

An IR drop-robust Mapping Method for Reliable Memristive Accelerators

Jinpeng Liu, Shiyi Song, Bing Wu*, Huan Cheng, Heng Zhou, Xueliang Wei, Wei Tong, Dan Feng
 Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System,
 Engineering Research Center of data storage systems and Technology, Ministry of Education of China,
 School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
 {jinpengliu98, songshiyi, wubin200, chenghuan, heng_zhou, xueliang_wei, tongwei, dfeng}@hust.edu.cn

*Corresponding author

Abstract—Memristive accelerators (MAs) facilitate efficient matrix-vector multiplication (MVM) by performing in situ computation within memory crossbar arrays, thereby ensuring a fast and energy-efficient application acceleration. A significant challenge associated with the MA lies in the limited computing accuracy caused by the IR drop effect. However, existing IR drop mitigation works provide an approximate compensation, resulting in less accurate results. In this paper, we propose an IR drop-robust mapping method for reliable memristive accelerators. Firstly, the IR drop-robust mapping (IRM) method exploits the residuals between the equivalent matrix after the IR drop effect and the original matrix, and iteratively maps them to the crossbars for IR drop compensation. Based on the IRM method, a novel mechanism of the matrix-vector multiplication (MVM) operation is derived, ensuring that MVM is computed correctly. Secondly, the Calibrate-Shift-Reflect (CSR) strategy is developed to significantly reduce the number of arrays required by the IRM method to map the residuals. Thirdly, the hardware support for the IRM method is designed, and the overhead is reduced by sharing drivers/selectors between neighboring arrays. The experimental results indicate that the IRM-CSR method can effectively mitigate the IR drop effect, restoring inference accuracy by at most 80% (for neural network applications), and achieving a reduction in the relative root-mean-squared error by $10^3 \times \sim 10^{10} \times$ (for scientific computing), compared with the state-of-the-art methods.

Index Terms—In-memory computing, Memristive accelerator, IR drop compensation

I. INTRODUCTION

The recently emerged memristive accelerator (MA) offers a promising solution to the “memory wall” problem inherent in conventional von-Neumann architectures by computing in memory (CIM). Its ability to perform highly paralleled analog in-situ matrix-vector multiplication (MVM) within memory crossbar arrays (CIM arrays) minimizes data movement. This benefit is key to achieving application acceleration with superior energy efficiency, particularly for applications that utilize MVMs as the primary computing operations.

Unfortunately, a key challenge for CIM arrays is that the analog in-situ MVM, while offering performance benefits,

suffers a significant degradation in computational accuracy from the parasitic wire resistance, which is also known as the IR drop effect. The impact of the IR drop effect has been observed and analyzed in many simulations and experiments [1]. However, existing works for IR drop mitigation are not sufficiently precise, resulting in less accurate compensation results. Lepri et al [2] observe the correlation between the IR drop effect and array cell position. And they aim to mitigate the IR drop effect by averaging the output of the original array and the mirrored arrays. However, the IR drop effect is also related to the conductance pattern of the array, which is not taken into account. [3]–[6] attempt to calibrate the cell conductance using the heuristic algorithm before computation, so that the equivalent array conductance after IR drop approximates the ideal array conductance. However, these methods cannot always guarantee finding the correct target cell conductance, and can thus only return approximate values. [7], [8] try to retrain the neural network (NN) to be IR drop-robust. Nevertheless, the complex correlation between the IR drop and array conductance pattern often hinders retraining effectiveness.

In this paper, we propose an IR drop-robust mapping method for reliable memristive accelerators. We first recognize that the residual between the equivalent matrix after the IR drop effect and the original matrix is the cause of the inaccurate results. As these residuals are not involved in the MVM, the output inevitably differs from the ideal result. Instead of compensating for the output, our method iteratively maps the residuals to arrays, thereby incorporating them into the computation to achieve the correct result. Furthermore, the residual intrinsically encodes the influences of the IR drop effect, cell positions, and array conductance patterns. Consequently, our proposed IR drop-robust mapping method achieves application-irrelevant, high-accuracy computation. We make the following contributions to mitigate the IR drop effect.

- We propose the IR drop-robust mapping (IRM) method, which iteratively maps the residual matrices to arrays. Furthermore, we derive a novel computational mechanism for MVM with the proposed IRM method.
- We develop the Calibrate-Shift-Reflect (CSR) strategy to

This research was supported in part by the Young Scientists Fund of the National Natural Science Foundation of China under Grant 62302179, the National Natural Science Foundation of China under Grant 62172178, the Young Scientists Fund of the National Natural Science Foundation of China under Grant 62302182, and Cross-Research Support Program of Huazhong University of Science and Technology under Grant 2023CJYJ042.

enhance our IRM method with a decreased number of arrays needed. And we design the hardware support for our mapping method.

- We demonstrate the effectiveness of our method using various applications, including neural network inference and scientific computing. Experiments show that our method has the highest computing accuracy compared with the state-of-the-art (SOTA) compensation methods, with acceptable power/area overhead.

II. BACKGROUND

A. Memristive accelerator & Analog in-memory computing

Memristive accelerators offer a promising solution to mitigate the memory bottleneck of conventional von Neumann architectures by analog in-memory computing. Fig. 1a presents a typical MA architecture, consisting of a hierarchical structure with multiple levels: CIM array, processing element (PE), tile, bank, and chip. As shown in Fig. 1b, CIM arrays can be built in a crossbar structure, which achieves both storage and analog computing of MVM ($G^T \mathbf{v}$) between the input voltage vector \mathbf{v} and the array conductance \mathbf{G} within $O(1)$ time complexity. Thus, efficient MVM between the weight matrix \mathbf{W} and the input vector \mathbf{x} can be achieved by mapping \mathbf{W} to the conductance matrix and \mathbf{x} to the input voltage vector.

