SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2025, Vol. 68, Iss. 2, 122406:1–122406:16
https://doi.org/10.1007/s11432-024-4240-x

Optimizing hardware-software co-design based on non-ideality in memristor crossbars for in-memory computing

Pinfeng JIANG[1], Danzhe SONG[1], Menghua HUANG[1], Fan YANG[1], Letian WANG[1], Pan LIU[1], Xiangshui MIAO[1,2] & Xingsheng WANG[1,2*]

[1]*School of Integrated Circuits, Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*Hubei Yangtze Memory Labs, Wuhan 450205, China*

Received 3 April 2024/Revised 30 June 2024/Accepted 27 November 2024/Published online 14 January 2025

Abstract The memristor crossbar, with its exceptionally high storage density and parallelism, enables efficient vector matrix multiplication (VMM), significantly improving data throughput and computational efficiency. However, its analog computing is vulnerable to issues like IR-drop, device-to-device (D2D) variation, and stuck-at-fault (SAF), leading to a substantial decrease in the inference accuracy of neural networks deployed on crossbars. This work presents a hardware-software co-design approach tailored to deal with memristor crossbar non-ideality. We introduce an end-to-end functional array simulator (FAST) for precise and ultra fast end-to-end training, mapping, and evaluation of neural networks on the memristor crossbar. Utilizing the sparsity of the memristor crossbar coefficient matrix, it achieves simulation with low storage and computational resource requirements, dynamically selecting the optimal solution to complete the process. It can also precisely simulate the impact of non-ideal effects such as IR-drop, retention, variation, SAF, and AD/DA precision. Using FAST, we assess memristor crossbar matrix operations under non-ideal conditions, identifying the max throughput and the most energy-efficient crossbar configurations. Additionally, we propose a comparator-based activation function modulation (CAFM) scheme and its corresponding hardware architecture with programmable activation function circuits to address the IR-drop issue, enabling low power and area overheads, resulting in the recovery of neural network accuracy by 54% or more. This is validated within FAST, demonstrating the success of our hardware-software optimization co-design.

Keywords memristor crossbar, IR-drop, neural network, activation function, hardware-software co-design

Citation Jiang P F, Song D Z, Huang M H, et al. Optimizing hardware-software co-design based on non-ideality in memristor crossbars for in-memory computing. Sci China Inf Sci, 2025, 68(2): 122406, https://doi.org/10.1007/s11432-024-4240-x

1 Introduction

Inspired by neuromorphic intelligence, advanced neural network (NN) algorithms have emerged. Leveraging the parallelism, fault tolerance, and efficiency of NN has become a growing trend for handling big data. Resistive random-access-memory (RRAM, also called memrister), as a non-volatile device capable of retaining multiple states within a single device, is an essential component of high-density storage arrays [1–3]. It enables highly efficient storage and in-memory computing, making it a strong contender for addressing the "memory wall" issue. However, in large-scale memristor crossbar, analog in-memory computing is vulnerable to non-ideal effects like IR-drop, retention, device-to-device (D2D) variation, and stuck-at-fault (SAF) [4–7]. These factors significantly impact the accuracy of NN deployed on crossbars, posing a substantial challenge to hardware performance.

To mitigate these significant impacts, a comprehensive evaluation of non-ideality is essential before deploying NN [8–13]. Currently, prominent memristor system simulators such as PIM-HLS, NeuroSim, PytorX and SySCIM are available. PIM-HLS facilitates NN simulation based on a process-in-memory (PIM) array, although its primary purpose is to accelerate the evaluation of PIM system architectures, and it does not offer support for simulating non-ideal effects [10]. NeuroSim provides simulation support for memristor systems from the circuit to algorithm levels, with adaptations for mainstream NNs [11]. However, it does not comprehensively simulate the non-idealities at the array level. PytorX can simulate

* Corresponding author (email: xswang@hust.edu.cn)

the impact of various non-ideal effects in memristor crossbar configurations on NN accuracy [12]. Nevertheless, its computational demands are significant, imposing high requirements on computing devices, and it operates at extremely slow simulation speeds. SySCIM employs SystemC and SystemC-AMS methods to architect a simulator capable of highly detailed non-ideal behavior modeling [13]. However, a notable limitation arises with regard to the computational speed during the simulation of NNs.

As the crossbar size continues to grow, the impact of IR-drop becomes increasingly severe and a key factor affecting the precision of NN [14–17]. In recent work, many solutions have been proposed in both hardware and software domains to address this issue. Belkin et al. [16] have conducted in-situ training of memristor cells through repeated writing in memristor crossbars, demanding high endurance from the devices. Han et al. [17] introduced a weight mapping correction method, which involves weight adjustments prior to network mapping. This method requires iterative calculations to modify weights, consuming significant computational resources. He et al. [12] modeled the distortion of the crossbar output current, treating it as noise injected into the training process to enhance the robustness of line resistance. However, this method requires recalibration and remodeling when there are changes in crossbar or line resistance parameters, thus lacking universality. Shin et al. [18] have conducted a novel reordering weight mapping scheme that effectively clusters zero weights to address the IR-drop issue, but it requirs adding a large number of storage units, which increases the area and transmission overhead. Amin et al. [19] proposed a crossbar partitioning scheme in the full analog domain to mitigate the impact of IR-drop, utilizing memristors as part of the activation function, which did not consider the non-ideal endurance of memristors.

Therefore, it is essential to propose a fast array-level simulator that can accurately simulate non-idealities and present a solution that uses fewer resources to address the IR-drop issue. In this work, we present a hardware-software co-design approach based on non-ideality in memristor crossbars. Firstly, we propose an end-to-end functional array simulator (FAST), which utilizes the characteristics of sparse matrices to achieve fast simulation of large-scale memristor crossbar, and evaluate the impact of various non-idealities. Additionally, we propose an activation function modulation (AFM) optimization and its corresponding hardware architecture to address the IR-drop issue, which can significantly improve NN inference accuracy with lower power and area overheads. The key contributions are as follows:

• Propose FAST, an end-to-end FAST that utilizes sparse matrix computation optimization to simulate large-scale memristor crossbar faster (Section 3).

• Assess the impact of non-idealities in memristor crossbars using FAST, exploring max throughput and energy efficiency under various non-ideal factors to determine the most resource-efficient crossbar configurations (Section 4).

• Propose a comparator-based activation function modulation (CAFM) scheme to address IR-drop issues (Section 5).

• Present corresponding hardware architecture and programmable activation function circuits, achieving hardware-software co-optimization design for the memristor crossbar (Section 6).

2 Preliminaries

2.1 Memristor crossbar

In this subsection, we will explain the working principles of memristor compute-in-memory (CIM) operations. A typical 1T1R memristor crossbar is illustrated in Figure 1. Each memristor cell is connected to wordlines (WLs), bitlines (BLs) and sourcelines (SLs). Vector matrix multiplication (VMM) in the analog domain is achieved by simultaneously activating WLs and collecting currents from SLs [20, 21]. For example, to perform a $1 \times M$ vector multiplication with an $M \times N$ matrix, the vector is encoded as input voltages (V_i), and the matrix is encoded as conductance values (G_{ij}). The output current obtained from the jth SL represents the cumulative currents from all cells in the same column: $I_j = \sum_{i=1}^{M} V_i \times G_{ij}$. NN weights can have positive and negative values, but memristor cells cannot represent negative values. Therefore, a dedicated structure is needed to store negative weights. In this work, a differential structure divides the weights into two crossbars, each responsible for storing positive and negative weights separately. Other structures include using a single crossbar with one column for positive weights and another for negative weights, or employing 2T2R cells in the same column to represent both positive and negative weights simultaneously. Here the jth column output currents from the two crossbars are subtracted using

