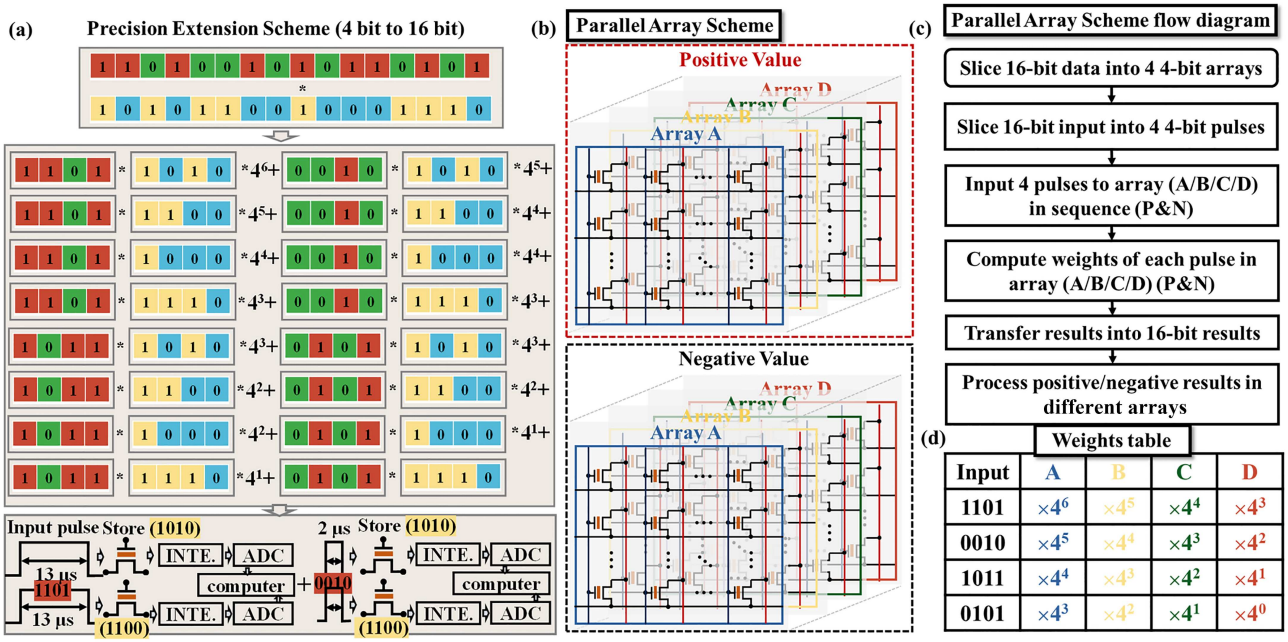

Figure 4 (Color online) (a) The precision extension scheme and computing principle. The precision is extended from 4-bit to 16-bit as an example. (b) The schematic of the parallel array scheme. The 16-bit data extension strategy is exhibited as an example. 16-bit data are sliced into four 4-bit data, which is stored in the Arrays A–D. In addition, the negative and positive data need to be stored in different arrays. (c) The flow diagram of the parallel array scheme. (d) The 16-bit input also needs to be sliced into four 4-bit inputs, which needs to be input 4 times. For different inputs, the weights of the output need to be adjusted, which can also refer to (a).

In the majority of analog VMM operations, a precision extension strategy is employed, wherein matrix values are represented using distinct conductance levels. In addition, positive and negative values are mapped separately onto different arrays. These design choices significantly enhance parallelism, boosting computational efficiency, particularly for large-scale operations.

However, the precision extension process is resource-intensive, as illustrated in Figure 4(a). For instance, multiplying two 16-bit vectors requires slicing each vector into four 4-bit segments, followed by multiple multiplications to achieve the desired precision. Fortunately, these multiplications can be executed in parallel, along with the processing of positive and negative values, as depicted in Figure 4(b). Figure 4(c) outlines the workflow for multiplying two 16-bit vectors, while Figure 4(d) provides a weights table that indexes the weights used in precision extension. In this scheme, one operand is represented by the $V_{\rm th}$ mapped to the flash array, and the other is represented by pulse duration. The integration of the resulting current yields the VMM output, which remains a 16-bit vector. The process can be described as

$$\begin{bmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{bmatrix} \begin{bmatrix} W_1^1 & W_2^1 & \dots & W_N^1 \\ W_1^2 & \ddots & \ddots & W_N^2 \\ \vdots & \dots & \ddots & \vdots \\ W_1^N & W_2^N & \dots & W_N^N \end{bmatrix} = \begin{bmatrix} F(1) \\ F(2) \\ \vdots \\ F(N) \end{bmatrix}. \tag{4}$$

Additionally, positive and negative components are computed separately, with the final result obtained by subtracting the negative component from the positive one.

