

Accurate and Energy-Efficient Bit-Slicing for RRAM-Based Neural Networks

Diware, Sumit; Singh, Abhairaj ; Gebregiorgis, Anteneh; Joshi, Rajiv V.; Hamdioui, Said; Bishnoi, Rajendra

DOI

[10.1109/TETCI.2022.3191397](https://doi.org/10.1109/TETCI.2022.3191397)

Publication date

2023

Document Version

Final published version

Published in

IEEE Transactions on Emerging Topics in Computational Intelligence

Citation (APA)

Diware, S., Singh, A., Gebregiorgis, A., Joshi, R. V., Hamdioui, S., & Bishnoi, R. (2023). Accurate and Energy-Efficient Bit-Slicing for RRAM-Based Neural Networks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(1), 164 - 177. Article 9840507. <https://doi.org/10.1109/TETCI.2022.3191397>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.




Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

<https://www.openaccess.nl/en/you-share-we-take-care>

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Accurate and Energy-Efficient Bit-Slicing for RRAM-Based Neural Networks

Sumit Diware , Abhairaj Singh, Anteneh Gebregiorgis , *Member, IEEE*, Rajiv V. Joshi, *Fellow, IEEE*, Said Hamdioui , *Senior Member, IEEE*, and Rajendra Bishnoi

Abstract—*Computation-in-memory* (CIM) paradigm leverages emerging memory technologies such as resistive random access memories (RRAMs) to process the data within the memory itself. This alleviates the memory-processor bottleneck resulting in much higher hardware efficiency compared to von-Neumann architecture-based conventional hardware. Hence, CIM becomes an attractive alternative for applications like neural networks which require a huge number of data transfer operations in conventional hardware. CIM-based neural networks typically employ bit-slicing scheme which represents a single neural weight using multiple RRAM devices (called slices) to meet the high bit-precision demand. However, such neural networks suffer from significant accuracy degradation due to non-zero G_{\min} error where a zero weight in the neural network is represented by an RRAM device with a non-zero conductance. This paper proposes an unbalanced bit-slicing scheme to mitigate the impact of non-zero G_{\min} error. It achieves this by allocating appropriate sensing margins for different slices based on their binary positions. It also tunes the sensing margins to meet the demands of either high accuracy or energy-efficiency. The sensing margin allocation is supported by 2's complement arithmetic which further reduces the influence of non-zero G_{\min} error. Simulation results show that our proposed scheme achieves up to $7.3\times$ accuracy and up to $7.8\times$ correct operations per unit energy consumption compared to state-of-the-art.

Index Terms—*Computation-in-memory*, bit-slicing, neural networks, non-zero G_{\min} error, conductance variation, non-idealities.

I. INTRODUCTION

WITH recent advancements in the field of Artificial Intelligence (AI), neural network-based cognitive applications such as image recognition have become an integral part of many real-world products and services [1]–[4]. Existing computing systems such as CPUs [5], GPUs [6] and AI-oriented ASICs like TPUs [7] are developed using CMOS-based von-Neumann architecture. The physical separation of memory and computation

units in these systems results in a huge number of data transfers to execute *vector-matrix multiplication* (VMM) operations for neural network applications leading to degradation of performance and energy-efficiency [8]–[11]. Moreover, CMOS technology is struggling with challenges such as excessive sub-threshold leakage and scalability issues [12]–[14]. *Computation-in-memory* (CIM) utilizes emerging memory technologies such as resistive random access memories (RRAMs) to alleviate the data transfer bottleneck by performing in-place computations within the memory using simple circuit laws [15]–[18]. Moreover, RRAM devices used in CIM are non-volatile (leakage-free), highly scalable and small in size making CIM a more efficient alternative to the conventional hardware [19]–[21]. Despite these advantages, CIM architectures face the limitation of not being able to support the high bit-precision demands of neural network applications [22]. Therefore, a bit-slicing scheme [23], [24] is commonly employed in CIM architectures where multiple RRAM devices represent a full-precision neural network weight. Bit-slicing CIM architectures represent a zero weight in the neural network using an RRAM device with non-zero conductance equal to the minimum possible RRAM device conductance denoted as G_{\min} [25]. Multiplication of any non-zero digital input with a zero digital weight must produce a zero digital output. However, a non-zero output current is produced when a non-zero input in the form of a voltage is applied to an RRAM with G_{\min} conductance. This is known as non-zero G_{\min} error, which violates the functional equivalence between the digital output and CIM output, leading to errors in VMM and degraded neural network accuracy.