Specifically, \mathbf{W} is linearly mapped to the conductance range of the array cells $[G_{\min}, G_{\max}]$ by $\mathbf{G} = \mathbf{K} \cdot \mathbf{W} + \mathbf{B}^1$, an approach commonly referred to as true analog computing [9]. Another approach is to map W using a limited set of predetermined states (typically binary) and perform the computation through a bit-slicing scheme across multiple arrays [10]. The true analog approach has a higher computational efficiency. In this paper, we adopt the true analog computing, but our solution is also applicable to the bit-slicing approach.

B. IR drop effect

Ideally, for MVM in CIM arrays, the input voltage applied to the rows (word lines) is fully transferred to the cells within the array without attenuation, and the aggregated cell currents within each column (bit line) produce the correct output current. However, the parasitic wire resistances inevitably cause a voltage drop (IR drop), which distorts cell currents as well as the final output current in a highly complex manner [1], [6], leading to a degradation in computing accuracy.

¹The scaling factors \mathbf{K} and \mathbf{B} can be calculated by $\mathbf{K}_{ij} = \frac{G_{\max} - G_{\min}}{\mathbf{W}_{j_{\max}} - \mathbf{W}_{j_{\min}}}$ and $\mathbf{B}_{ij} = -\mathbf{K}_{ij} \cdot \mathbf{W}_{j_{\min}} + G_{\min}$ for each matrix element. Unless explicitly specified, the “ \cdot ” and “ $/$ ” operations in this paper are element-wise operations between matrices or vectors with the same shape, while $\mathbf{W}^T \mathbf{x}$ is the matrix-vector multiplication operation.

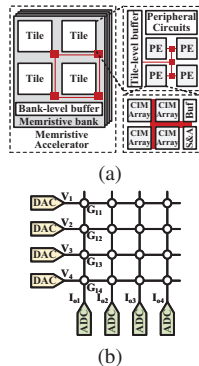


Fig. 1. Analog in-memory computing. (a) Memristive accelerator architecture. (b) CIM array structure.

Fig. 2 illustrates the effective voltage of the cells under the influence of IR drop during MVM operations. The test parameters are provided in TABLE I, and we set all cells to G_{\max} (the worst IR drop case). The experimental results reveal a significant voltage drop in both the row and column directions, with the worst attenuation reaching 70% of the input voltage. This substantial voltage drop can lead to significant computational errors, underscoring the importance of addressing the IR drop issue.

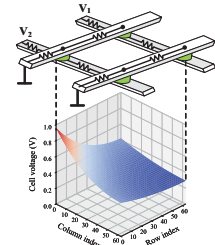


Fig. 2. Cell voltage distribution under IR drop effect.

C. Related Work

Recently, many works have been proposed to handle the IR drop problem. [2] conducts experiments on arrays with uniform conductance patterns and observes that the correlation between IR drop and array cell position can be eliminated by averaging the output of the original array and the mirrored arrays. And the IR drop problem can be mitigated by multiplying the same scaling factor to the output result of each column because cells on different columns suffer the same amount of voltage distortion. However, the IR drop effect also has a relationship with the array conductance pattern, which is not considered in Mirrored. [3], [5], [6] try to calibrate the cell conductance using the gradient descent algorithms prior to computing, so that the equivalent conductance after the IR drop effect is close to the ideal cell conductance. However, the gradient descent approach adopted by these methods is sometimes not able to find the proper cell conductance for larger arrays or cases where IR drop has a significant impact. [7], [8] try to retrain the neural network to be IR drop-robust. However, it is often impossible to compute the gradient of the accurate IR drop simulation algorithms. The approximate simulation methods used in their works, and the difficulty of predicting the complex IR drop effect, result in poor effectiveness even after retraining the neural network. What is more, the applicable cases of these methods are limited to neural network applications. [4] examines the role of the batchnorm (BN) layers of the NN in IR drop mitigation, and finds out that calibrating parameters of the BN layers without retraining the entire NN can resolve the IR drop effect. However, the BN layers are often integrated into the convolution layers in the NN inference process, and calibrating the parameters of the BN is the same as calibrating the cell conductance.

In summary, existing works have limitations in accurately compensating for the IR drop effect, due to ignoring the data pattern, the uncertainty of gradient descent, or ineffective learning algorithms. Therefore, there is an urgent need for accurate methods to compensate for the IR drop effect.

III. MAPPING METHOD DESIGN

The key insight of the proposed mapping method is as follows. After the IR drop effect, according to [11], the

equivalent matrix (\mathbf{W}_e) of the original matrix (\mathbf{W}) can be easily acquired, and the residual matrix (\mathbf{R}) between \mathbf{W}_e and \mathbf{W} can be computed ($\mathbf{R} = \mathbf{W} - \mathbf{W}_e$). To compensate for the IR drop effect, the residual matrix must be incorporated into the computation, apart from the original matrix. Unfortunately, mapping the residual matrix to the crossbar array also suffers from the IR drop effect, and its residual should be taken into consideration. The same applies to the subsequent residuals. Therefore, an iterative mapping of the residual matrices is required. After the iterative mapping and computation, the IR drop effect is compensated by accumulating the output results of the original matrix and the residual matrices.

In this section, the IR drop-robust mapping method (section III-A) following the key insight is proposed. And the new MVM mechanism for the proposed mapping method is derived in section III-B. Since each iteration requires mapping a residual matrix to a new crossbar array, section III-C provides a strategy to reduce the number of iterations and minimize the number of crossbars required.

A. IR drop-robust mapping (IRM) method

Algorithm 1: IR drop-robust mapping method.

