Dealing with Non-Idealities in Memristor Based Computation-In-Memory Designs

Gebregiorgis, Anteneh ; Singh, Abhairaj; Diware, Sumit ; Bishnoi, Rajendra; Hamdioui, Said

DOI
10.1109/VLSI-SoC54400.2022.9939618

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)

2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC) | 978-1-6654-9005-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/VLSI-SoC54400.2022.9939618

Dealing with Non-Idealities in Memristor Based Computation-In-Memory Designs

Anteneh Gebregiorgis, Abhairaj Singh, Sumit Diware, Rajendra Bishnoi, Said Hamdioui
Computer Engineering Lab, Delft University of Technology, Delft, The Netherlands
Email: {A.B.Gebregiorgis, A.Singh-5, S.S.Diware, R.K.Bishnoi, S.Hamdioui}@tudelft.nl

***Abstract*—Computation-In-Memory (CIM) using memristor devices provides an energy-efficient hardware implementation of arithmetic and logic operations for numerous applications, such as neuromorphic computing and database query. However, memristor-based CIM suffers from various non-idealities such as conductance drift, read disturb, wire parasitics, endurance and device degradation. These negatively impact the computation accuracy of CIM. It is therefore essential to deal with these non-idealities and fabrication imperfections in order to harness the full potential of CIM. This paper discusses the non-ideality challenges and provides potential solutions. Furthermore, the paper outlines the potential future directions for CIM architectures.**

***Index Terms*—CIM, memristor, non-idealities, variability, accuracy.**

I. Introduction

Emerging memristor-based Computation-In-Memory (CIM) has the potential to break the well-knwon Von-Neumann challenges and offer superior performance with limited energy budget [1–4]. Memristor-based CIM architectures use non-volatile devices such as RRAM to store the data while exploiting their inherent capability to perform computation on the stored data [5]. This enables CIM to circumvent the costly data movement of Von-Neumann based systems [6]. Several recent works on CIM have demonstrated that multiply-and-accumulate (MAC) operations, which are the fundamental operations in AI applications such as Deep Neural Networks (DNNs), and logic operations can be efficiently implemented on CIM crossbar [7, 8]. In addition, memristor-based CIM architectures have various advantages, such as zero leakage, non-volatility and density. However, the practical implementation of memristor-based CIM is hindered by various non-ideality issues, such as variations, conductance drift etc [6]. Therefore, addressing these non-idealities is essential to realize accurate and energy-efficient memristor-based CIM.

Several software and hardware solutions have been proposed to mitigate the impact of non-idealities on the computation accuracy of CIM blocks [9–19]. Some of the software-based solutions focus on finding an optimal mapping where less relevant values (e.g., LSB) are mapped to the non-ideal memristor devices [12, 16, 18], while others focus on retraining techniques to partially regain the non-ideality induced accuracy reduction [13, 17, 19]. Similarly, the hardware-based solutions utilize redundancy schemes to tackle the impact of non-idealities [14, 15]. However, these state-of-the-art solutions are inefficient and impose several hardware and software overheads. Therefore, understanding the nature of memristor non-idealities and providing efficient solutions at different levels of abstraction is essential to realize reliable and efficient CIM operations.

This paper highlights the challenge CIM faces from non-idealities, and presents different solutions to deal with them. First it discusses the opportunities of CIM and the non-ideality challenges that needs to be addressed for reliable operations. Then, the paper provides a detailed discussion of two major solutions at mapping level and micro-architectural level. Finally, the paper outlines future directions for reliable and energy-efficient CIM operation to deploy AI applications on resource constrained edge platforms, commonly known as edge-computing.

II. CIM opportunities

A. CIM basics

CIM is a computing paradigm where the computation (i.e., execution) of an operation is performed within the memory where the data resides. Figure 1 shows a high level micro-architecture of memristor-based CIM crossbar, where memristor such as RRAM device is used at each crossbar junction. The communication to the crossbar is realized with the support of peripheral circuits which perform different functions depending on the targeted CIM architecture; for example input/output data format conversion may require Digital-to-Analog Conversion (DAC) in row decoding part or Analog-to-Digital Conversion (ADC), dedicated sense amplifiers in the read path. The control block is responsible for the overall control of the CIM core operation.