State-of-the-art bit-slicing CIM architectures cannot provide good accuracy in presence of non-zero G_{\min} error. For instance, ISAAC [23] and PUMA [24] use *balanced* bit-slicing (BBS) scheme which suffers from accuracy degradation due to non-zero G_{\min} error. PANTHER [22] proposes *heterogeneous* bit-slicing (HBS) scheme, which is an extension of BBS and thereby struggles to provide good accuracy in the presence of non-zero G_{\min} error. *Current subtraction technique* (CST) [25] can be utilized to mitigate the impact of non-zero G_{\min} on BBS and HBS. However, it becomes less effective when conductance variation is considered along with non-zero G_{\min} error. Hence, there is a strong need for an effective solution to mitigate the impact of non-zero G_{\min} error on bit-slicing CIM architectures while taking conductance variation into account.

We propose an *unbalanced* bit-slicing (UBS) scheme for CIM architectures in which appropriate sensing margin is provided to

Manuscript received 13 October 2021; revised 23 February 2022; accepted 11 July 2022. Date of publication 26 July 2022; date of current version 30 January 2023. This work was supported by ECSEL Joint Undertaking (JU) that has received funding from the EU H2020 through “DAIS” under Grant Agreement 101007273. (*Corresponding author: Sumit Diware.*)

Sumit Diware, Abhairaj Singh, Anteneh Gebregiorgis, Said Hamdioui, and Rajendra Bishnoi are with the Computer Engineering Department, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: s.s.diware@tudelft.nl; a.singh-5@tudelft.nl; a.b.gebregiorgis@tudelft.nl; s.hamdioui@tudelft.nl; r.k.bishnoi@tudelft.nl).

Rajiv V. Joshi is with the IBM Thomas J. Watson Research Centre, Yorktown Heights, NY 10598 USA (e-mail: rvjoshi@us.ibm.com).

Digital Object Identifier 10.1109/TETCI.2022.3191397

different slices by adjusting their bit-size to reduce the impact of non-zero G_{\min} error in presence of conductance variation. In addition, UBS is supported with 2's complement arithmetic whose inherent differential nature further helps in this mitigation task leading to improved neural network accuracy. A preliminary version of this work was published in [26] which assigns higher sensing margin for more important bits i.e. most significant bits (MSBs) to achieve high accuracy at the expense of energy overheads. In this paper, we extend our work with a new variant of UBS which allocates just good enough sensing margin to the slices to improve the energy-efficiency by reducing the hardware requirements while retaining the accuracy benefits. Moreover, we provide an algorithm for optimal energy-efficient slice size selection which leverages an inherent accuracy versus energy tradeoff. We also demonstrate the effectiveness of UBS across different datasets and neural networks. Our overall contributions for this paper are as follows:

- We propose an unbalanced bit-slicing scheme which provisions high sensing margin for more important slices to achieve high accuracy in presence of non-zero G_{\min} error and an algorithm to find slice sizes for optimal accuracy under given resource constraints.
- We develop a methodology to tailor the unbalanced bit-slicing scheme for energy-efficiency by constraining the sensing margin for slices in order to reduce the hardware resources while maintaining good accuracy.
- We present a holistic solution consisting of unbalanced bit-slicing logic and 2's complement arithmetic which mitigates non-zero G_{\min} error impact in presence of conductance variation.
- We demonstrate the effectiveness of the proposed unbalanced bit-slicing scheme by performing comprehensive analysis and comparison with state-of-the-art across different datasets and neural networks.

Simulation results across five datasets using fully-connected neural network and convolutional neural network show that UBS achieves up to $7.3\times$ accuracy and up to $7.8\times$ correct operations per unit energy consumption compared to state-of-the-art with reasonable overheads.

The rest of this paper is organized as follows. Section II presents the fundamentals of CIM-based neural network implementation. Challenges in bit-slicing CIM architectures are discussed in Section III. Section IV provides details of the proposed bit-slicing scheme. The simulation setup and experiments are described in Section V, followed by simulation results in Section VI. Finally, Section VII concludes the paper.