Input : Weight matrix (\mathbf{W}), maximum array number (N), user defined tolerance (tol).
Output: List of conductance arrays to be programmed to the crossbars (\mathbf{G}_s), list of scaling factors ($\mathbf{K}_s, \mathbf{B}_s$).

```

1  $\mathbf{R}^{(0)} = \mathbf{W}$ 
2 for  $s \leftarrow 0$  to  $N - 1$  do
3   if  $\|\mathbf{R}^{(s)}\|_F < tol$  then
4     Break
5   end
6    $\mathbf{G}^{(s)}, \mathbf{K}^{(s)}, \mathbf{B}^{(s)} = \text{toCondArray}(\mathbf{R}^{(s)})$ 
7    $\mathbf{G}_s.append(\mathbf{G}^{(s)}); \mathbf{K}_s.append(\mathbf{K}^{(s)}); \mathbf{B}_s.append(\mathbf{B}^{(s)})$ 
8    $\mathbf{G}_e^{(s)} = \text{equiArray}(\mathbf{G}^{(s)})$ 
9    $\mathbf{R}_e^{(s)} = \text{toMatrix}(\mathbf{G}_e^{(s)}, \mathbf{K}^{(s)}, \mathbf{B}^{(s)})$ 
10   $\mathbf{R}^{(s+1)} = \mathbf{R}^{(s)} - \mathbf{R}_e^{(s)}$ 
11 end
12 return  $\mathbf{G}_s, \mathbf{K}_s, \mathbf{B}_s$ 
```

The IR drop-robust mapping (IRM) method is demonstrated in Alg. 1. First in Line #1, the original matrix (\mathbf{W}) is regarded as the initial residual matrix ($\mathbf{R}^{(0)}$). Then, Line #2-Line #11 are the iterative mapping process of the residual matrices. The iteration terminates if the number of residual matrices exceeds the maximum, or the residual matrix is close to zero².

When mapping the weight matrix (\mathbf{W}) or the residual matrix (\mathbf{R}) to the crossbar array, it should first be converted into the conductance array as $\mathbf{G} = \mathbf{K} \cdot \mathbf{R} + \mathbf{B}$ (Line #6), where \mathbf{K} and \mathbf{B} are the scaling factors.

After the IR drop effect, according to [11], the equivalent conductance array is $\mathbf{G}_e = \text{equiArray}(\mathbf{G})$ (Line #8), and the real output current of the array is $\mathbf{i}_{\text{real}} = \mathbf{G}_e^T \mathbf{v}$. To compensate for the IR drop effect, it is obvious that a compensation current, which can be derived as $\Delta \mathbf{i} = \mathbf{i}_{\text{ideal}} - \mathbf{i}_{\text{real}} = (\mathbf{G}^T - \mathbf{G}_e^T) \mathbf{v}$, should be added to the real output current, and an array whose equivalent conductance is $\mathbf{G} - \mathbf{G}_e$ is needed. However, programming these residuals directly to the crossbar may not be possible. For example, if there exists

a $\mathbf{G}_{ij} = G_{\text{min}}$, whose equivalent conductance can become $0 < \mathbf{G}_{e_{ij}} < G_{\text{min}}$, then $\mathbf{G}_{ij} - \mathbf{G}_{e_{ij}} < G_{\text{min}}$, which is not a legal cell conductance.

Therefore, our method adopts another approach. We convert the equivalent conductance array back into the matrix with $\mathbf{R}_e = (\mathbf{G}_e - \mathbf{B})/\mathbf{K}$ (Line #9), and define \mathbf{R}_e as the equivalent matrix of \mathbf{R}^3 . We define $\mathbf{R}' = \mathbf{R} - \mathbf{R}_e$ as the residual of \mathbf{R} (Line #10). Thus, our method needs to map \mathbf{R}' to a new crossbar. Unfortunately, due to the IR drop effect again, mapping \mathbf{R} to the crossbar array directly is unable to accomplish this task; therefore, we iteratively map the following residual matrices to new crossbar arrays.

B. MVM with IRM

As mentioned in section III-A, the original matrix \mathbf{W} is mapped as \mathbf{G}_s (whose equivalent conductance arrays are $\mathbf{G}_{e,s}$), with scaling factors $\mathbf{K}_s, \mathbf{B}_s$, and the relationship between \mathbf{W} and $\mathbf{G}_{e,s}$ is derived as $\mathbf{W} = \mathbf{R}^{(0)} \approx \sum_{s=0}^{N-1} [\mathbf{R}^{(s)}] = \sum_{s=0}^{N-1} [\frac{\mathbf{G}_e^{(s)} - \mathbf{B}^{(s)}}{\mathbf{K}^{(s)}}]$. After matrix mapping with the IRM method, the MVM is processed equivalently as $\mathbf{W}^T \mathbf{x} \approx \sum_{s=0}^{N-1} [\frac{\mathbf{G}_e^{(s)} - \mathbf{B}^{(s)}}{\mathbf{K}^{(s)}}]^T \frac{\mathbf{v}}{c} = \frac{\sum_{s=0}^{N-1} \mathbf{y}^{(s)}}{c}$ ⁴. As mentioned in section II-A, elements on the j -th column of $\mathbf{B}^{(s)}$ are the same, and their values can be represented as $\mathbf{B}_{0j}^{(s)}$. Similarly, elements on the j -th column of $\mathbf{K}^{(s)}$ can be represented as $\mathbf{K}_{0j}^{(s)}$. Therefore, $\mathbf{y}_j^{(s)} = \{[\frac{\mathbf{G}_e^{(s)} - \mathbf{B}^{(s)}}{\mathbf{K}^{(s)}}]^T \mathbf{v}\}_j = \frac{\mathbf{i}_j^{(s)} - \mathbf{B}_{0j}^{(s)} \cdot \mathbf{v}_{\text{sum}}}{\mathbf{K}_{0j}^{(s)}}$

holds for the output of crossbar s , where $\mathbf{i}_j^{(s)}$ is the output current of the j -th column of the crossbar, and \mathbf{v}_{sum} is the sum of all input voltages. The computation of $\mathbf{y}^{(s)}$ is performed by our designed analog arithmetic unit in section ??.