B. CIM benefits

Memristor-based CIM has many features that make it feasible to realize ultra-low power and energy-efficient computing [6, 20]:

- **Practically zero leakage computing** [20]: The non-volatile nature of the resistive devices enables CIM to maintain the stored values in a leakage free manner when it is not operating, which solves the leakage bottleneck of SRAM-based architectures.
- **Massive parallelism** [6]: CIM provides high parallelism as typically all columns in a crossbar can be accessed

This work was supported in part by the EU H2020 grant "DAIS" that has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007273

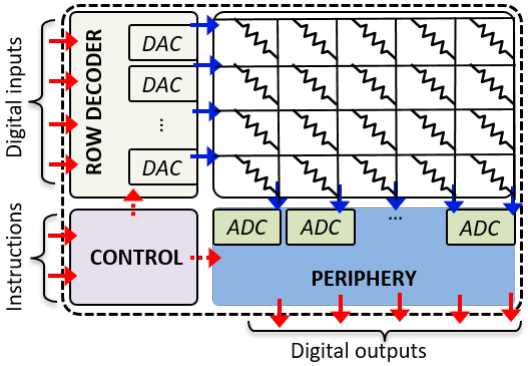


Fig. 1: CIM core architecture concept.

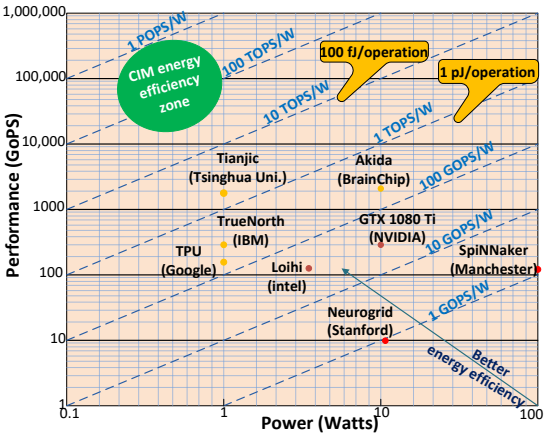


Fig. 2: CIM potential for energy efficient computing.

concurrently, leading to maximal parallelism. Moreover, the scalability of memristor technology enables to increase the number of columns per crossbar, which in turn increases the degree of parallelism CIM can offer.

- **Near zero data bandwidth requirement** [21]: Integration of storage and computation in the same physical location circumvents the bandwidth bottleneck associated with the traditional computation centered systems, which need significant data movement between the memory and processing units.
- **Extremely energy-efficient computing** [20]: The combination of non-volatility (near zero leakage), parallelism and near zero bandwidth enables CIM to offer extremely energy-efficient computing, in the order of fJ per operation, when compared to the existing solutions as shown in Figure 2.

C. CIM applications

CIM has a wide application range as it overcomes the shortcomings of conventional technologies and architectures. Applications that can benefit from CIM can be classified into two main classes based on their kernel operation. Some representative applications in these classes are discussed below.

- **Arithmetic operation intensive applications**: Arithmetic operations, in particular, Vector Matrix Multiplication (VMM) can be easily accelerated using memristive-based CIM by mapping the binary VMM kernel to a CIM crossbar array [6]. Applications involving such operations include pattern matching with automata processor [22], guided image filtering [23], image recognition and classification [24] and sparse coding [25].
- **Logic operation intensive applications**: Bulk bitwise logic operations such as AND, OR, XOR, and data comparison can be readily accelerated in CIM, where either both input vectors are mapped to the crossbar or one of them is applied as an enable signal [5]. Target applications include database queries [26, 27], Hyper-Dimensional Computing (HDC) [21] and associative memories [28].

III. CIM NON-IDEALITY CHALLENGES

Emerging memristor devices play a key role in enabling energy-efficient and highly parallel computations within the memory core itself. However, these devices suffer from several non-idealities where some occur at the production time before shipping and others, during the course of the product life cycle. We classify these as time-zero and time-dependent non-idealities, respectively.