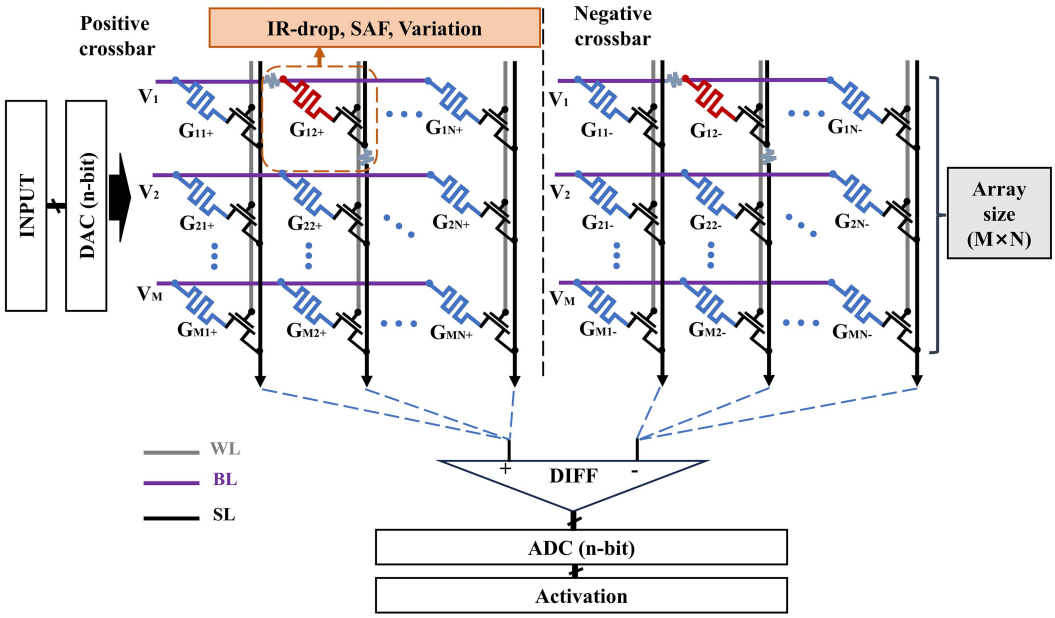

Figure 1 (Color online) Hardware implementation of single $M \times N$ memristor crossbar pair (positive and negative crossbars) with non-ideality.

a differential structure, and the final output is represented as $I_j = \sum_{i=1}^{M} V_i \times (G_{ij+} - G_{ij-})$. The N columns of output currents constitute the output vector of VMM.

Typically, memristor crossbars require peripheral circuits such as analog-to-digital converter (ADC) and digital-to-analog converter (DAC). DACs convert digital signals into analog voltage inputs, while ADCs convert analog currents into digital signal outputs, facilitating system-level signal transmission. Due to the potentially large number of weights in a single layer of an NN, such as in a fully connected layer, it is not feasible to deploy all weights in a single crossbar. Instead, the weights are distributed across multiple crossbars, and these crossbars simultaneously perform VMM operations. The digital output signals from these crossbars are then transmitted and accumulated through the control circuit, ultimately reaching the neuron module for activation. This process is repeated layer by layer, completing the inference process of the NN.

2.2 Non-ideality

Within memristor crossbars, various non-idealities impact performance, including IR-drop due to parasitic resistance of the lines, SAF effects in memristor cells, finite states in memristor conductance, retention and variation [22–26]. Additionally, the precision of ADC and DAC circuits also affects the results [27].

Parasitic resistance widely exists in memristor crossbar, with the impact of line resistance being predominant. As shown in Figure 1, the line resistance bridging between two memristor cells significantly affects current flow due to IR-drop and the sneak path. It also alters the circuit's topology, making the voltage distribution problem more complex. With advanced technology and increased integration density, the increase in interconnect line resistance and array size exacerbates the IR-drop issue, becoming a primary factor in the decline of computational accuracy.

Due to the limitations of immature manufacturing technologies, manufacturing yield remains a major issue in memristor-based neuromorphic systems. One important and common fault that can occur is SAF, where faulty devices become stuck in either a high-resistance state (HRS) or a low-resistance state (LRS). When the proportion of SAF is too high, the output current of the memristor crossbar will suffer severe distortion.

Since the randomness of oxygen vacancy generation through oxygen ion trapping and de-trapping, memristors exhibit stochastic resistance states, leading to D2D variation rather than maintaining a stable conductance level. This is a significant problem in memristor arrays, as it causes each memristor cell representing the same weight to have different conductance values, resulting in errors in the crossbar output and severely impacting the accuracy of offline-trained NNs.

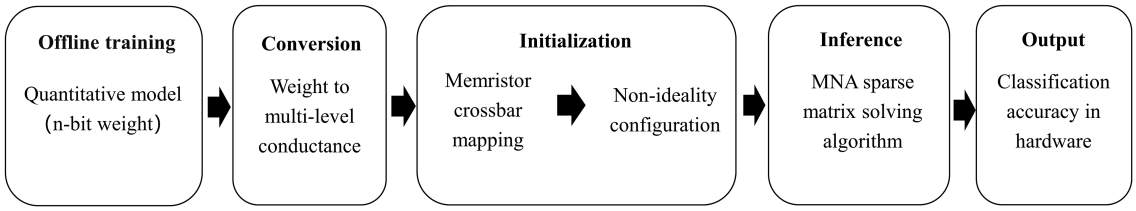

Figure 2 Workflow of end-to-end FAST.

Reliability is crucial for neuromorphic systems based on non-volatile devices. Retention is a reliability metric for non-volatile devices, indicating the ability of the device to maintain the current weight state effectively. Inference requires multiple stable conductance, and if the conductance occurs drift or overlap, it could lead to a significant decrease in computational accuracy.

3 End-to-end FAST

3.1 FAST overview

To comprehensively analyze the impact of various non-ideal effects on NN computations based on memristor crossbars, we have developed a comprehensive end-to-end FAST. This simulator allows us to incorporate the influence of multiple non-ideal effects while performing NN inference on memristor crossbars. The specific operational workflow of this simulator is illustrated in Figure 2.

The first step involves offline training to obtain network model files for subsequent inference processes. In this process, conductance states can be determined based on measured real memristor conductance data. This data is used to complete the quantization process in the training phase, ensuring that the weight values of the resulting model correspond precisely to the actual conductance data. This approach enables a more accurate inference process. In cases where measured conductance data is unavailable, weight quantization can also be accomplished in this step based on the specified weight bit-width.

The second step involves the conversion of weights into multi-level conductance. To correctly transform positive and negative weights into the conductance of two separate memristor crossbars, we employ (1) and (2) to facilitate this process [28]:

$$G_+ = \begin{cases} w \times (G_{\max} - G_{\min}) + G_{\min}, & w > 0, \\ G_{\min}, & w \leqslant 0; \end{cases} \tag{1}$$

$$G_- = \begin{cases} G_{\min}, & w \geqslant 0, \\ -w \times (G_{\max} - G_{\min}) + G_{\min}, & w < 0. \end{cases} \tag{2}$$

Here, G_+ and G_- represent the conductance values for the positive and negative weight crossbars, respectively. w corresponds to the weight, and $G_{\max}$ and $G_{\min}$ denote the maximum and minimum conductance values of the memristor cells.

The third step involves the initialization of relevant parameters in the memristor crossbar. Since the number of cells in each crossbar might not suffice to accommodate the weights for every layer of a deep NN, an effective weight mapping scheme is needed to distribute the weights of each layer across multiple crossbars. In this step, we transform the weights of each layer into two four-dimensional conductance matrices, denoted as $\{N_{\text{row}}, N_{\text{col}}, A_r, A_c\}$, corresponding to G_+ and G_-. Here, $N_{\text{row}} \times N_{\text{col}}$ represents the number of crossbars required for the weights of that layer. Additionally, we configure the parameters related to non-ideal effects, allowing for the adjustment of line resistance, SAF impact, D2D variation, and the precision of ADC/DAC circuits.

The fourth step is to complete NN inference process. We use the modified nodal analysis algorithm (MNA) [29] to construct the coefficient matrix using the conductance matrix obtained in the previous step. This matrix is then combined with the voltage vector formed by the input encoding to complete the solution process. Since the conductance coefficient matrix is sparse, the entire solution process can be optimized for sparsity and accelerated using CUDA with NVIDIA GPU, which we will explain in detail later in the text. As a result, this inference process offers faster solving speeds compared to traditional methods.

Table 1 Comparison of memristor simulator.

	MNSIM [30]	NeuroSim [11]	SySCIM [13]	PytorX [12]	FAST
Device model	✓	✓	✓	✓	✓
Parasitics	✓	✓	✓	✓	✓
Variation	✓	✓	✓	✓	✓
Retention	–	✓	–	–	✓
SAF	–	–	✓	✓	✓
Precise IR-drop evaluation	–	–	–	✓	✓
Finite conductance evaluation	–	–	–	–	✓
Sparse matrix acceleration	–	–	–	–	✓
Hardware-software co-design	–	–	–	–	✓

The final step aims to achieve inference with an accuracy roughly equivalent to that achieved through hardware implementation of NN in memristor crossbars. This guides subsequent hardware mapping and circuit design.

Table 1 [11–13, 30] summarizes the important features of the related simulators and illustrates how FAST differentiates from them. MNSIM [30] proposes a coarse-grained estimation model for memristor crossbars, allowing performance evaluation by modifying top-level configuration file parameters, which is more inclined towards top-level architecture simulation. NeuroSim [11] and SySCIM [13] can simulate various non-idealities in memristor crossbars with finer granularity, but they use approximate calculations for IR-drop caused by line resistance, resulting in inaccurate computational results. PytorX [12] can also achieve detailed non-ideality simulation and accurately simulate IR-drop, but it runs very slowly and requires significant computational resources. FAST utilizes a sparse matrix method to achieve accurate and fast evaluation of IR-drop caused by line resistance. It also supports the entire process from measured conductance data and offline NN training to inference. The finite conductance-aware evaluation helps improve NN accuracy. Additionally, FAST supports hardware-software co-design, allowing for the simulation and validation of hardware optimization effects directly in software, thereby guiding hardware design and improving computational accuracy.