C. Calibrate-Shift-Reflect (CSR) strategy

As demonstrated in Fig. 3a, when using the IRM method mentioned in section III-A, we observe that the Frobenius norm (FN) of the residual matrix ($\|\mathbf{R}^{(s)}\|_F$) decreases slowly, which inevitably increases the number of crossbar arrays used for mapping. The reasons for the slow decline are as follows. (1) The equivalent matrix infected by the IR drop effect is very different from the original matrix, leading to a large residual. (2) Elements with larger values in the residual matrix are mapped to the crossbar positions where the IR drop effect is severe, and these elements hardly ever become small. Therefore, we propose the Calibrate-Shift-Reflect (CSR) strategy as an enhancement for the IRM method to quickly converge the FN to the user-defined tolerance (e.g., 1e-4), using as few crossbar arrays as possible.

The ‘‘Calibrate’’ strategy is proposed to solve problem (1) mentioned above. To make the equivalent matrix (\mathbf{W}_e) approximately the same as the original matrix (\mathbf{W}), we choose to calibrate the value of the conductance array (\mathbf{G}) in the ‘‘toCondArray’’ process in Alg. 1, Line #6. The calibrated conductance array is denoted as \mathbf{G}_c . Intuitively, it is optimal

³ $\mathbf{G} - \mathbf{G}_e$ can be calculated by $\mathbf{G} - \mathbf{G}_e = \mathbf{K} \cdot (\frac{\mathbf{G} - \mathbf{B}}{\mathbf{K}} - \frac{\mathbf{G}_e - \mathbf{B}}{\mathbf{K}}) = \mathbf{K} \cdot (\mathbf{W} - \mathbf{W}_e)$

⁴ c is the DAC conversion constant.

²Its Frobenius norm (FN) is smaller than the user-defined tolerance. The Frobenius norm is defined as $\|\mathbf{W}\|_F = (\sum_i \sum_j \mathbf{W}_{ij}^2)^{\frac{1}{2}}$

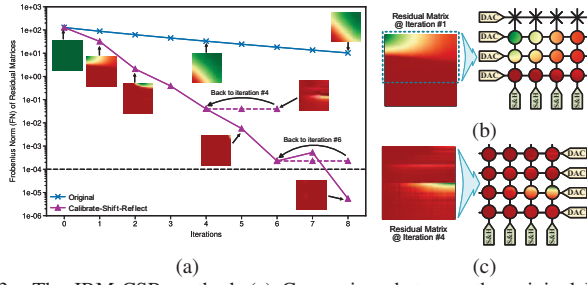


Fig. 3. The IRM-CSR method. (a) Comparison between the original IRM method and the enhanced IRM-CSR method. The matrix in the figure represents the residual matrix for each iteration, with colors ranging from red to green indicating values that increase from low to high. (b) Adopting the “Shift” strategy for quick residual reduction. (c) Adopting the “Reflect” strategy when the residual converges to a value larger than the given tolerance.

that $\mathbf{Gc}_e \approx \mathbf{G}$, so that the equivalent matrix ($\mathbf{Wc}_e = (\mathbf{Gc}_e - \mathbf{B})/\mathbf{K} \approx \mathbf{W}$), and the residual matrix will be close to zero. Unfortunately, this may not be possible for two reasons. First, if there exists a $\mathbf{G}_{ij} = G_{max}$, the $\mathbf{Gc}_{e_{ij}}$ needs to be scaled up to exceed G_{max} so that $\mathbf{Gc}_{e_{ij}} \approx G_{max}$ after the IR drop effect, which is not possible. Second, crossbar cells suffer from the IR drop effect differently; cells that are more severely affected by the IR drop can exhibit conductance that exceeds G_{max} after calibration, even when less affected cells do not. Therefore, our strategy adopts other approaches. First, regarding the “Scaling beyond boundary” issue, we attempt to narrow the range of conductance values mapped from the matrix. We set a scaling factor ($\mu < 1.0$) and $[G_{min} + \delta G, G_{max} - \delta G]^5$ as the new conductance range⁶. In that case, the calibrated cell conductance can be less than G_{max} even if the original conductance is $G_{max} - \delta G$. Should the calibrated cell conductance does exceed G_{max} , it will be set to G_{max} , and the residual will not be zero. That is when the IRM method takes place and further reduces the residuals. Second, taking into account the differences in the effect of IR drop between cells, we set an IR drop-aware constant matrix (\mathbf{C}) for the crossbar as follows. We compute the ratio of the equivalent conductance matrix to the original, and select the value in the middle of each column of the ratio as the constant for the corresponding crossbar column (e.g., $\mathbf{C}_{ij} = [\mathbf{G}_e/\mathbf{G}]_{\frac{m}{2}, j}$, m is the number of rows of the crossbar). And our goal becomes that for every cell $\mathbf{Gc}_{e_{ij}} \approx \mathbf{C}_{ij} \cdot \mathbf{G}_{ij}$ (or in another way, $\min(\text{FN}(\mathbf{Gc}_e - \mathbf{C} \cdot \mathbf{G}))$). In that case, cells that are more severely affected by the IR drop can exhibit conductance scaled to the same multiple after calibration as less affected cells. To achieve this goal and calibrate the conductance array, we adopt the Newton-Raphson method with $\mathbf{Gc}^{(t+1)} = \mathbf{Gc}^{(t)} - \frac{\mathbf{Gc}^{(t)}}{\mathbf{Gc}_e^{(t)}} \cdot [\mathbf{Gc}_e^{(t)} - \mathbf{C} \cdot \mathbf{G}]$ as the iterative formula. With these designs, the “toCondArray” process now becomes $\mathbf{G} = \mathbf{K}' \cdot \mathbf{R} + \mathbf{B}'$, $\mathbf{G}_{ij} \in [G_{min} + \delta G, G_{max} - \delta G]$, and calibrates the conductance array to \mathbf{Gc} , $\mathbf{Gc}_{ij} \in [G_{min}, G_{max}]$. Then the equivalent conductance array becomes \mathbf{Gc}_e after