1) **Time-zero non-idealities**

- **Variation**: Variation is the deviation of the resistance value of the memristor after programming from the expected resistance value, which can lead to incorrect computations [29–31]. Variation happens mainly due to fabrication imperfections and the stochastic nature of underlying physics. Additionally, traditional CMOS process, voltage and temperature (PVT) variations further impact the computational correctness [32].
- **Wire parasitics**: Due to the finite parasitic resistance and capacitance of the interconnect wires, signals suffer from delay mismatch and voltage degradation, which can lead to erroneous outputs [33]. For instance, in logic operations, the reference and input signals reaching the sensing circuits (for e.g., sense amplifier) suffer from delay mismatch caused by different critical paths. Additionally, the wordline degrades along the path reaching farther columns that degrades the associated current output.
- **Non-zero G_{min} error**: In the digital domain, multiplying any non-zero input with a zero weight must result in a zero output. However, when such computation is mapped to memristors as shown in Figure 3, a non-zero output current is produced when a non-zero input voltage is applied to a memristor with G_{min} conductance which represents digital zero. This phenomenon is known as non-zero G_{min} error which causes a functional mismatch between the expected digital value and the actual memristive computation result [34].

2) **Time-dependent non-idealities**

- **Endurance**: Memristors suffer from limited endurance due to the destructive nature of the programming operations [35]. For instance, in RRAMs, the material assumes presence and absence of conductive ions by forming and rupturing the conductive filament during a write operation; in PCRAMs, the material assumes amorphous to crystalline forms by aligning or dis-aligning the covalent

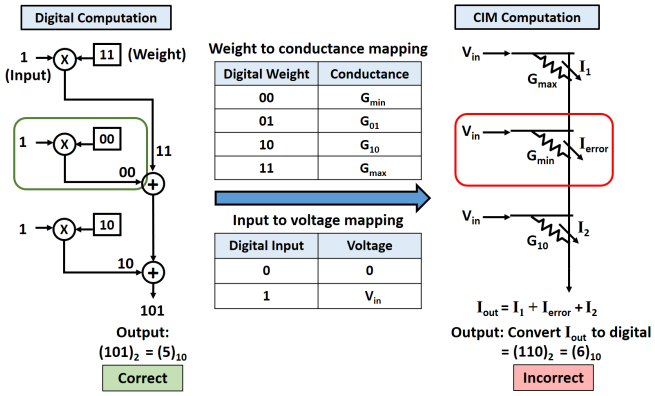


Fig. 3: Illustration of non-zero G_{min} error.

bonds. However, continuous write operations gradually degrade the ON/OFF resistance ratio of the devices, eventually leading to endurance failure [36, 37].

- **Device degradation**: Due to stress and ageing, CMOS periphery and memristors in CIM suffer from device degradation [38]. These phenomena are aggravated by high voltage of operation and temperature.
- **Conductance drift**: The conductance states of the memristors tend to drift with time and can eventually lead to unwanted bit-flips [39].
- **Read disturb**: Read disturb is a phenomenon where a correct value is returned when a read operation is performed in a cell, while the data stored by the cell is flipped by the read operation [40, 41].

These non-idealities lead to erroneous computations and incorrect functionality in CIM architecture. Hence, it is necessary to mitigate their impact. The solutions for such mitigation are discussed in the next section.

IV. Overview of Potential Solutions

Solutions for mitigating the impact of non-idealities can be developed and deployed at various abstraction levels such as device-level, circuit-level, architecture-level and application-level. Typical solutions at each of these abstraction levels are discussed below.

- **Device-level solutions** typically involve improving the memristor device structure and material composition for better characteristics to ensure well-defined distinctive resistance states. A high resistance ratio along with low variations corresponding to OFF and ON states are desired to ensure high read sensing margins.
- **Circuit-level solutions** include innovative circuit designs which can provide correct functionality in the presence of non-idealities. Solutions explore the design space of several bitcell configurations, dedicated referencing schemes and sense amplifiers for logic operations and variation-aware ADC and DAC converters. In addition, circuit solutions that deploy local and global compensation schemes can enable high robustness against conductance drift and temperature fluctuations.
- **Architecture-level solutions** can mitigate the impact of non-idealities by changing the way in which an application is mapped to CIM hardware, changing how the data flows through various CIM system components or introducing error correction mechanisms.
- **Application-level solutions** involve adapting the underlying applications so that inexact computations will suffice instead of exact ones. Applications that require one or more static-valued operands also are well suited for memristor-based CIM as endurance problem is alleviated.