3.2 Sparse matrix optimization for weights

In this subsection, we will illustrate the use of the MNA algorithm for a 2×2 memristor crossbar. As shown in Figure 3, following the MNA algorithm requirements, we select the reference node and name the nodes in the crossbar sequentially. After identifying the current sources and listing the current equations for each node and the voltage equations for the voltage sources, we can transform the equations into a matrix multiplication form $Ax = z$. Since the resistance of the transistor in the 1T1R cell is significantly different from that of the memristor, it can be considered negligible.

Similarly, FAST can also simulate 0T1R and 1S1R arrays. Since the equation in Figure 3 ignores the resistance of the transistor, the results for arrays composed of 0T1R cells are consistent with those of 1T1R cells. For the 1S1R array, since all cells need to conduct during the VMM operation, it is necessary to evaluate the LRS of the selector. Then, calculate the current passing through both memristor and selector to obtain the final result. For example, assuming the LRS of the selector is $R_{\text{S-LRS}}$, as shown in node 10 of Figure 3(b), R_4 needs to be modified to $R_4 + R_{\text{S-LRS}}$. Similar modifications would be applied to other nodes, allowing us to continue using the MNA method to solve the circuit equations.

The coefficient matrix A is sparse, and for a memristor crossbar size of 256×256, the dimension of A will reach 130 thousand, making it a very large matrix. Storing A in dense matrix format would consume a significant amount of storage resources and could even lead to insufficient GPU memory.

Here we use the linked list matrix (LIL) [31] data structure to store the sparse matrix A, which stores the non-zero elements and their positions in the matrix as linked lists. The LIL uses two linked lists to represent the rows and columns. The nodes in each row's linked list are arranged in column index order, while the nodes in each column's linked list are arranged in row index order. By utilizing the properties of linked lists, non-zero elements can be efficiently added, making it suitable for generating and storing matrix A. As shown in Figure 4, we save the matrix A for different array sizes in both dense matrix and LIL formats as numpy files. It can be seen that the file size saved in the LIL sparse matrix format is much smaller than that of the dense matrix format. For example, when the crossbar size is 128×128, the dimension of the square matrix A reaches 30 thousand. At this size, using the LIL format requires

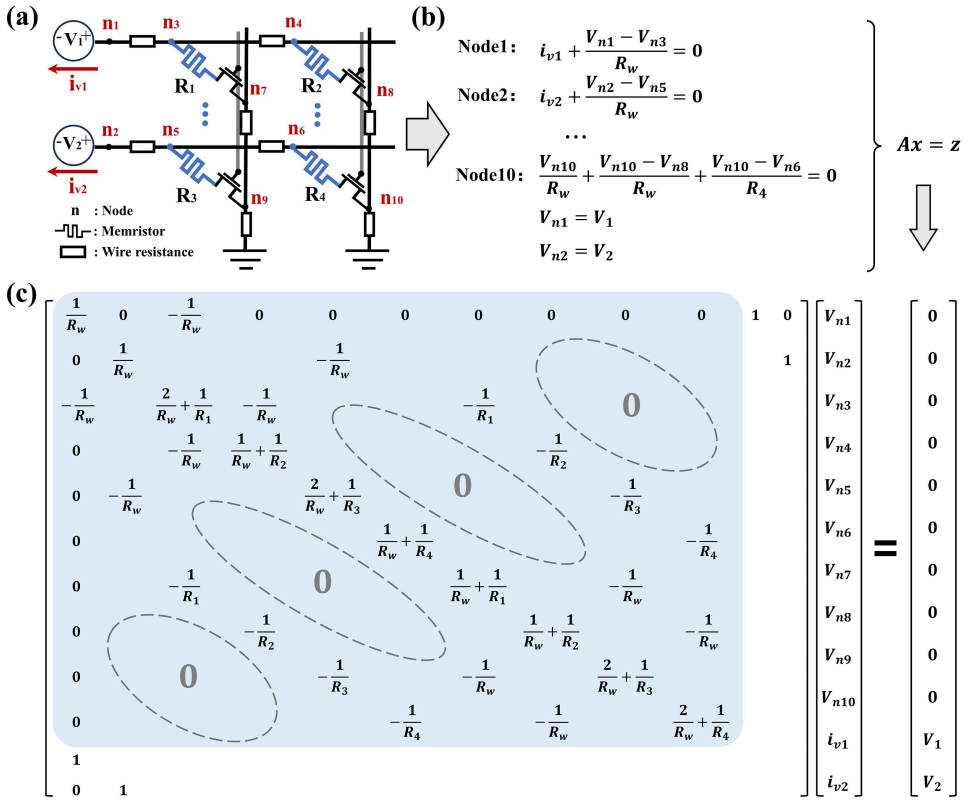

Figure 3 (Color online) Schematic diagram of the MNA algorithm. (a) 2 × 2 memristor crossbar; (b) current and voltage equations; (c) matrix multiplication $Ax = z$.

only 1.63 MB of file size to save the entire sparse matrix, greatly saving storage resources and facilitating subsequent inference processes.

3.3 Dynamic selection of solution methods

To accurately assess the impact of IR-drop, PytorX employs the method of matrix inversion to solve the linear equations derived from the MNA algorithm [12]. However, when the array size becomes too large, even a simple LeNet-5 implementation can take several days to converge on a workstation equipped with four Nvidia TitanX GPUs. This approach leads to intensive matrix operations, consuming a significant amount of GPU memory and time to complete the inference process, making it impractical. The process of matrix inversion consumes a large amount of resources for large-scale matrices. As shown in Figure 5, when the crossbar size reaches 256×256, the computing device is unable to perform the calculations. Therefore, improving the efficiency of solving linear equations can enhance computational efficiency.

LU decomposition is one of the commonly used optimization methods for solving linear equations [32]. It decomposes the original matrix into two triangular matrices, which are easier to invert. This significantly reduces the dimension of the inverse matrix, making the solution of linear equations simpler, more efficient, and widely applicable. We used LU decomposition to solve the linear equations in Figure 3(b), and the results are shown in Figure 5. When the array size is 128 × 128 or smaller, the solution is very fast, taking only 0.25 s. However, as the array size increases to 512 × 512, the dimension of the conductance matrix reaches 500k, and the solution speed significantly decreases, taking 13.73 s to complete the calculation. When the array size is too large, LU decomposition still requires a significant amount of resources to solve. Therefore, LU decomposition is still not a good choice for large-scale arrays.

Taking a 2 × 2 memristor crossbar as an example, we analyze the conductance matrix (matrix A) in Figure 3(c). This matrix is a real symmetric non-positive definite matrix and is extremely sparse. The conjugate gradient (CG) method is an excellent algorithm for solving large matrices [33], but it is only suitable for linear systems of equations where the coefficient matrix is a symmetric positive definite matrix, which is not the case here. The minimal residual (MINRES) method [34] is an iterative algorithm suitable for solving linear systems of equations with real symmetric positive definite or approximately

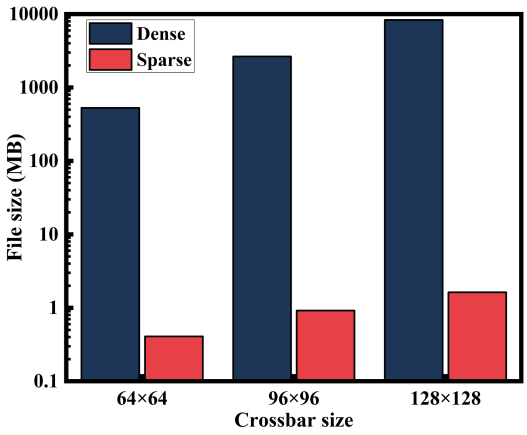

Figure 4 (Color online) Storage file sizes for different crossbar sizes.

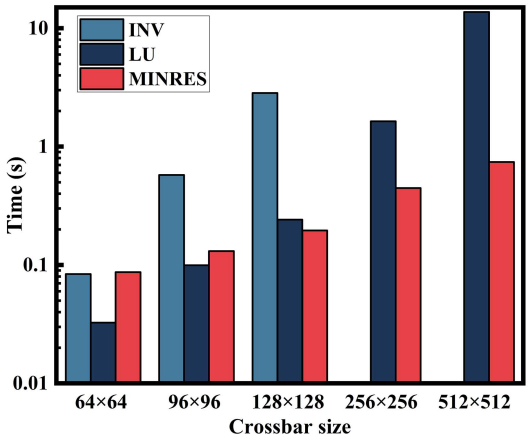

Figure 5 (Color online) Solving time of matrix A for different crossbar sizes using different solution methods (computing device: NVIDIA GeForce RTX 4090 24 GB).