The precision extension method can be further optimized by reconfiguring the mapping strategy. In the traditional approach, each 4-bit segment of a 16-bit operand must be multiplied by every 4-bit segment of the other operand. This necessitates four 4-bit flash arrays to store the data, and the pulse-based data must be input sequentially four times, leading to increased latency and power consumption. Such delays are particularly undesirable in CIM applications requiring high processing speeds. However, it was observed that certain data points (e.g., 4^1, 4^2, 4^3) share the same weights in the weights table, enabling them to be processed simultaneously.

To enhance parallelism, we propose a parallel input scheme. In this approach, matrix values are mapped to the flash array in the arrangement shown in Figure 5(a). For VMM results involving the same weights, these values can share the same bit line (BL) and peripheral circuits, simplifying the complex circuit designs required in traditional

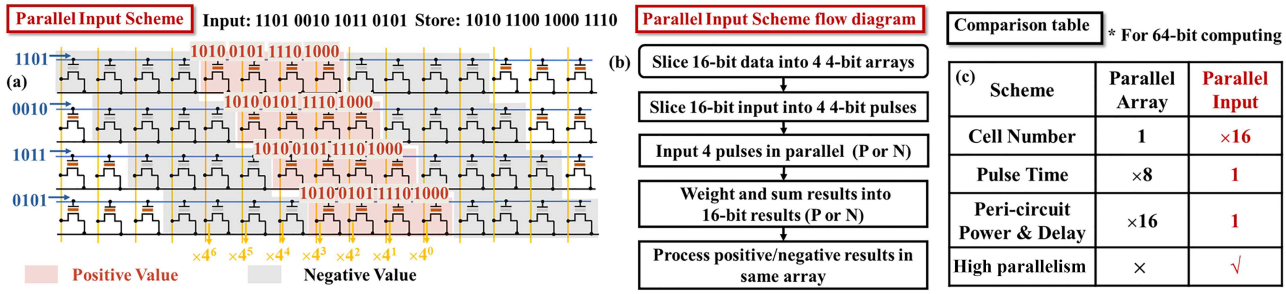

Figure 5 (Color online) (a) The schematic of the parallel input scheme. One 16-bit data sliced into 4 different cells needs to be stored 4 times to achieve parallel input to improve computing efficiency and reduce the peripheral circuits' consumption. (b) The flow diagram of the parallel input scheme. (c) The comparison of the parallel array scheme and the parallel input scheme.

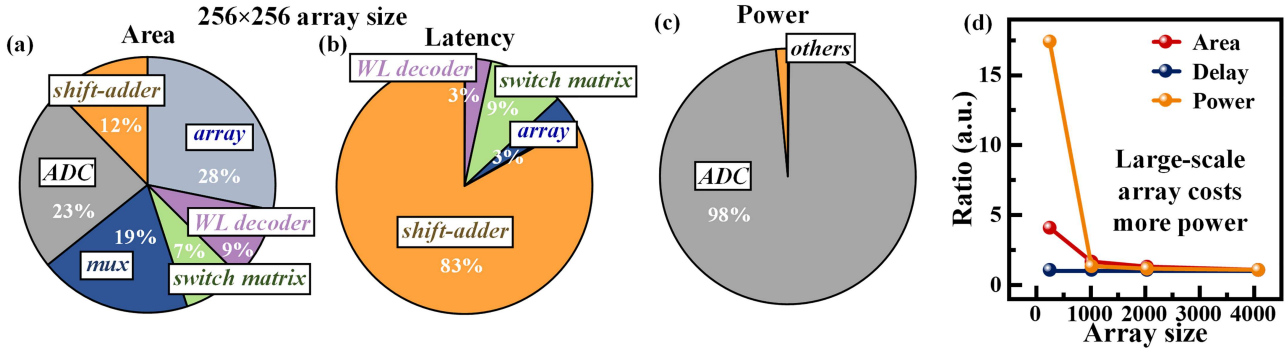

Figure 6 (Color online) Performance analysis of flash memory arrays. (a) Area distribution between memory array and peripheral circuits in a 256 × 256 array. (b) Latency distribution. (c) Power consumption distribution. In small arrays, peripheral circuits dominate latency and power. (d) Impact of array size reduction on delay and power consumption. As the size of individual arrays decreases (e.g., replacing one 1024 × 1024 array with sixteen 256 × 256 arrays), delay remains stable while power consumption is significantly increased.