The reminder of this article focuses on circuit and architecture-level solutions to mitigate effect of non-idealities.

V. Circuit-Level Solutions

In relation to performing fundamental logic operations, non-idealities such as *variations* and *wire parasitics* can be addressed at the circuit-level by deploying variation-aware reference circuits and placing them such that they adapt to delay mismatch along the row and column wire parasitics [32]. Another potential solution is based on the use of two-transistor-two-resistor (2T2R) bitcell configuration that can inherently improve the robustness against *variations* and *wire parasitics* and efficiency at the expense of reduced storage density [42]. Utilizing such bitcell configuration allows us to suppress *device degradation*, *conductance drift* and eventual unwanted bit-flip and the events of *read disturb* suffered by memristors by reducing the voltage of operation. These solutions are described in detail below.

A. Robust Logic Accelerator for 1T1R Bitcell Configuration

We have proposed an adaptive scheme to generate a reference signal V_{RL} which adopts non-idealities such as variations and wire delay mismatch from which the input signal V_{BL} suffers, during a multi-row select read-assisted logic operation [32]. Below are the design and implementation of the reference generators and the arrangement to realize the proposed robust logic accelerator.

1) Reference generator design: Reference generators are build using a combination of memristor and CMOS devices to address process, voltage and temperature (PVT) variations. The idea is to arrange memristor devices in such a way that it produces an average equivalent resistance of the two critical resistance states (decision determining states) for the desired operation. This ensures maximum possible sensing margins to realize robust operations. For instance, the logic AND reference falls in the middle of the two critical states 01/10 (i.e., one in 'OFF' and one in 'ON' state with an equivalent resistance $R_{OFF}//R_{ON}$) and 11 (i.e., both in 'ON' state with an equivalent resistance $R_{ON}//R_{ON}$). Hence the mean resistance is $\frac{R_{OFF}//R_{ON}+R_{ON}//R_{ON}}{2}$. In a similar fashion, logic OR reference signal has an effective resistance of $\frac{R_{OFF}//R_{OFF}+R_{OFF}//R_{ON}}{2}$. Figure 4(a) shows the circuit implementations of the reference generators and graphical distribution of input and reference signals V_{RO} and V_{RA} generated for the OR and AND operations, respectively.

2) Reference arrangement: Reference Arrangement is deployed to account for the wire delay mismatch due to the position of the accessed bitcell in the crossbar array. For instance, the input signal V_{BL} while accessing the bitcell near

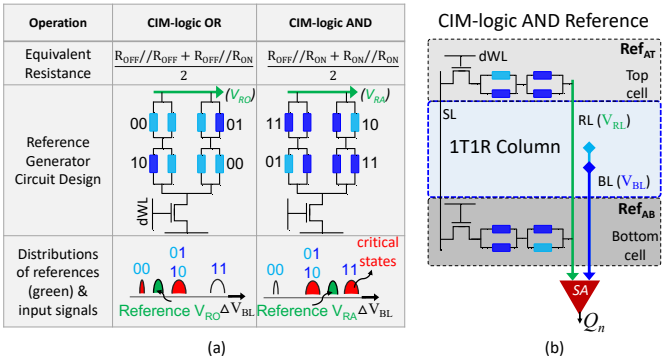


Fig. 4: (a) Reference generator for CIM-logic OR and CIM-logic AND operations. (b) Symmetrical reference cell arrangement in each column [32].