Algorithm 1 MINRES algorithm for solving memristor crossbar equations.

Input: Matrix A, vector z, initial guess x_0, tolerance tol, maximum iterations $N_{\max}$.
Output: Solution x.

$r_0 = z - A \times x_0$;
$p_0 = r_0$;
$q_0 = A \times p_0$;
$\alpha_0 = \frac{\langle r_0, q_0 \rangle}{\langle q_0, q_0 \rangle}$;
for $k = 1$ **to** $N_{\max}$ **do**
  $r_k = r_{k-1} - \alpha \times q_{k-1}$;
  $\beta_k = \frac{\langle r_k, A \times p_{k-1} \rangle}{\langle p_{k-1}, A \times p_{k-1} \rangle}$;
  $p_k = r_k + \beta_k \times p_{k-1}$;
  $q_k = A \times p_k$;
  $\alpha_k = \frac{\langle r_k, q_k \rangle}{\langle q_k, q_k \rangle}$;
  $x_k = x_{k-1} + \alpha_k \times p_k$;
  if $\|r_k\| <$ tol **then**
    break
  end if
end for

symmetric positive definite coefficient matrices. MINRES only requires the coefficient matrix of the linear system of equations to be symmetric, not necessarily positive definite, making it highly suitable in this case. The process of using MINRES to solve the memristor crossbar equations is shown in Algorithm 1. Compared to other solution methods, MINRES does not require storing the entire coefficient matrix. Instead, it uses matrix-vector multiplication to achieve iteration, which reduces memory consumption. Additionally, MINRES is efficient in solving large sparse matrices, thereby speeding up the inference process of the FAST simulator.

Figure 5 also shows the speed of MINRES in solving the memristor crossbar equations. It can be observed that when the array size is smaller than 96×96, the solving time of MINRES is longer than that of LU decomposition. However, when the array size reaches 512×512, the solving speed of MINRES is $18.55\times$ faster than LU decomposition, indicating that MINRES is highly suitable for computing extremely large-scale memristive array equations. Therefore, FAST integrates the advantages of various solution methods and can dynamically select the solution method for solving memristor crossbar equations based on the crossbar size requirements, thereby achieving faster simulation speeds.

Since some simulators do not evaluate the impact of IR-drop that alters the voltage distribution of the entire resistive network, we only compare FAST with SySCIM [13] and PytorX [12] under different memristor crossbar sizes, as shown in Table 2 [12,13]. This includes the runtime required for each crossbar and the error evaluation of the output current for each column in the crossbar compared to the accurate output current obtained from HSPICE, represented by normalized root mean squared error (NRMSE). It can be seen that SySCIM has a much higher NRMSE compared to PytorX and FAST. This is because SySCIM uses an approximate method to handle the IR-drop issue, resulting in much more accuracy loss, whereas PytorX and FAST run based on Kirchhoff's laws, achieving higher accuracy. Additionally,

Table 2 Comparison of simulation accuracy and runtime.

	SySCIM [13]		PytorX [12]		FAST		
Crossbar size	64×64	128×128	64×64	128×128	64×64	128×128	1024×1024
NRMSE	0.021	0.035	0.92×10^{-4}	4.77×10^{-4}	1.01×10^{-4}	2.53×10^{-4}	2.44×10^{-3}
Runtime (s)	1779	4685	0.084	2.836	0.033	0.197	2.147

Table 3 Parameters of memristor crossbar system.

Symbol	Description	Value
V_{ref} (V)	Max input coding voltage	0.3
$G_{\min}, G_{\max}$ (μS)	Memristor min, max conductance	20, 100
N_B (bit)	Resolution of ADC/DAC	8
C_{size}	Crossbar dimension	64

SySCIM's runtime is much longer than the others because it does not utilize GPU acceleration and performs detailed simulations of other non-idealities. When the crossbar size is 64×64, FAST achieves $2\times$ speedups compared with PytorX, and when the crossbar size is 128×128, FAST becomes $14.5\times$ faster than PytorX. This is due to FAST utilizing a sparse matrix acceleration method that dynamically selects LU decomposition and MINRES methods to achieve optimal acceleration at different crossbar sizes, with the acceleration being more pronounced as the crossbar size increases. Especially, FAST still maintains fast runtime and preserves simulation accuracy at ultra large scale crossbar size of 1024×1024.

4 Analysis of memristor non-ideal effects

In this section, we will analyze the impact of non-ideal effects on NN inference accuracy when deployed on memristor crossbars. Subsequently, we will explore the max throughput and energy efficiency under various non-ideal effects. To evaluate these non-ideal effects, we will utilize the LeNet5 network with a Sigmoid activation function for the recognition task on the MNIST dataset as an illustrative example, while employing other parameters as outlined in Table 3.

4.1 IR-drop

In an ideal scenario, the voltage applied to the crossbar inputs is transmitted to the top electrode of the memristor cell without any loss, while the bottom electrode is clamped at 0 V. However, in practical situations, line resistance cannot be ignored, resulting in varying electrode voltages for each memristor cell. We employed FAST to calculate and analyze the voltage distribution of 256×256 crossbar at the top and bottom electrodes of memristor cells affected by IR-drop, as illustrated in Figure 6(a). When the input voltages are applied on BLs, it is evident that the top electrodes farther from the input end undergo more significant input voltage losses, while the bottom electrodes farther from the current output end attain higher voltage.

Furthermore, FAST can assess the line resistance impact on devices with different resistance on/off ratios and ranges, as shown in Figures 6(b) and (c). In this case, we randomly initialize the conductance of each cell in the crossbar within the set resistance on/off ratio and range, and apply a constant voltage to BLs while grounding SLs, to statistically analyze the normalized voltage distribution of each cell in the crossbar with different line resistances. Figure 6(b) demonstrates the HfO_x/AlO_y superlattice-like memristor with $R_{\text{on}}/R_{\text{off}} = 10/50\,\text{k}\Omega$ [28], and Figure 6(c) depicts the $SrFeO_x$ memristor with $R_{\text{on}}/R_{\text{off}} = 5/20\ \text{k}\Omega$ [35]. Notably, $SrFeO_x$ memristor exhibits greater sensitivity to IR-drop, with memristor cells in the same position in the $SrFeO_x$ crossbar undergoing more severe voltage losses compared to the HfO_x/AlO_y crossbar under the same line resistance conditions, as the line resistance consumes relatively large division voltage compared to small $R_{\text{on}}/R_{\text{off}}$ memristor. This functionality can provide valuable guidance for the development of memristor devices.

4.2 Multi-conductance, retention and D2D variation

Due to the multi-level characteristics of memristors, memristor crossbars offer a higher storage density compared to SRAM arrays. FAST can transform the memristor conductance obtained from experiments into weight with a specific ratio during offline training, resulting in a quantized model. These weights

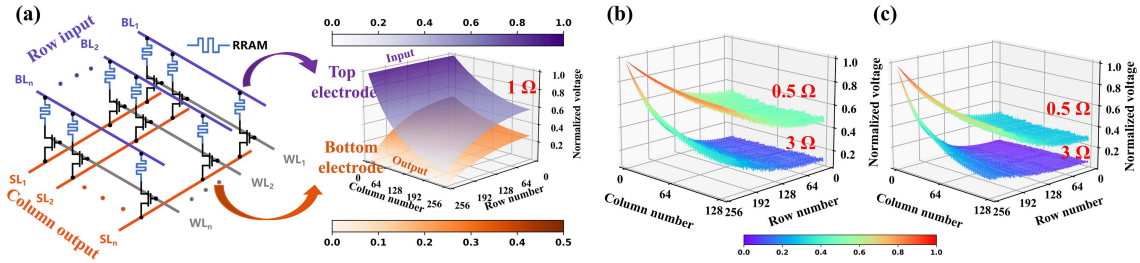

Figure 6 (Color online) (a) 256 × 256 memristor crossbar and the voltage distributions of top/bottom electrodes with 1 Ω line resistance between neighbor cells. The voltage distribution of (b) HfO_x/AlO_y memristor and (c) $SrFeO_x$ memristor in 256 × 128 crossbar with line resistances of 0.5/3 Ω.