⁵ $\delta G = \frac{1-\mu}{2}(G_{max} - G_{min})$

⁶Please note that we only change to map the original matrix elements to $[G_{min} + \delta G, G_{max} - \delta G]$, and the range of the memristor conductance is still $[G_{min}, G_{max}]$

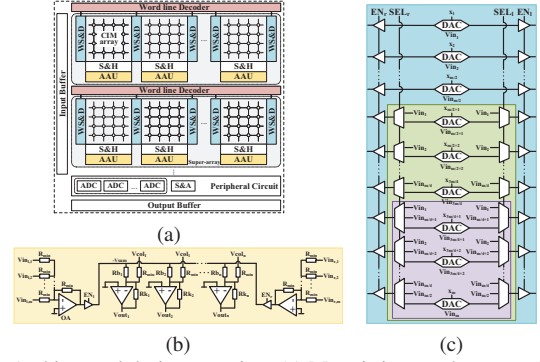


Fig. 4. Architectural design overview. (a) Memristive accelerator architecture at the PE level. (b) Analog arithmetic unit (AAU) circuit. (c) Word line selector and driver (WS&D) circuit.

the IR drop effect. And the “toMatrix” process now becomes $\mathbf{Rc} = (\mathbf{Gc}_e/\mathbf{C} - \mathbf{B}')/\mathbf{K}'$. The computation for $\mathbf{y}^{(s)}$ in section III-B becomes $\mathbf{y}_j^{(s)} = \frac{\mathbf{i}_j^{(s)} - \mathbf{B}'_{0j}^{(s)} \cdot \mathbf{C}_{0j}^{(s)} \cdot \mathbf{v}_{sum}}{\mathbf{K}'_{0j}^{(s)} \cdot \mathbf{C}_{0j}^{(s)}}$.

The “Shift” strategy tries to solve problem (2). As demonstrated in Fig. 3a, after the first iteration, the upper part of the residual matrix has larger values. It will be wise to map this part to the crossbar position where the IR drop effect is mild. Therefore, as demonstrated in Fig. 3b, in the second iteration, we shift the upper part of the residual matrix⁷ to the bottom part of the crossbar array, compute the new residual matrix, and concatenate the new residual with the lower part of the residual matrix from the previous iteration. And the concatenated residual matrix will be left for the next iteration.

As can be noticed in Fig. 3a, the residual converges to a value larger than the given tolerance from iteration #4 to iteration #6. This is because through continuous iteration and shifting, the portion of the residual matrix mapped to the crossbar becomes progressively smaller, and these mapped portions exhibit relatively small residuals. Conversely, the residuals in the portions not yet mapped to the array become relatively large. Therefore, we go back to the previous iteration and find out the right part of the residual matrix is larger. Then, we reflect the DAC from adding input voltage from the left to adding input voltage from the right, and map the entire residual matrix to the new crossbar array. We then repeat the Calibrate-Shift strategy until reaching the next convergence, and reflect the DAC back.

IV. HARDWARE ARCHITECTURE DESIGN

As illustrated in Fig. 4a, our architectural design focuses on the PE level. We group and define crossbars that share the same input as the super-array, and add analog arithmetic units (AAU), word line selectors and drivers (WS&D), word line decoders (WD), as the hardware support for our IRM-CSR method. The AAU is designed to support the new mechanism for matrix-vector multiplication (mentioned in section III-B). The WS&D is shared by adjacent crossbars to reduce hardware

⁷For simplicity, we set a fixed proportion (λ) as the size of the portion shifted based on the previous iteration. For example, the size of the shift in iteration #2 is $\lambda \cdot m$ rows, and it will be $\lambda^2 \cdot m$ in iteration #3, and the subsequent iterations follow the same pattern.

overhead and provides support for the proposed Shift-Reflect strategy.

As demonstrated in Fig. 4c, sharing by the neighboring crossbars, the WS&D reuses the DACs from the original PE to generate the same input voltages for both crossbars. Meanwhile, we add two sets of transmission gates and two signals generated from the word line decoder to enable input voltages added to the right side of the crossbar to the left (signal EN_r), and the left side of the crossbar to the right (signal EN_l) at the same time. Recalling that in the Shift strategy, the upper part of the residual matrix is shifted to the bottom of the crossbar array at a ratio λ , and the input voltages also need to be shifted to the bottom of the array. Therefore, we add two sets of multiplexers and two signals (SEL_r and SEL_l) to support input voltage shifting. For example, if half of the rows at the upper part of the residual matrix are shifted to the bottom part of the crossbar array, $V_{in_1}-V_{in_{\frac{m}{2}}}$ will be selected in the green box in Fig. 4c.

V. EVALUATION

In this section, we first demonstrate how we build up the evaluation, followed by the validation of the effectiveness of the IRM-CSR method through a comparison with the Baseline and the SOTA compensation methods. Then we conduct sensitivity studies with multiple crossbar/cell configurations. Subsequently, the overhead of the proposed hardware is presented.

A. Evaluation Setup

TABLE I
PARAMETERS USED IN THE EVALUATIONS.