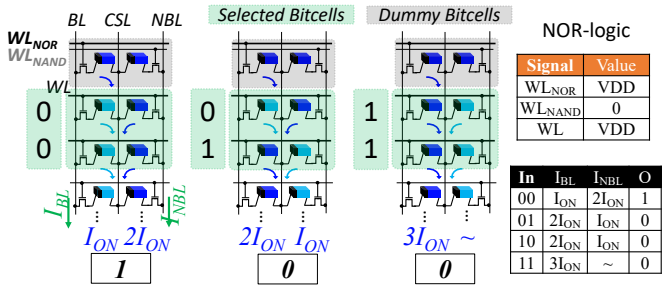


Fig. 5: Working of NOR logic operation using complementary 2T2R bitcell configuration [42].

the SA incurs by far less delay as compared with accessing the bitcell far along from the SA in the column. The idea is to place the reference generators in such as way that the reference signal V_{RL} capture similar parasitic delays as those experienced by the worst-case (top and bottom row) bitcell accesses. Hence, the reference cells (for e.g., AND operation) as described earlier, are split into two sub-cells, each with a resistance equivalent to the sum of the critical resistance states. These sub-cells are placed as the top Ref_{AT} and the bottom Ref_{AB} cells in each column, as shown in Figure 4(b). This arrangement serves three purposes: (1) The parallel combination of the two reference sub-cells effectively ensures an average resistance of the two critical resistance states related to AND logic operation. (2) The placement at the top and bottom ensures row-wise tracking as the reference signal encounters an average delay of the worst-case bitcell access delays. (3) Placed in each column, these cells ensure column-wise tracking since they suffer with similar WL voltage degradation as experienced by the accessed bitcells. Similarly, logic OR reference circuit is split into two sub-cells.

These schemes have been validated by performing extensive SPICE simulations in TSMC 28 nm technology involving 3σ variation analysis for the global PVT (worst-case corner) cases while taking into account memristor, sense amplifier and reference signal variations. Additionally, silicon-verified wire parasitics have been incorporated in a 128x512 memristor-based CIM array. Results show that with negligible area overhead, this architecture for binary neural networks (BNN) achieves up to 17.8 TOPS/W on the MNIST dataset and 130× performance improvement for the text encryption compared to the software implementation on Intel Haswell processor.

B. Referencing-in-Array for 2T2R Bitcell Configuration

We have presented a voltage-based differential referencing-in-array scheme that improves the accuracy of logic operations [42]. The scheme makes use of a 2T2R cell configuration to create a complementary bitcell structure that inherently acts also as a reference during the operation execution; this results in a high sensing margin.

Cell structure is based on 2T2R bitcell configuration as shown in Figure 5. Data-bit '1' is represented by dark blue (LRS) and the complementary data-bit as light blue (HRS), and the pass transistors connect these memristors to bitline (BL) and negative bitline (NBL). The bitcell has a wordline (WL) and a common select line (CSL), connecting the top electrode of one memristor to the bottom electrode of the other.

Logic operations Figure 5(b) illustrates NOR logic operation performed using a dummy row of 2T2R bitcells. The idea is to bias the BL(NBL) side with an ON current while performing NOR(NAND) operation. The dummy cells require two modifications: (1) they need to have both memristor devices of the cell in the LRS; (2) to get two independent WLs, namely WL_{NOR} and WL_{NAND}, connected to the BL and NBL-sided pass transistors, respectively. The one-time configuration of dummy cells requires individual memristor device programming to LRS by selecting the dedicated WLs one at a time. To perform a NOR function, two rows storing operands are activated along with WL_{NOR} in the dummy row.

For simplicity, ON currents are assumed to be I_{ON}, and OFF currents to be zero. In case both operands are in the OFF state (00); the BL/NBL discharge currents are I_{ON}/$2I_{ON}$, resulting in '1' as NOR result. In case the two operands are in different states (01/10); the BL/NBL discharge currents are $2I_{ON}$/I_{ON}, implying a value '0'. In case both operands are in the ON state (11); the BL/NBL discharge currents are $3I_{ON}$/0, leading to a value '0'. In a similar manner, the distribution of BL/NBL currents for NAND can be derived for each of the above cases.

Results show that this scheme accurately and reliably performs (N)OR/(N)AND operation whiling offering 11.4X better energy-efficiency compared to state-of-the-art solutions.

VI. Architecture-Level Solution

CIM-based neural network hardware typically employs bit-slicing scheme to map weights into a memristor crossbar, as the bit-capacity of memristor devices is less than the bit-width demanded by neural network applications [43, 44]. This involves division of neural network weights and inputs into smaller chunks called slices which are mapped to conductances and voltages, respectively. Column currents resulting from the interaction of time-multiplexed voltage inputs with sliced conductances are converted to digital, then shifted and added to get the final output. The CIM architectures using conventional bit-slicing scheme suffer from non-zero minimum conductance (G_{min}) error, which degrades the classification accuracy of the bit-slicing CIM-based neural network hardware despite having high numeric precision.