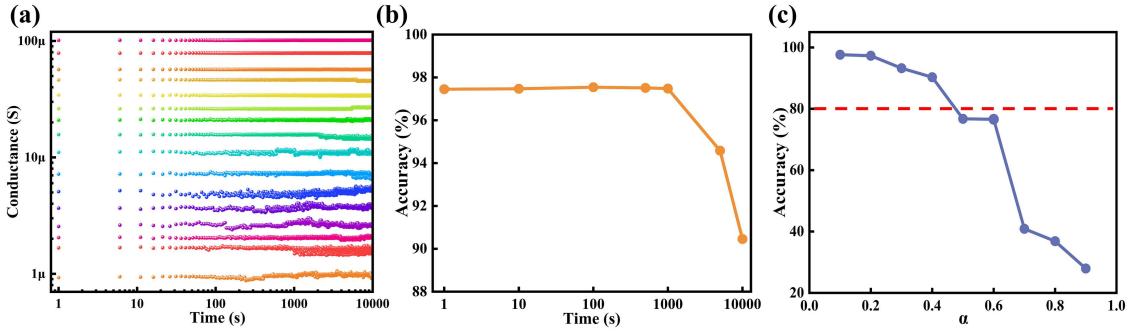

Figure 7 (Color online) (a) Retention properties for 16 conductance levels at 85°C; (b) accuracy affected by retention; (c) accuracy decreasing as α increases.

can then be converted back into conductance with minimal loss, effectively representing experimental data values. Precisely deploying this model in a memristor crossbar can enhance inference accuracy on the hardware. We utilized 16 levels of the conductance states obtained from experiments, as depicted in Figure 7(a), as the quantization reference during offline training.

Retention is also an important characteristic of memristors, referring to their ability to retain programmed states over a long period. FAST supports the simulation of retention, with data being either measured data or derived from formulas. Figure 7(a) shows the retention property of HfO_x/AlO_y memristors for 16 conductance levels at 85°C. FAST can analyze changes in conductance over time, modify the conductance of each cell based on these changes, and complete the entire inference process to obtain final accuracy, as shown in Figure 7(b). Due to the good retention property of these 16 conductance levels, the accuracy does not decrease significantly.

Additionally, FAST can simulate the retention of devices based on the conductance drift formula. The formula for modeling conductance drift behavior can be assumed as $G = G_0(t/t_0)^v$, where G_0 is the initial conductance, t is the retention time, v is the drift coefficient, and t_0 is the time constant assumed to be 1 s in this framework [11]. After modifying the conductance of each cell according to this formula, the inference process can be completed.

D2D variation is a significant concern in memristor crossbars, as it causes each memristor cell representing the same weight to exhibit different conductance values, leading to errors in crossbar output. Using the network model obtained through the above steps, the effects of D2D variation are introduced during programming the conductance. We assume that D2D variation follows a Gaussian distribution $N(0, \sigma^2)$, where $\sigma = G_{\min} \times \alpha$, with $G_{\min}$ being the minimum conductance in the memristor and α being a variable that can adjust the impact of variation. As shown in Figure 7(b), it is evident that when α exceeds 0.4, accuracy drops below 80%. This implies that a substantial impact on the accuracy of NN occurs when the standard deviation of conductance state variation exceeds $0.4 \times G_{\min}$. This observation can be used to inform the choice of conductance states.

4.3 SAF

SAF comes in two types: stuck-at-zero (SA0) and stuck-at-one (SA1). SA0 results in memristor cells being fixed at the lowest conductance ($G_{\min}$), while SA1 leads to memristor cells being fixed at the

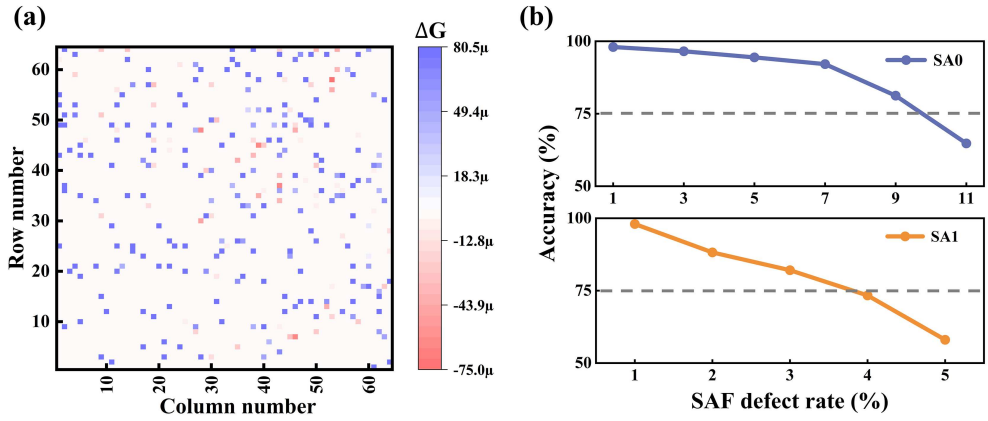

Figure 8 (Color online) (a) The conductance change in one of the memristor crossbars in the fully connected layer before and after SAF influence (the proportion of both SA0 and SA1 are set to 5%, $\triangle G = G_{\text{Ideal}} - G_{\text{SAF}}$); (b) the impact of SA0 and SA1 proportions on accuracy.

highest conductance ($G_{\max}$). Both of these faults can significantly impact the accuracy of NN [36]. FAST allows for the random setting of a proportion of memristor cells in each crossbar as SA0 or SA1, followed by the completion of the NN inference process. Figure 8(a) displays the conductance variation in one of the memristor crossbars in the intermediate layer before and after SAF influence. In this case, the proportion of both SA0 and SA1 are set to 5%, and $\triangle G$ represents the difference between the ideal conductance (G_{Ideal}) and the conductance affected by SAF (G_{SAF}). It can be observed from the figure that the majority of cells exhibit an increase in conductance. Figure 8(b) illustrates the change in NN accuracy under different SAF ratios, confirming this observation. When the proportion of SA1 exceeds 4%, accuracy drops to below 75%, while accuracy only drops to the same level when the proportion of SA0 exceeds 9%. This indicates that the impact of SA1 is significantly greater than that of SA0, primarily due to the weight matrix itself being a sparse matrix, and the introduction of non-zero elements profoundly distorts its computational results.

4.4 Max throughput and energy efficiency under various non-ideal effects

As the size of the crossbar increases, it can simultaneously perform larger matrix operations, allowing for a linear increase in the length of input vectors and, consequently, linear growth in throughput. However, when the number of columns is fixed, with an increase in the number of rows, as shown in Figure 6, the impact of IR-drop becomes more significant. Therefore, while increasing throughput, it is essential to consider the effects of IR-drop. Figure 9(a) illustrates the change in accuracy as the number of rows increases, with a fixed number of columns at 64, when the line resistance is 1 Ω. Additionally, we applied an AFM scheme proposed in [37] to mitigate the effects of IR-drop. This method employs

$$\text{Output} = f(B \times \text{input}). \tag{3}$$

Here f is the activation function, and B is an adjustable factor. During training and inference, increasing factor B helps recover the current that is reduced due to IR-drop, thereby enhancing accuracy. Figure 9(a) shows that without using AFM when the number of rows increases to 208, accuracy drops to below 90%. In contrast, when AFM is employed, accuracy remains above 90% even as the number of rows increases to 288, resulting in an approximately 1.38× increase in throughput compared to w/o AFM approach.

In memristor-based CIM systems, the power consumption of ADC/DAC constitutes a significant portion of the overall system power consumption, especially when high-precision ADC/DAC is employed, which consumes even more power. Thus, using low-precision ADC/DAC can save a substantial amount of energy. Utilizing lower precision for the weights allows for selecting more stable conductance states in memristor cells, reducing the impact of variation. Figure 9(b) depicts the relationship between weights, AD/DA precision, and accuracy. It is evident that accuracy significantly decreases only when the weight and AD/DA precision drop to below 3 bits. This suggests that optimizing the memristor cells and peripheral circuits can be achieved by using fewer stable conductance states and lower-precision ADC/DAC.

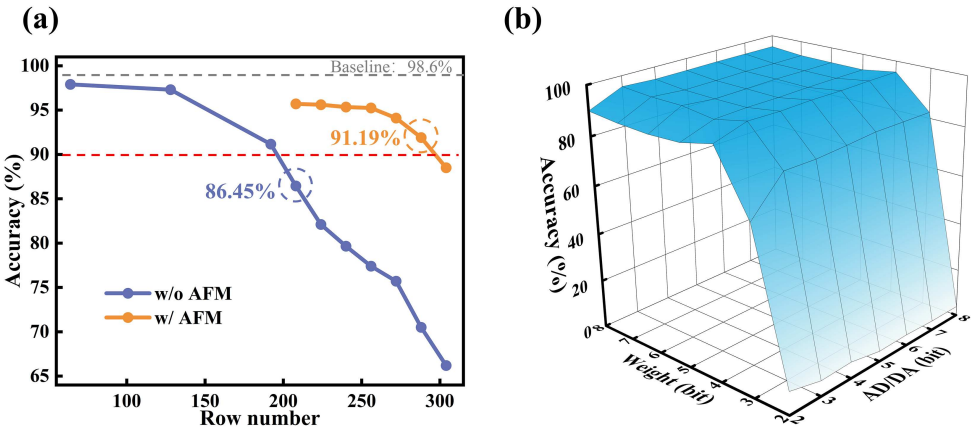

Figure 9 (Color online) (a) Under 1 Ω line resistance conditions, with the crossbar having 64 columns, inference accuracy decreases with an increase in the number of rows; (b) accuracy under different weight precision and AD/DA precision conditions.