precision extension strategies. Leveraging the ability of NOR flash cells to be individually tuned and controlled, non-essential cells (those not marked with red shading) can be omitted entirely. This allows all slices of the 16-bit data to be computed within a single array simultaneously, reducing both power consumption and delay by minimizing the number of peripheral circuits. Additionally, positive and negative values are processed separately, with the negative component (marked in gray) calculated after the positive component. This ensures full utilization of the array cells, as illustrated in Figure 5(b).

Extending this scheme to 64-bit high-precision computing maintains high accuracy while significantly improving latency and power efficiency. A 64-bit array requires sixteen 4-bit arrays, and pulses must be input sixteen times to ensure all 4-bit data are multiplied. In the traditional parallel array scheme, this necessitates sixteen peripheral circuits operating simultaneously, increasing hardware costs. In contrast, the parallel input scheme stores all 64-bit data in a single, larger array, allowing pulses to be input concurrently, thereby reducing pulse duration to 1/16 of that required in the traditional scheme. While the number of shift-adders and analog-to-digital converters (ADCs) may slightly exceed 1/16 of the parallel array scheme, the overall cost is significantly reduced, as summarized in Figure 5(c). However, the parallel input scheme requires larger arrays, necessitating simulations to evaluate its overall performance.

Peripheral circuits play a critical role in the overall performance of CIM systems [26]. To assess their impact, we estimated the proportions of total area, latency, and power consumption for various array sizes and peripheral circuit configurations. The system-level NeuroSim simulation is shown in Figures 6(a)–(c), where the successive-approximation-register (SAR) ADC is adopted and the technology node is defined as 14 nm. For a 256 × 256 array, the shift-adder dominates latency, while the ADC accounts for approximately 98% of energy consumption. This highlights that peripheral circuits are the primary contributors to latency and energy costs in smaller arrays. Simulations for larger array sizes (Figure 6(d)) reveal that increasing the array size while maintaining a fixed total array area enhances throughput and energy efficiency. This is achieved by reducing the number of ADCs and other peripheral circuits in larger CIM arrays. Consequently, the robust reliability of large flash memory arrays ensures high-precision computing, while the optimized parallel input scheme further enhances power efficiency.

5 Conclusion

In summary, we have introduced a novel parallel input method designed for 64-bit high-precision computing in large-scale flash memory arrays. The proposed approach effectively enhances the parallelism of data input, thereby accelerating CIM operations. By minimizing the need for complex peripheral circuits, the design achieves significant reductions in latency and power consumption. Furthermore, the flash memory exhibits strong reliability in terms of endurance and retention, ensuring stable high-precision performance. Comprehensive evaluations confirm the advantages of the proposed method in terms of area efficiency, throughput, and energy efficiency, making it a promising solution for next-generation high-speed CIM architectures.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 62034006, U23B2040, 92264201), Natural Science Foundation of Shandong Province (Grant Nos. ZR2025ZD41, ZR2023LZH007), and MIND Project (Grant No. MINDXZ202407).