Symbol	Description	Value
$G_{max}(S)$	Maximum conductance	1.8e-4 [12], 5.9e-5 [13], 2.9e-4 [14]
$G_{min}(S)$	Minimum conductance	2.5e-5, 1.3e-5, 1.2e-4
σ	Device variation	1.5% [15], 5% [13]
$R_w(\Omega)$	Wire resistance	1.0, 3.0, 5.0, 10.0
$V_{op}(V)$	Operation voltage	0.1
S_{cbr}	Crossbar array size	64, 128, 256

1) *Compensation methods*: A comparative analysis will be conducted with the following methods. (a) the Baseline, which lacks any compensation; (b) the Mirrored method, which adopts the ‘‘R8’’ strategy mentioned in [2], and the scaling factor multiplied to the output current is computed as $\text{avg}(\mathbf{G}_j/\mathbf{G}_{e_j})$ for each crossbar column; (c) the Deembedding method, which is an enhanced version of the original method in [6]; Since the gradient descent algorithm in the original method is hard to converge, the wire resistance is set to 0.1Ω at the beginning, and a 0.5Ω is added to the wire resistance each iteration, until the wire resistance is the same as R_w in TABLE I. Meanwhile, the calibration in the original method faces the ‘‘Scaling beyond boundary’’ issue. Therefore, we narrow the range of the conductance value to $[G_{min}, G_{min} + \mu \cdot \Delta G]^8$. (d) the Retraining method, which uses the SCN mentioned in [16] for approximate IR drop simulation to enable retraining; (e) our IRM-CSR method.

⁸ $\Delta G = G_{max} - G_{min}$.

TABLE II
COMPARISON OF DIFFERENT COMPENSATION METHODS.

Network & Dataset		LeNet-5 & MNIST		VGG-11 & Cifar-10		GoogleNet & Cifar-10	
S_{cbr}		64	128	64	128	64	128
Baseline	Accuracy(^①)	12.40	10.57	10.00	10.00	9.78	10.08
IRM-CSR (Ours)	Accuracy(%)	98.35	94.64	90.85	90.02	89.72	88.96
	Recovery(%)	+84.07	+86.58	+80.85	+80.02	+79.94	+78.88
	Array# ^②	1	2	1	3	2	3
Mirrored	Accuracy(%)	14.14	10.11	18.54	10.00	10.00	10.00
Deembedding	Accuracy(%)	98.58	10.24	91.00	10.04	89.79	10.02
Retraining	Accuracy(%)	98.67	93.98	- ^③	-	-	-
GPU	Accuracy(%)	99.02		91.33		89.98	

^① The evaluations are conducted with the following configuration. $G_{max}=1.8e-4S$, $G_{min}=2.5e-5S$, $R_w=3.0\Omega$, $\mu=0.2$, $\lambda=0.5$.

^② ‘‘Array#’’ is the number of arrays needed for mapping when using the IRM-CSR method.

^③ ‘‘-’’ means that we are not able to reproduce the result of the previous work.

2) *Benchmarks & evaluation criteria*: The evaluation of the methods is conducted using a variety of real-world benchmarks. For the NN benchmarks with MNIST as the dataset, we use LeNet-5. For the NN benchmarks with Cifar-10 as the dataset, we use VGG-11 and GoogleNet. We first train these NNs on high-performance GPUs with 16-bit floating point precision, and the inference accuracies are 99.02%, 91.33%, and 89.98%, respectively. Subsequent deployment of these models to the ideal memristive accelerator yields inference accuracies of 98.51%, 91.40%, and 89.95%, respectively. The input features are quantized to 8-bit, and the weight matrices are mapped to the crossbar array using the approach mentioned in section II-A. For the non-NN benchmarks, we use scientific computing. The scientific computing solves for $x = A^{-1}b$, and is implemented using the conjugate gradient (CG) approach as in [17] with bcstk22 (a 138×138 symmetric matrix with 696 nonzeros) and blkhole (a 2132×2132 symmetric matrix with 14872 nonzeros) from the SuiteSparse matrix collection [18] as the dataset. As for the data representation, we use the IEEE-754 double-precision floating-point number with the mantissa mapped to the array after alignment. We run the scientific computing applications on the ideal memristive accelerator, and the RRMSE⁹ of the result x , computed using the CG approach, from the theoretical result y is beneath 10^{-5} after 409 iterations (for bcstk22) and 6684 iterations (for blkhole). For the NN benchmarks, the compensation method is more effective with a higher inference accuracy after IR drop mitigation. In the context of scientific computing, a suitable compensation method can rapidly reduce the RRMSE to an acceptable level.

3) *Evaluation environment*: In this work, all of the above mentioned applications and compensation methods are implemented by extending Fadesim [1]. And the evaluations are running on a $4 \times$ Tesla V100 server. And the parameters, derived from real laboratory-fabricated memristor cells and crossbar arrays, are tabulated in TABLE I.

B. Effectiveness of the IRM-CSR method

As illustrated in TABLE II, for neural network applications, compared to Baseline, the IRM-CSR method restores the inference accuracy of LeNet-5, VGG-11, GoogleNet by 84.07%, 80.85%, 79.94% for 64×64 crossbar array, and 86.58%, 80.02%, 78.88% for 128×128 , respectively. For scientific computing applications, as shown in Fig. 5, experiment

⁹The RRMSE is defined as $[\sum_i (\mathbf{x}_i - \mathbf{y}_i)^2]^{\frac{1}{2}} / [\sum_i \mathbf{y}_i^2]^{\frac{1}{2}}$.