5 CAFM

From Figure 6, it can be observed that due to line resistance, the cell voltages within the memristor crossbar differ significantly from the ideal scenario. Cells farther from the input suffer more severe IR-drop effects, resulting in a drastic reduction in the output current for almost all columns. The AFM optimization method proposed in [30] only applies to NN using Sigmoid and Tanh as activation functions. However, the ReLU activation function plays an important role in NN, making it essential to optimize its usage.

In this section, we conducted an analysis of LeNet-5 using ReLU as the activation function. For a well-trained NN, a few large weights usually correspond to specific input features and can activate neurons, while the outputs of other neurons remain close to zero. These strong connections with high conductance generate large currents in the memristor crossbar, and the corresponding cells are more significantly affected by IR-drop compared to other cells. We utilized FAST to statistically analyze the distribution of activation outputs. Figure 10(a) represents the output distribution for one of the linear layers, illustrating that outputs affected by line resistance suffer significant reduction, and neurons' outputs gather near zero. With increasing line resistance, large output currents responding to input features shrink more severely, leading to the input features almost disappearing, thereby severely affecting accuracy.

Unlike Sigmoid and Tanh activation functions, which have finite output ranges (Sigmoid ranges from 0 to 1, Tanh ranges from -1 to 1), the ideal ReLU activation function has no upper limit on its output range. However, in hardware implementations, due to the constraints of the reference voltages from ADC/DAC and the maximum read voltage of memristor cells, the ReLU activation function's max output is limited. To address this issue, we introduced the CAFM method, optimized using

$$\text{Output} = \max(f(B \times \text{Input}), V_C). \tag{4}$$

Here, Input is the output current from the memristor crossbar, $\max(\cdot)$ is a comparator that outputs the larger value, $f(\cdot)$ represents the ReLU activation function, B is a configurable coefficient, and V_C is also a configurable constant voltage. Due to the significant reduction in output current caused by IR-drop in the memristor crossbar, scaling up B can compensate for this effect. Although V_C is limited by the reference voltages or the maximum read voltage, it can be set to different values depending on the NN or crossbar size to achieve optimal accuracy. Since the main and minor features of the output differ significantly, this method can preserve the characteristics of the main features and avoid confusion with minor features while satisfying hardware specifications. From Figure 10(b), it can be seen that the output optimized by CAFM essentially restores the ideal distribution, greatly reducing the probability near zero and expanding the distribution range, with the main features clustered around 1. As shown in Figure 11, we deployed the LeNet-5 network with a 256×64 crossbar and evaluated the accuracy under different resistance conditions. While the line resistance is 5 Ω, the accuracy is restored to over 86% after CAFM optimization.

Compared to other optimization methods for IR-drop, CAFM demonstrates several advantages, as shown in Table 4. Refs. [12,38] both employ iterative computation of the weight matrix before mapping,

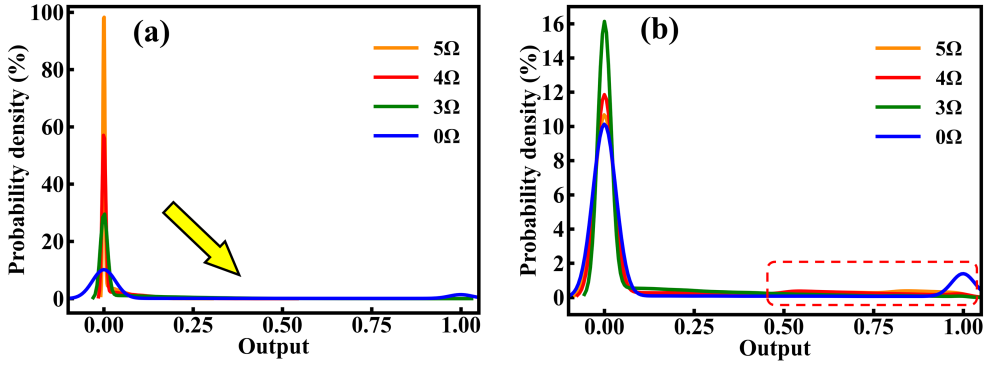

Figure 10 (Color online) Kernel density estimation (KDE) for the output of ReLU activation function. (a) w/o, (b) w/ CAFM optimization.

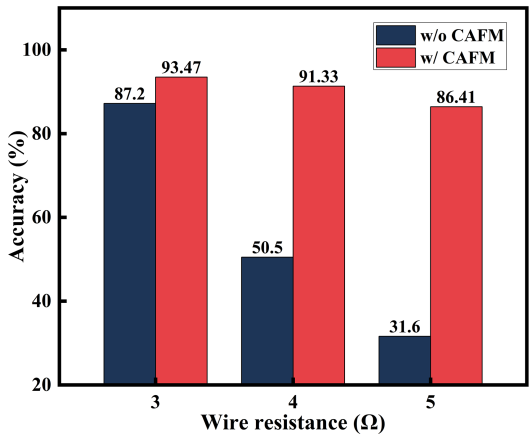

Figure 11 (Color online) Accuracy versus line resistance, w/o or w/ CAFM.

Table 4 Comparison between CAFM and other methods.

	[12]	[38]	[39]	CAFM
Network	Lenet-5	5-layer CNN	Lenet-5	Lenet-5
Dataset	MNIST	Fashion-MNIST	MNIST	MNIST
Iterative computation	✓	✓	–	–
Iteration time	Several hours	At least 4×10^2 s	–	–
Specific hardware design	–	–	✓	–
Ideal accuracy (%)	99.1	92.03	98.78	98.92
w/ R_{line} accuracy (%)	32	38.45	12	31.3
Recovered accuracy (%)	98.3	86.37	90	86.41

consuming additional computational resources and requiring long iteration times, and Ref. [12] even needs several hours to retrain the network. Ref. [39], although not requiring iterative computation, necessitates the design of special trans-impedance amplifiers (TIA) to restore accuracy, with each TIA parameter needing to design carefully, incurring additional hardware modules and lacking flexibility. Although the compensatory of CAFM is relatively limited compared to other methods, it does not require retraining the network or iterative computation, thus demanding fewer computational resources. Since the activation function module is essential in a memristor-based in-memory computing system, this work does not require changes to the crossbar configuration or special hardware overhead; it only needs little modifications based on the basic activation function module, providing flexibility. Additionally, if CAFM is fully implemented in the analog domain, the entire system will achieve better energy efficiency, which will be analyzed in Section 6.

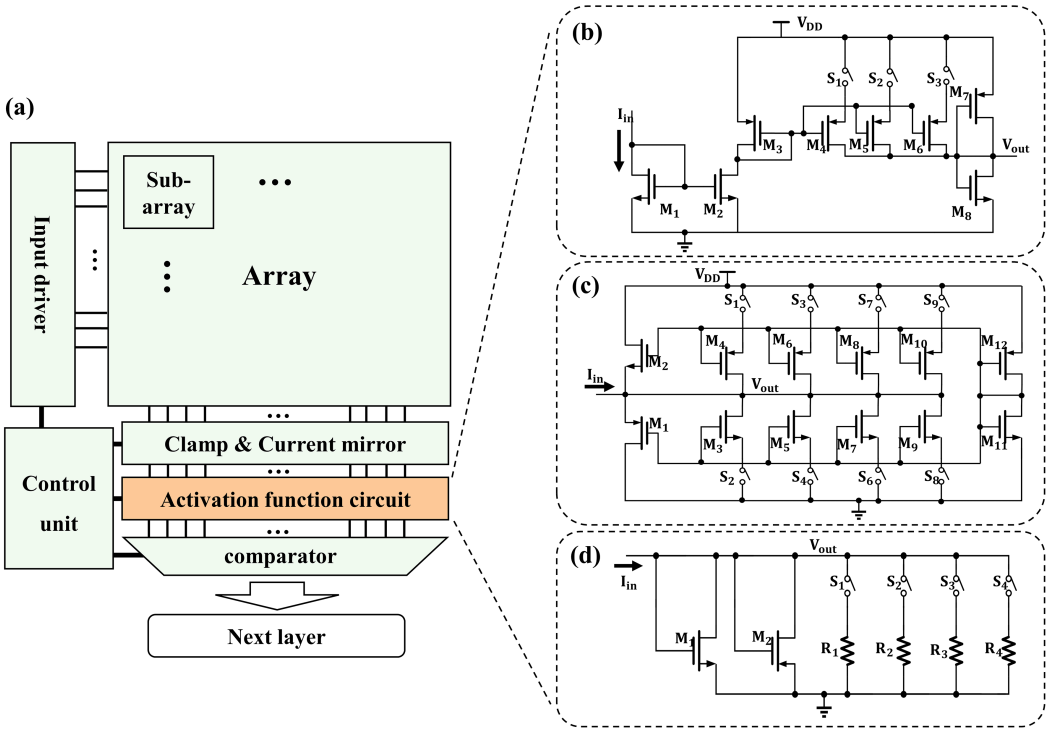

Figure 12 (Color online) Programmable activation function circuit. (a) Analog CIM system; (b) ReLU circuit; (c) Sigmoid circuit; (d) Tanh circuit.