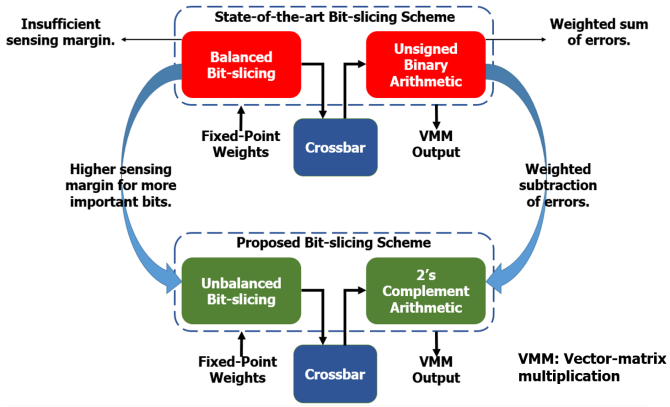


Fig. 6: Overview of bit-slicing schemes [45].

A bit-slicing scheme consists of two fundamental components: 1) bit-slicing logic which determines how the slices are created, and 2) arithmetic which determines how the partial outputs from sliced columns are combined. The *balanced* bit-slicing (BBS) logic in state-of-the-art bit-slicing schemes provides low sensing margin resulting in significant impact of non-zero G_{min} errors, while unsigned binary arithmetic in these schemes leads to high accumulative non-zero G_{min} error on combining the partial outputs. We have proposed an unbalanced bit-slicing (UBS) scheme in [45] which changes the way in which neural network weights are mapped to bit-slicing CIM crossbar to mitigate the impact of non-zero G_{min} error. The proposed UBS scheme provides high sensing margin for important bits (MSBs) by using a memristor with n-bit maximum capacity as an m-bit memory-cell (slice) where $m<n$. This provides sufficient sensing margin to the MSB column output and make them immune to non-zero G_{min} error as shown in Figure 7. This suffices for good accuracy due to the robustness of neural networks to minor computational fluctuations i.e. errors in less important bits (LSBs). Moreover, use of 2's complement arithmetic in [45] leads to reduction in accumulative non-zero G_{min} error after combining the partial outputs due to weighted subtraction as shown in Eq. 1a and Eq. 1b obtained for Figure 8 using 8-bit weights with maximum 2 bits/memristor as an example. The conductance subscripts indicate binary slice value. The digital outputs of

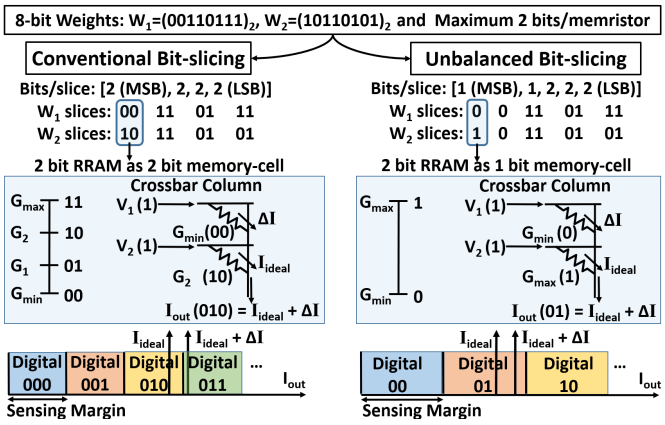
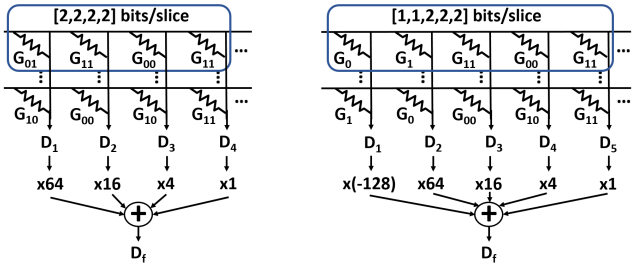


Fig. 7: Impact of sensing margin on bit-slicing schemes [45].