References

1 Sebastian A, Gallo M L, Khaddam-Aljameh R, et al. Memory devices and applications for in-memory computing. Nat Nanotechnol, 2020, 15: 529–544
2 Yao P, Wu H, Gao B, et al. Fully hardware-implemented memristor convolutional neural network. Nature, 2020, 577: 641–646
3 Sun C, Zhang Y, Liu J, et al. First demonstration of BEOL-compatible ferroelectric TCAM featuring a-IGZO Fe-TFTs with large memory window of 2.9 V, scaled channel length of 40 nm, and high endurance of 10^8 cycles. In: Proceedings of the 2021 Symposium on VLSI Technology, 2021. 1–2
4 Shin J H, Jeong Y J, Zidan M A, et al. Hardware acceleration of simulated annealing of spin glass by RRAM crossbar array. In: Proceedings of the 2018 IEEE International Electron Devices Meeting (IEDM), 2018
5 Fisher M, Nocedal J, Trémolet Y, et al. Data assimilation in weather forecasting: a case study in PDE-constrained optimization. Optim Eng, 2009, 10: 409–426
6 Ara A, Khan N A, Razzaq O A, et al. Wavelets optimization method for evaluation of fractional partial differential equations: an application to financial modeling. In: Proceedings of the Advances in Difference Equations, 2018. 1–13
7 Karimzadeh F, Yoon J H, Raychowdhury A. BitS-Net: bit-sparse deep neural network for energy-efficient RRAM-based compute-in-memory. IEEE Trans Circuits Syst I, 2022, 69: 1952–1961
8 Chen T, Jacob B, Teyuh C, et al. An SRAM-based accelerator for solving partial differential equations. In: Proceedings of the 2019 IEEE Custom Integrated Circuits Conference (CICC), 2019. 1–4
9 Feng Y, Sun Z, Wang C, et al. An efficient flash-based computing-in-memory (CIM) demonstration of high-precision (32-bit) nonlinear partial differential equation (PDE) solver with ultra-high endurance and reliability. IEEE Trans Circuits Syst I, 2025, 72: 3247–3257
10 Yi W, Mo K, Wang W, et al. RDCIM: RISC-V supported full-digital computing-in-memory processor with high energy efficiency and low area overhead. IEEE Trans Circuits Syst I, 2024, 71: 1719–1732
11 Bauer P, Thorpe A, Brunet G. The quiet revolution of numerical weather prediction. Nature, 2015, 525: 47–55
12 Zhou J, Kim K, Lu W. Crossbar RRAM arrays: selector device requirements during read operation. IEEE Trans Electron Devices, 2014, 61: 1369–1376
13 Lue H T, Hsu P K, Wei M L, et al. Optimal design methods to transform 3D NAND flash into a high-density, high-bandwidth and low-power nonvolatile computing in memory (nvCIM) accelerator for deep-learning neural networks (DNN). In: Proceedings of the IEEE International Electron Devices Meeting (IEDM), 2019
14 Guo X, Bayat F M, Bavandpour M, et al. Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology. In: Proceedings of the IEEE International Electron Devices Meeting (IEDM), 2017
15 Mach S. Floating-Point Architectures for Energy-Efficient Transprecision Computing. Konstanz: ETH Zurich, 2021
16 Soundari D V, Shanker Ganesh M K, Raman I, et al. Enhancing network-on-chip performance by 32-bit RISC processor based on power and area efficiency. Mater Today-Proc, 2021, 45: 2713–2720
17 Le G M, Sebastian A, Mathis R, et al. Mixed-precision in-memory computing. Nat Electron, 2018, 1: 246–253
18 Wen T, Hsu H, Khwa W, et al. 34.8 A 22 nm 16 Mb floating-point ReRAM compute-in-memory macro with 31.2 TFLOPS/W for AI edge devices. In: Proceedings of the 2024 IEEE International Solid-State Circuits Conference (ISSCC), 2024. 580–582
19 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
20 Long Y, Lee E, Kim D, et al. Flex-PIM: a ferroelectric FET based vector matrix multiplication engine with dynamical bitwidth and floating point precision. In: Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), 2020. 1–8
21 Flores-Vergara A, García-Guerrero E E, Inzunza-González E, et al. Implementing a chaotic cryptosystem in a 64-bit embedded system by using multiple-precision arithmetic. Nonlinear Dyn, 2019, 96: 497–516
22 Yang X, Yan X, Xing Z, et al. A 64-bit stream processor architecture for scientific applications. SIGARCH Comput Archit News, 2007, 35: 210–219
23 Mallasén D, Del Barrio A A, Prieto-Matias M. Big-PERCIVAL: exploring the native use of 64-bit posit arithmetic in scientific computing. IEEE Trans Comput, 2024, 73: 1472–1485
24 Ielmini D, Ghetti A, Spinelli A S, et al. A study of hot-hole injection during programming drain disturb in flash memories. IEEE Trans Electron Devices, 2006, 53: 668–676
25 Feng Y, Chen B, Liu J, et al. Design-technology co-optimizations (DTCO) for general-purpose computing in-memory based on 55 nm NOR flash technology. In: Proceedings of the 2021 IEEE International Electron Devices Meeting (IEDM), 2021
26 Chen P, Peng X C, Yu S M. NeuroSim+: an integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures. In: Proceedings of the IEEE International Electron Devices Meeting (IEDM), 2017