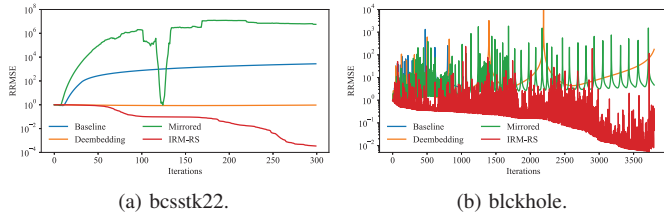


Fig. 5. Compensation results of the IRM-CSR method for scientific computing compared with SOTA compensation methods.

results demonstrate a consistent decline in RRMSE for the IRM-CSR method, ultimately converging to 1.4×10^{-4} and 7.3×10^{-3} for *bcstk22* and *blkhole* at 300 iterations and 3600 iterations, respectively. When compared with the SOTA methods, the IRM-CSR method restores inference accuracy by at most 80% for neural network applications. And the IRM-CSR method achieves a substantial reduction in mean-squared error by $10^3 \times \sim 10^{10} \times$ for scientific computing compared with the SOTA methods. These are because the IRM-CSR method brings accurate inference (of the NN) and convergence (of the scientific computing), while the less accurate SOTA methods do not. The IRM-CSR method triumphs over the SOTA methods for the following reasons. (1) The IRM-CSR method utilizes the residual matrices that imply differences in the IR drop among cell positions in the crossbar array and the matrix data pattern, which the Mirrored method failed to take into consideration. (2) Our mapping method is determined, with the Frobenius norm (FN) of the residuals constantly converging to zero, rather than the approximation approach adopted by the previous conductance calibration methods. (3) Our method relies only on the residuals of the matrix and is application-irrelevant, compared with the retraining methods. The reason for the low accuracy of the baseline and SOTA methods is that true analog computing is extremely sensitive to the IR drop effect. The output current deviation caused by the IR drop effect is amplified by the scaling factors \mathbf{K} and \mathbf{B} . Our experiment on a single crossbar array illustrates that the actual output results deviate from the theoretical results by 834% without compensation. And the SOTA methods are not suitable for true analog computing.

C. Sensitivity study

In this section, an investigation is conducted into the functionality of the IRM-CSR method in various hardware

TABLE III
COMPENSATION RESULTS OF THE IRM-CSR METHOD WITH DIFFERENT DEVICE CONFIGURATIONS.

S_{xbr}	$G_{max}/G_{min}(S)$	$R_w(\Omega)$	σ	Cellbit	S_{OU}	IRM-CSR	
						Baseline GoogleNet	Accuracy(%) Array#
64	1.8e-4/2.5e-5	3.0	-	-	-	9.78	89.72 2
128	1.8e-4/2.5e-5	3.0	-	-	-	10.08	88.96 3
256	1.8e-4/2.5e-5	3.0	-	-	32	10.00	90.05 1
128	1.8e-4/2.5e-5	1.0	-	-	-	10.01	89.66 1
128	1.8e-4/2.5e-5	5.0	-	-	64	10.00	89.92 2
128	1.8e-4/2.5e-5	10.0	-	-	32	10.02	89.87 2
128	5.9e-5/1.3e-5	3.0	-	-	-	10.01	88.39 2
128	2.9e-4/1.2e-4	3.0	-	-	32	10.00	89.85 1
128	1.8e-4/2.5e-5	3.0	1.5%	-	64	10.01	89.82 4
128	1.8e-4/2.5e-5	3.0	5%	-	32	10.00	89.95 6
128	1.8e-4/2.5e-5	3.0	-	2	64	32.47	87.83 7
128	1.8e-4/2.5e-5	3.0	-	4	64	34.86	88.69 5

^① "-" means that the device variation is not taken into consideration, or the true analog computing is adopted, or the computing granularity is the entire crossbar array instead of the operation unit.

configurations. To this end, laboratory-fabricated devices with distinct cell conductance, wire resistance, and other characteristics are utilized. Additionally, the applicability of the IRM-CSR method with device variation and bit-slicing is explored. The results in TABLE III demonstrate that our approach can effectively mitigate the impact of the IR drop at varying severity levels on the computational results. Furthermore, our method can be applied to configurations where the device itself exhibits variability, as well as to bit-slicing configurations. To minimize the number of crossbar arrays used as much as possible, we performed calculations at the operation unit (OU) granularity in some evaluations (e.g., $S_{OU} = 32$). The OU is a logical subblock of the entire crossbar array, which has been studied and adopted by several previous works [19] for nonideality mitigation.

D. Hardware overhead

TABLE IV
HARDWARE OVERHEAD COMPARED WITH THE ORIGINAL PE.

Component	Power (mW)	Area (μm^2)
Original	613.977	10088
AAU	92.5	172.65
WS&D	14.017	189.03
WD	30.64	188.088
Total	751.134	10637.768

In this section, we evaluate the overhead of our designed hardware at the PE-level. The structure of the PE is illustrated in Fig. 4 with eight CIM arrays, eight S&Hs, eight ADCs, and one S&A circuit. And a WS&D, a WD, and an AAU are incorporated per crossbar array. The overhead of the IRM-CSR method is evaluated using Design Compiler at a 130nm technology node and scaled down to 32nm. The overhead for all other circuits is based on metrics from FORMS [19]. The area and power overhead are shown in TABLE IV. The WS&Ds, WDs, and AAUs take up 5.17% and 18.26% of the total area and power overhead of the entire PE. And as demonstrated in the evaluations, the IRM-CSR method takes up only 1 to 2 times more crossbar array numbers in most cases. Due to the high memory capacity and energy efficiency (2 to 4 orders of magnitude higher than conventional architectures [19]) of in-memory computing, we consider our hardware energy consumption to be acceptable.