II. NEURAL NETWORKS USING COMPUTATION-IN-MEMORY

A. Computation-in-Memory (CIM) Architecture

Vector-matrix multiplication (VMM) operations account for more than 75% of the computations and energy consumption in neural networks [27]. Computation-in-memory (CIM) can act as a more efficient alternative to von-Neumann architecture for implementing VMM in neural network hardware [22]–[24]. VMM operation between two neural network layers can be

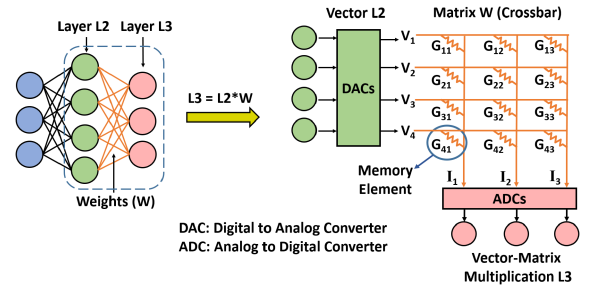


Fig. 1. CIM-based vector-matrix multiplication for neural networks.

mapped to a CIM architecture as shown in Fig. 1. CIM architecture uses memory elements which store the data in the form of conductance. They are arranged in a grid-like structure known as crossbar. The crossbar operates in analog domain and interacts with other digital components in the system using periphery circuits like digital-to-analog converters (DACs) and analog-to-digital converters (ADCs). Weight matrix is mapped to conductances (G 's) in the crossbar and inputs are applied as voltages (V 's) using DACs. This results in a flow of current through all the G 's as per Ohm's law, which is equivalent to element-wise multiplication of V 's and G 's. Currents from G 's belonging to the same column get accumulated according to Kirchhoff's law, leading to output currents (I 's). Thus, each column performs a multiply-and-accumulate operation in analog domain. Such multiply-and-accumulate operations across all the columns represent one VMM operation. As all the columns produce the outputs simultaneously, VMM is performed with $O(1)$ time complexity. The column currents are then converted to digital outputs using ADCs and sent to other system components for further processing or storage.

B. RRAM Device Technology

Resistive random access memory (RRAM) has gained notable attention as memory element in CIM due to its high switching speed, non-volatile data storage and high scalability [28]. RRAM device a.k.a memristor consists of an oxide material sandwiched between two metal electrodes. It exhibits a high-resistance state (HRS) and a low-resistance state (LRS) which can be used to store data as 0 and 1. The switching from HRS to LRS is called "SET", whereas that from LRS to HRS is called "RESET". Set voltage (V_{SET}) creates conductive path called filament to increase the conductivity of oxide layer leading to LRS. Reset voltage (V_{RESET}) causes rupture of the conductive filament which reduces oxide layer conductivity resulting in HRS. Both SET and RESET processes are depicted in Fig. 2(a) and (b). Moreover, multiple HRS/LRS states can be achieved by controlling the extent of filament creation or rupture which is called multi-level cell (MLC) operation [29]. This allows storing multiple bits of information in a single RRAM device. In order to read an RRAM device i.e. to detect its resistance state, a very small voltage V_{READ} ($V_{\text{READ}} \ll |V_{\text{SET}}|$ and $V_{\text{READ}} \ll |V_{\text{RESET}}|$) is applied across it and the resulting current is sensed. A small sensed current means device is in HRS and high sensed

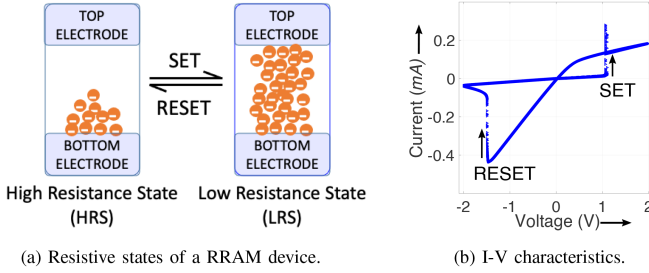


Fig. 2. RRAM device technology.

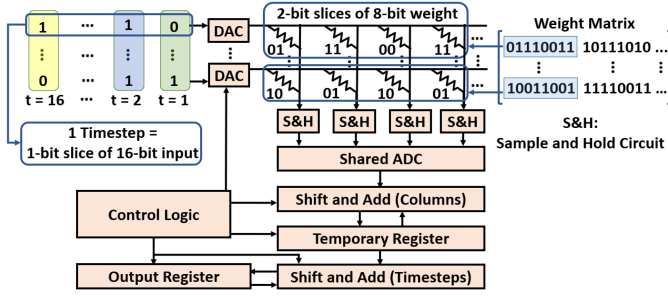


Fig. 3. RRAM-based CIM architecture with bit-slicing.

current means device is in LRS. Read operation can be similarly extended to MLCs.