(a) (b)

Fig. 8: Partial output accumulation in (a) BBS, (b) UBS [45].

the columns are denoted by D_i, while the accumulated digital outputs are indicated by D_f. D_i=T_i+E_i, where T_i is the ideal output value and E_i the error due to non-zero G_{min}. Similarly, D_f=$T_f + E_f$, where T_f is the ideal accumulated output value and E_f the accumulated error due to non-zero G_{min}. .

$$E_f = 64{\cdot}E_1 + 16{\cdot}E_2 + 4{\cdot}E_3 + E_4 \quad (1a)$$

$$E_f = (-128){\cdot}E_1 + 64{\cdot}E_2 + 16{\cdot}E_3 + 4{\cdot}E_4 + E_5 \quad (1b)$$

If we assume each column has all memristors of G_{min} value, then all of them have the maximum non-zero G_{min} error E_{max}. In this scenario, BBS leads to E_f = 85×E_{max} and UBS leads to E_f = −43×E_{max}. Thus, UBS halves the non-zero G_{min} error. The sign does not matter as each digital state has a finite window as shown in Figure 7 and only the magnitude matters as it decides if output exceeds the window boundaries.

Owing to the cumulative effect of high MSB sensing margin and 2's complement arithmetic on non-zero G_{min} error, unbalanced bit-slicing scheme achieves up to 8.8× and 1.8× classification accuracy compared to state-of-the-art CIM architectures for single-bit memristors and two-bit memristors respectively, at reasonable energy overheads arising due to extra columns.

VII. Conclusions and Future Directions

Emerging AI and big-data applications demand energy-efficient, compact and reliable hardware. In recent years, specialized hardware (like Tensor processor units) have been developed to cope up with these demands and deliver better performance than GPUs. However, these specialized hardware accelerators are costly to be deployed in resource-constrained edge platforms as they are not energy-efficient. Clearly, the demand of energy-efficiency cannot be full-filled by the existing hardware. Memristor-base CIM architectures have the potential to close this gap as memristive devices have several advantageous features such as non-volatility, scalability, high density, CMOS compatibility, etc. Despite these advantages, CIM architectures face several non-ideality challenges. Therefore, addressing the non-ideality challenges is essential to unlock the full potential of CIM and deploy AI application on resource-constrained edge platforms.

References

[1] M. T. Bohr *et al.*, "Cmos scaling trends and beyond," *IEEE Micro*, 2017.

[2] M. A. Zidan *et al.*, "The future of electronics based on memristive systems," *Nature Electronics*, 2018.

[3] Hsu *et al.*, "Ai edge devices using computing-in-memory and processing-in-sensor: From system to device," in *IEDM*, 2019.

[4] Z. Zhou *et al.*, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *IEEE*, 2019.

[5] H. A. D. Nguyen *et al.*, "A classification of memory-centric computing," *JETC*, 2020.

[6] A. Singh *et al.*, "Low-power memristor-based computing for edge-ai applications," in *ISCAS*, 2021.

[7] M. Le Gallo *et al.*, "Mixed-precision in-memory computing," *Nature Electronics*, 2018.

[8] D. Ielmini *et al.*, "In-memory computing with resistive switching devices," *Nature Electronics*, 2018.

[9] B. Liu *et al.*, "Reduction and ir-drop compensations techniques for reliable neuromorphic computing systems," in *ICCAD*, 2014.

[10] M. Hu *et al.*, "Bsb training scheme implementation on memristor-based circuit," in *Computational Intelligence for Security and Defense Applications*, 2013.

[11] B. Liu *et al.*, "Vortex: Variation-aware training for memristor x-bar," in *DAC*, 2015.

[12] L. Chen *et al.*, "Accelerator-friendly neural-network training: Learning variations and defects in rram crossbar," in *DATE*, 2017.

[13] C. Liu *et al.*, "Rescuing memristor-based neuromorphic design with high defects," in *DAC*, 2017.

[14] W.-Q. Pan *et al.*, "Strategies to improve the accuracy of memristor-based convolutional neural networks," *TED*, 2020.