6 Programmable activation function circuit

Based on the aforementioned AFM method, we designed the corresponding hardware architecture and activation function circuits. As shown in Figure 12(a), the entire hardware architecture consists of a large array containing multiple sub-arrays, a clamp and current mirror circuit, activation function circuits, comparators, input drivers, and a digital control unit.

This architecture adopts an ADC-free method, enabling full analog CIM and further saving energy. Firstly, the feature map is input by the driver, and the control unit distributes the input to corresponding sub-arrays. After the sub-array MAC operation is completed, the output is generated along the column direction. The outputs of each column's sub-arrays are accumulated through current mirrors. The accumulated result is then processed by a programmable activation function circuit, where the modulation of circuits is achieved by the control unit. If the circuit is for ReLU, it needs to be output to the next layer after passing through a comparator; otherwise, it is directly output.

Inspired by [40–42], we designed programmable CMOS activation function circuits, as shown in Figure 12(b), which implement the functionalities of ReLU, Sigmoid, and Tanh activation functions. The ReLU circuit primarily relies on the unidirectional conduction characteristics of NMOS and a current mirror structure to achieve activation functions. The transistors of Sigmoid circuit operate in different regions depending on the input current, and the Tanh circuit utilizes two transistors working in different states in combination with input currents and parallel resistors. These three circuits can control factor B by parallel-connecting transistors with varying width-to-length ratios or resistors of different sizes using switches. Additionally, these circuits replace the need for ADC/DAC, eliminating the process of analog-to-digital and digital-to-analog conversion, which is the most energy-consuming part in CIM systems. This enables the implementation of a full analog CIM system, further reducing energy consumption while minimizing precision loss.

The circuits were designed and simulated using Cadence Virtuoso, employing the CSMC 180-nm technology, with a frequency of 500 MHz. The output results of these three circuits are presented in Figure 13. The output of ReLU closely matches the ideal activation function output curve. The outputs of Sigmoid and Tanh are almost identical to the ideal values within the linear region, but slight differences are observed on the curves' extremities. We performed joint simulations with these three circuits and FAST while employing the AFM scheme.

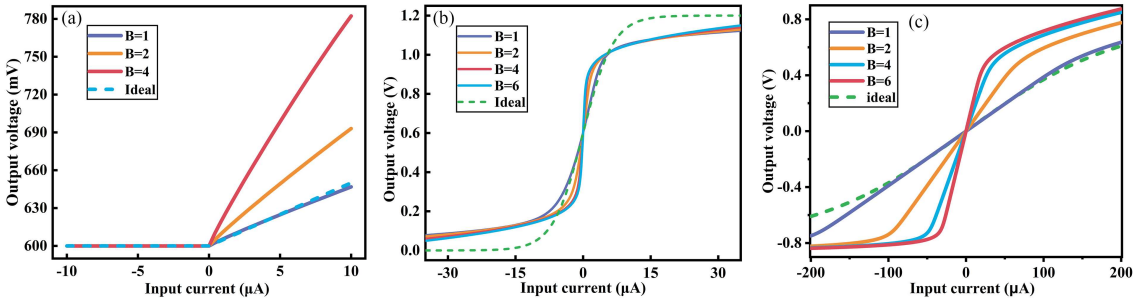

Figure 13 (Color online) Input-output curves of the programmable activation function circuits. (a) ReLU; (b) Sigmoid; (c) Tanh.

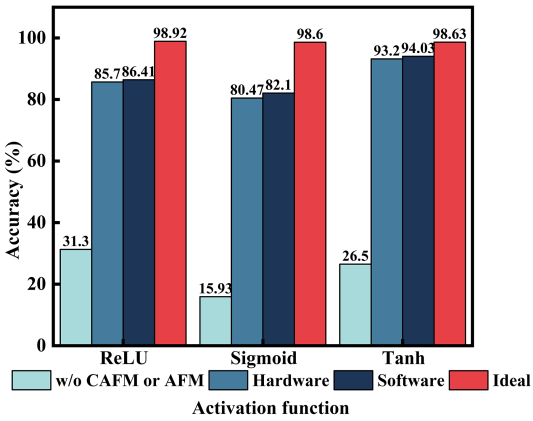

Figure 14 (Color online) Accuracy after optimization with three types of activation functions.

Under the conditions of line resistance of 5 Ω and a crossbar size of 256 × 64, we compared the output accuracy of the Lenet5 network under three scenarios: without CAFM/AFM, with CAFM/AFM implemented by circuit-based hardware and software. The results are shown in Figure 14. It can be seen that after CAFM and AFM optimization, the accuracy of circuit-based optimization scheme is restored to above 80%. In particular, the optimization scheme for the tanh activation function restores the accuracy to 93.2%. The results of the circuit-based optimization scheme are consistent with those of software AFM, demonstrating that the error in the output of the activation function circuit can be ignored. These circuits successfully implement programmable activation functions, demonstrating the collaborative optimization of hardware and software design.

The power and layout area of these circuits were analyzed and compared with prior activation function circuits, as shown in Table 5. Ref. [43] proposed a Sigmoid activation function circuit based on the coordinate rotation digital computer (CORDIC), which employs complex digital circuits, resulting in significant energy consumption. Ref. [44] presented an approximation method for Tanh activation functions using stochastic computing logic based on piecewise-linear approximation (PWL), where extensive use of ROM leads to large area overhead. Ref. [45] also used the PWL method to implement the Tanh activation function but employed a modified approximate compressor to save energy and area overhead. Leveraging the characteristics of memristor-based in-memory computing systems, three activation functions were implemented using analog circuits, requiring only a few transistors, which present significant advantages in terms of energy and area. Additionally, this design supports programmability and can be seamlessly integrated into analog CIM systems, features not present in the other studies.

7 Conclusion

In this work, we propose FAST, an end-to-end FAST, which achieves fast simulation at the array level with less storage and computational resources, and supports simulation of ultra-large-scale memristor crossbar. FAST uses the LIL format for sparse matrices to generate and store the conductance coefficient matrix, reducing the storage file size by more than 1300× compared to the dense matrix format, significantly saving storage resources. FAST dynamically selects matrix solution methods when simulating

Table 5 Comparison with prior activation function circuit.

	[43]	[44]	[45]	This work		
Technology (nm)	180	90	90	180	180	180
Frequency (MHz)	125	500	710	500	500	500
Activation function	Sigmoid	Tanh	Tanh	ReLU	Sigmoid	Tanh
Area (μm^2)	2825	3650	582	299	760	357
Power (μW)	1015.2	81.6	70.7	32.4	127.2	68.4
Programmable function	–	–	–	✓	✓	✓
Analog in-memory computing system	–	–	–	✓	✓	✓

crossbars of different sizes, using LU decomposition for small-scale crossbars and MINRES for large-scale crossbars. This approach greatly reduces computation time compared to matrix inversion, achieving optimal solution speed, especially for large-scale arrays. FAST can also achieve various functional array-level simulations, including implementing NN quantization training, completing weight mapping and inference processes on memristor crossbar, and accurately simulating the effects of various non-ideal effects on the accuracy of NNs. Using FAST, we evaluated the impact of non-ideal effects such as IR-drop, multi-level conductance, D2D variation, SAF, and the accuracy of AD/DA conversion on memristor NN accelerators. This evaluation led to the determination of optimal crossbar configurations to achieve maximum throughput and energy efficiency under non-ideal effects.