VI. CONCLUSION

Memristive accelerators hold great promise for high energy-efficient in-memory computing, while suffering from IR drop effect. In this paper, we propose an IR drop-robust mapping method for reliable memristive accelerators, which exploits the residuals and maps them iteratively to the crossbars. The corresponding mechanism of MVM and hardware support are designed for the IRM method. Last, the effectiveness of the IRM method is validated, ensuring high computation accuracy of the memristive accelerators.

REFERENCES

- [1] B. Wu *et al.*, "FADESIM: Enable Fast and Accurate Design Exploration for Memristive Accelerators Considering Nonidealities," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 4, pp. 1529–1543, Apr. 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/10729885>

- [2] N. Lepri *et al.*, “Mitigating read-program variation and IR drop by circuit architecture in RRAM-based neural network accelerators,” in *2022 IEEE International Reliability Physics Symposium (IRPS)*, Mar. 2022, pp. 3C.2–1–3C.2–6, iISSN: 1938-1891. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9764486>
- [3] B. Liu *et al.*, “Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems,” in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2014, pp. 63–70, iISSN: 1558-2434.
- [4] A. Bhattacharjee *et al.*, “Examining the Role and Limits of Batchnorm Optimization to Mitigate Diverse Hardware-noise in In-memory Computing,” in *Proceedings of the Great Lakes Symposium on VLSI 2023*, ser. GLSVLSI '23. New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 619–624. [Online]. Available: <https://dl.acm.org/doi/10.1145/3583781.3590241>
- [5] F. L. Aguirre *et al.*, “Minimization of the Line Resistance Impact on Memdiode-Based Simulations of Multilayer Perceptron Arrays Applied to Pattern Recognition,” *Journal of Low Power Electronics and Applications*, vol. 11, no. 1, p. 9, Mar. 2021, number: 1 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: <https://www.mdpi.com/2079-9268/11/1/9>
- [6] S. Gao *et al.*, “Fast Algorithms for Exact IR Drop De-Embedding in Analog Multiply–Accumulate Computing,” *IEEE Transactions on Electron Devices*, vol. 69, no. 11, pp. 6376–6383, Nov. 2022, conference Name: IEEE Transactions on Electron Devices.
- [7] S. Lee *et al.*, “Offline Training-Based Mitigation of IR Drop for ReRAM-Based Deep Neural Network Accelerators,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 2, pp. 521–532, Feb. 2023, conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.
- [8] M. E. Fouda *et al.*, “IR-QNN Framework: An IR Drop-Aware Offline Training of Quantized Crossbar Arrays,” *IEEE Access*, vol. 8, pp. 228 392–228 408, 2020, conference Name: IEEE Access.
- [9] W. Song *et al.*, “Programming memristor arrays with arbitrarily high precision for analog computing,” *Science*, vol. 383, no. 6685, pp. 903–910, Feb. 2024, publisher: American Association for the Advancement of Science. [Online]. Available: <https://www.science.org/doi/full/10.1126/science.ad9405>
- [10] M. Le Gallo *et al.*, “Precision of bit slicing with in-memory computing based on analog phase-change memory crossbars,” *Neuromorphic Computing and Engineering*, vol. 2, no. 1, p. 014009, Feb. 2022, publisher: IOP Publishing. [Online]. Available: <https://dx.doi.org/10.1088/2634-4386/ac4fb7>
- [11] S. Jain *et al.*, “RxNN: A Framework for Evaluating Deep Neural Networks on Resistive Crossbars,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 2, pp. 326–338, 2021, conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.
- [12] H. Nili *et al.*, “Hardware-intrinsic security primitives enabled by analogue state and nonlinear conductance variations in integrated memristors,” *Nature Electronics*, vol. 1, no. 3, pp. 197–202, Mar. 2018, publisher: Nature Publishing Group. [Online]. Available: <https://www.nature.com/articles/s41928-018-0039-7>
- [13] J. Woo *et al.*, “Improved Synaptic Behavior Under Identical Pulses Using AlOx/HfO2 Bilayer RRAM Array for Neuromorphic Systems,” *IEEE Electron Device Letters*, vol. 37, no. 8, pp. 994–997, Aug. 2016, conference Name: IEEE Electron Device Letters. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/7496808>
- [14] Y. Luo *et al.*, “MLP+NeuroSimV3.0: Improving On-chip Learning Performance with Device to Algorithm Optimizations,” in *Proceedings of the International Conference on Neuromorphic Systems*, ser. ICONS '19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 1–7. [Online]. Available: <https://dl.acm.org/doi/10.1145/3354265.3354266>
- [15] D. Kuzum *et al.*, “Nanoelectronic Programmable Synapses Based on Phase Change Materials for Brain-Inspired Computing,” *Nano Letters*, vol. 12, no. 5, pp. 2179–2186, May 2012, publisher: American Chemical Society. [Online]. Available: <https://doi.org/10.1021/nl201040y>
- [16] S. Lee *et al.*, “Accurate Prediction of ReRAM Crossbar Performance Under I-V Nonlinearity and IR Drop,” in *2022 IEEE 40th International Conference on Computer Design (ICCD)*, Oct. 2022, pp. 9–16, iISSN: 2576-6996. [Online]. Available: <https://ieeexplore.ieee.org/document/9978475>
- [17] B. Feinberg *et al.*, “Enabling Scientific Computing on Memristive Accelerators,” in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. Los Angeles, CA, USA: IEEE, Jun. 2018, pp. 367–382, iISSN: 2575-713X.
- [18] T. A. Davis *et al.*, “The university of Florida sparse matrix collection,” *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1:1–1:25, Dec. 2011. [Online]. Available: <https://dl.acm.org/doi/10.1145/2049662.2049663>
- [19] G. Yuan *et al.*, “FORMS: Fine-grained Polarized ReRAM-based In-situ Computation for Mixed-signal DNN Accelerator,” in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2021, pp. 265–278, iISSN: 2575-713X.