[15] D. Joksas *et al.*, "Committee machines—a universal method to deal with non-idealities in memristor-based neural networks," *Nature communications*, 2020.

[16] F. M. Bayat *et al.*, "Memristor-based perceptron classifier: Increasing complexity and coping with imperfect hardware," in *ICCAD*, 2017.

[17] D. Joksas *et al.*, "Nonideality-aware training for accurate and robust low-power memristive neural networks," *Advanced Science*, 2022.

[18] A. Mehonic *et al.*, "Mitigating non-idealities of memristive-based artificial neural networks-an algorithmic approach," in *EDTM*, 2022.

[19] D. Gaol *et al.*, "Reliable memristor-based neuromorphic design using variation-and defect-aware training," in *ICCAD*, 2021.

[20] J. Yu *et al.*, "The power of computation-in-memory based on memristive devices," in *ASP-DAC*, 2020.

[21] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive computation*, 2009.

[22] A. Gebregiorgis *et al.*, "A survey on memory-centric computer architectures," *JETC*, 2022.

[23] K. He *et al.*, "Guided image filtering," *transactions on pattern analysis and machine intelligence*, 2012.

[24] M. Komar *et al.*, "Deep neural network for image recognition based on the caffe framework," in *DSMP*, 2018.

[25] P. M. Sheridan *et al.*, "Sparse coding with memristor networks," *Nature nanotechnology*, 2017.

[26] M. Imani *et al.*, "Nvquery: Efficient query processing in nonvolatile memory," *TCAD*, 2018.

[27] I. Giannopoulos *et al.*, "In-Memory Database Query," *Advanced Intelligent Systems*, 2020.

[28] R. Karam *et al.*, "Emerging trends in design and applications of memory-based computing and content-addressable memories," *Proceedings of the IEEE*, 2015.

[29] A. Singh *et al.*, "Accelerating rram testing with low-cost computation-in-memory based dft," in *ITC*, 2022.

[30] J.-H. Lee *et al.*, "Exploring cycle-to-cycle and device-to-device variation tolerance in MLC storage-based neural network training," *TED*, 2019.

[31] Y. Zhang *et al.*, "STT-RAM cell optimization considering MTJ and CMOS variations," *TMAG*, 2011.

[32] A. Singh *et al.*, "Cim-based robust logic accelerator using 28 nm stt-mram characterization chip tape-out," in *AICAS*, 2022.

[33] Y. Jeong *et al.*, "Parasitic effect analysis in memristor-array-based neuromorphic systems," *Nanotech*, 2018.

[34] P. Chen *et al.*, "Technological Benchmark of Analog Synaptic Devices for Neuroinspired Architectures," *IDT*, 2019.

[35] M. Fieback *et al.*, "Testing scouting logic-based computation-in-memory architectures," in *ETS*, 2020.

[36] M. Zhao *et al.*, "Characterizing endurance degradation of incremental switching in analog rram for neuromorphic systems," in *IEDM*, 2018.

[37] L. Martin-Monier *et al.*, "Endurance of chalcogenide optical phase change materials: a review," *Optical Materials Express*, 2022.

[38] M. Fieback *et al.*, "Defects, fault modeling, and test development framework for rrams," *JETC*, 2022.

[39] S. Ambrogio *et al.*, "Reducing the impact of phase-change memory conductance drift on the inference of large-scale hardware neural networks," in *IEDM*, 2019.

[40] W. Shim *et al.*, "Investigation of read disturb and bipolar read scheme on multilevel rram-based deep learning inference engine," *TED*, 2020.

[41] E. I. Vatajelu *et al.*, "Challenges and solutions in emerging memory testing," *TETC*, 2017.

[42] A. Singh *et al.*, "Referencing-in-array scheme for rram-based cim architecture," in *DATE*, 2022.

[43] A. Shafiee *et al.*, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *ISCA*, 2016.

[44] A. Ankit *et al.*, "PUMA: A Programmable Ultra-efficient Memristor-based Accelerator for Machine Learning Inference," in *ASPLOS*, 2019.

[45] S. Diware *et al.*, "Unbalanced bit-slicing scheme for accurate memristor-based neural network architecture," in *AICAS*, 2021.