We have also introduced CAFM, a scheme to recover the main features of activation outputs, which enabled the accuracy of ReLU-based NNs affected by IR-drop to close to 90%. Finally, we designed the corresponding computation architecture and activation function circuits. The architecture design adopts an ADC-free method, enabling low-power full analog domain information transmission. We also designed three programmable activation function circuits, whose output curves are almost identical to the ideal values. The accuracy achieved through circuit implementation is restored to above 80%, with the tanh activation function achieving an accuracy of 93.2%. In conclusion, our work encompasses the development of an array-level simulator, analysis of non-ideal effects in memristor crossbars, and the corresponding circuit designs, resulting in a hardware-software optimization co-design.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant Nos. U2341221, 62274070), Hubei Province Science and Technology Major Project (Grant No. 2022AEA001), Interdisciplinary Research Program of Huazhong University of Science and Technology (Grant No. 2023JCYJ042), and Hubei Key Laboratory of Advanced Memories.

References

1 Yan B N, Yang Y C, Huang R. Memristive dynamics enabled neuromorphic computing systems. Sci China Inf Sci, 2023, 66: 200401
2 Rao M, Tang H, Wu J, et al. Thousands of conductance levels in memristors integrated on CMOS. Nature, 2023, 615: 823–829
3 Chang L, Li C L, Zhang Z M, et al. Energy-efficient computing-in-memory architecture for AI processor: device, circuit, architecture perspective. Sci China Inf Sci, 2021, 64: 160403
4 Yao P, Wu H, Gao B, et al. Fully hardware-implemented memristor convolutional neural network. Nature, 2020, 577: 641–646
5 Fouda M E, Lee J, Eltawil A M, et al. Overcoming crossbar nonidealities in binary neural networks through learning. In: Proceedings of the 14th IEEE/ACM International Symposium on Nanoscale Architectures, 2018. 31–33
6 An J J, Wang L F, Ye W, et al. Design memristor-based computing-in-memory for AI accelerators considering the interplay between devices, circuits, and system. Sci China Inf Sci, 2023, 66: 182404
7 Rashed M R H, Awad A, Jha S K, et al. Towards resilient analog in-memory deep learning via data layout re-organization. In: Proceedings of the 59th ACM/IEEE Design Automation Conference, 2022. 859–864
8 Yang X, Belakaria S, Joardar B K, et al. Multi-objective optimization of ReRAM crossbars for robust DNN inferencing under stochastic noise. In: Proceedings of IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2021. 1–9
9 Cheng C D, Tiw P J, Cai Y M, et al. In-memory computing with emerging nonvolatile memory devices. Sci China Inf Sci, 2021, 64: 221402
10 Zhu Y, Zhu Z, Dai G, et al. PIM-HLS: an automatic hardware generation tool for heterogeneous processing-in-memory-based neural network accelerators. In: Proceedings of 60th ACM/IEEE Design Automation Conference (DAC), 2023. 1–6
11 Peng X, Huang S, Luo Y, et al. DNN+NeuroSim: an end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies. In: Proceedings of IEEE International Electron Devices Meeting (IEDM), 2019
12 He Z, Lin J, Ewetz R, et al. Noise injection adaption: end-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping. In: Proceedings of the 56th Annual Design Automation Conference, 2019
13 Shadmehri S H H, BanaGozar A, Kamal M, et al. SySCIM: SystemC-AMS simulation of memristive computation in-memory. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), 2022. 1467–1472
14 Han R Z, Huang P, Zhao Y D, et al. Efficient evaluation model including interconnect resistance effect for large scale RRAM crossbar array matrix computing. Sci China Inf Sci, 2019, 62: 22401
15 Zhang W Q, Gao B, Yao P, et al. Array-level boosting method with spatial extended allocation to improve the accuracy of memristor based computing-in-memory chips. Sci China Inf Sci, 2021, 64: 160406
16 Li C, Belkin D, Li Y, et al. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. Nat Commun, 2018, 9: 2385
17 Han L X, Xiang Y C, Huang P, et al. Novel weight mapping method for reliable NVM based neural network. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS), 2021. 1–6
18 Shin H, Park R, Lee S Y, et al. Effective zero compression on ReRAM-based sparse DNN accelerators. In: Proceedings of the 59th ACM/IEEE Design Automation Conference, 2022. 949–954

19 Amin M H, Elbtity M E, Zand R. Xbar-partitioning: a practical way for parasitics and noise tolerance in analog IMC circuits. IEEE J Emerg Sel Top Circ Syst, 2022, 12: 867–877
20 Sun Z, Kvatinsky S, Si X, et al. A full spectrum of computing-in-memory technologies. Nat Electron, 2023, 6: 823–835
21 Liu Y Y, Gao B, Tang J S, et al. Architecture-circuit-technology co-optimization for resistive random access memory-based computation-in-memory chips. Sci China Inf Sci, 2023, 66: 200408
22 Cao T, Liu C, Gao Y, et al. Parasitic-aware modeling and neural network training scheme for energy-efficient processing-in-memory with resistive crossbar array. IEEE J Emerg Sel Top Circ Syst, 2022, 12: 436–444
23 Kim K, Song M S, Hwang H, et al. A comprehensive review of advanced trends: from artificial synapses to neuromorphic systems with consideration of non-ideal effects. Front Neurosci, 2024, 18: 1279708
24 Wang J J, Zhang T, Liu S, et al. Design and implementation of a hybrid, ADC/DAC-free, input-sparsity-aware, precision reconfigurable RRAM processing-in-memory chip. IEEE J Solid-State Circ, 2024, 59: 595–604
25 Zhang W, Gao B, Tang J, et al. Neuro-inspired computing chips. Nat Electron, 2020, 3: 371–382
26 Wan W, Kubendran R, Schaefer C, et al. A compute-in-memory chip based on resistive random-access memory. Nature, 2022, 608: 504–512
27 Chen P Y, Peng X, Yu S. NeuroSim+: an integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures. In: Proceedings of IEEE International Electron Devices Meeting (IEDM), 2017
28 Wang C, Mao G, Huang M, et al. HfO_x/AlO_y superlattice-like memristive synapse. Adv Sci, 2022, 9: 2201446
29 Pillage L. Electronic Circuit and System Simulation Methods. New York: McGraw-Hill, 1995
30 Xia L, Li B, Tang T, et al. MNSIM: simulation platform for memristor-based neuromorphic computing system. IEEE Trans Comput-Aided Des Integr Circ Syst, 2018, 37: 1009–1022
31 Duff I S, Erisman A M, Reid J K. Direct Methods for Space Matrices. Berlin: Springer, 1983
32 Schwarzenberg-Czerny A. On matrix factorization and efficient least squares solution. Astronom Astrophys Suppl, 1995, 110: 405
33 Hestenes M R, Stiefel E. Methods of Conjugate Gradients for Solving Linear Systems. Washington: NBS, 1952
34 Paige C C, Saunders M A. Solution of sparse indefinite systems of linear equations. SIAM J Numer Anal, 1975, 12: 617–629
35 Su R, Xiao R, Shen C, et al. Oxygen ion migration induced polarity switchable $SrFeO_x$ memristor for high-precision handwriting recognition. Appl Surf Sci, 2023, 617: 156620
36 Xia L, Huangfu W, Tang T, et al. Stuck-at fault tolerance in RRAM computing systems. IEEE J Emerg Sel Top Circ Syst, 2017, 8: 102–115
37 Song D, Yang F, Wang C, et al. Mitigate IR-drop effect by modulating neuron activation functions for implementing neural networks on memristor crossbar arrays. IEEE Electron Dev Lett, 2023, 44: 1280–1283
38 Liao Y, Gao B, Yao P, et al. Diagonal matrix regression layer: training neural networks on resistive crossbars with interconnect resistance effect. IEEE Trans Comput-Aided Des Integr Circ Syst, 2020, 40: 1662–1671
39 Huang C, Xu N, Qiu K, et al. Efficient and optimized methods for alleviating the impacts of IR-drop and fault in RRAM based neural computing systems. IEEE J Electron Dev Soc, 2021, 9: 645–652
40 Zhu J, Huang Y, Yang Z, et al. Analog implementation of reconfigurable convolutional neural network kernels. In: Proceedings of IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2019. 265–268
41 Khodabandehloo G, Mirhassani M, Ahmadi M. Analog implementation of a novel resistive-type sigmoidal neuron. IEEE Trans VLSI Syst, 2011, 20: 750–754
42 Shamsi J, Amirsoleimani A, Mirzakuchaki S, et al. Hyperbolic tangent passive resistive-type neuron. In: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), 2015. 581–584
43 Raut G, Rai S, Vishvakarma S K, et al. A CORDIC based configurable activation function for ANN applications. In: Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2020. 78–83
44 Nguyen V T, Luong T K, Le Duc H, et al. An efficient hardware implementation of activation functions using stochastic computing for deep neural networks. In: Proceedings of the 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2018. 233–236
45 Liu K, Shi W, Huang C, et al. Cost effective Tanh activation function circuits based on fast piecewise linear logic. MicroElectron J, 2023, 138: 105821