C. Bit-Slicing Concept in CIM

CIM architectures cannot directly support high bit-precision demands of neural networks. This is due to the fact that bit-capacity of RRAM devices is typically less than bit-size needed for neural network weights. Moreover, full-precision neural network inputs and outputs require digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) with high resolutions. They are very expensive in terms of energy and area which significantly reduces the hardware efficiency benefit of CIM [22], [23]. To overcome these problems, bit-slicing [22]–[24] is utilized in CIM architectures, which is the process of splitting the full-precision neural network weights and inputs into smaller bit-size chunks called slices as shown in Fig. 3. 2-bit slices of a 8-bit weight are converted to conductances and mapped to RRAMs in different crossbar columns. 1-bit slices of a 16-bit input are converted to voltages and mapped to different time-steps in which they are applied to the crossbar. Column currents resulting from time-multiplexed voltage inputs are converted to digital values and undergo shift-and-add operations across the columns as well as the time-steps to obtain the full-precision output.

III. CHALLENGES IN BIT-SLICING CIM ARCHITECTURES

A. Non-Zero G_{\min} Error

Bit-slicing CIM architectures represent a zero weight-slice in the neural network using an RRAM device with non-zero conductance that is equal to the minimum possible RRAM device conductance (G_{\min}). In the digital domain, multiplying any non-zero input with a zero weight must result in a zero

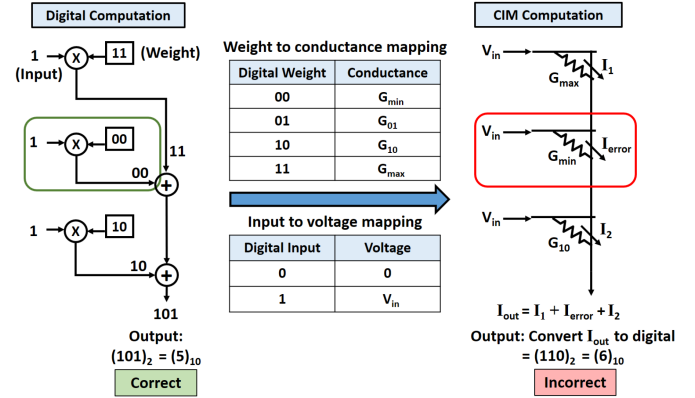
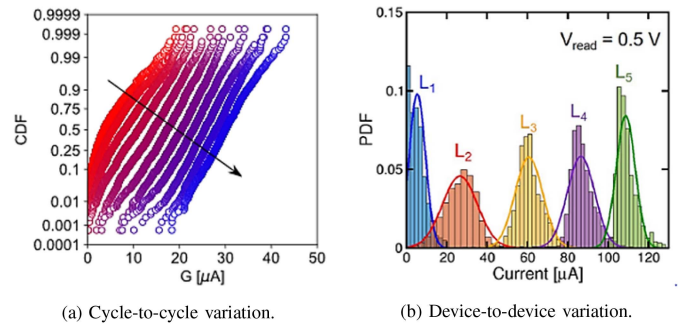
Fig. 4. Illustration of non-zero G_{\min} error.

Fig. 5. Conductance variation [30]. L's denote five programmable conductance levels for every RRAM device in the array.

output. However, when such digital computation is mapped to CIM as shown in Fig. 4, a non-zero output current is produced when a non-zero input in the form of a voltage is applied to an RRAM with G_{\min} conductance. This is known as non-zero G_{\min} error, which creates a functional mismatch between digital output and CIM output resulting in erroneous VMM operation and degraded neural network accuracy.

B. Conductance Variation

Conductance variation in RRAM devices is composed of a temporal component known as cycle-to-cycle variation and a spatial component known as device-to-device variation. Cycle-to-cycle variation refers to the fact that the same RRAM device exhibits different programmed conductance at different points in time under identical programming conditions [31]. This variation arises due to the stochastic nature of underlying physics of RRAM device which involves formation and rupture of the conductive filament. On the other hand, device-to-device variation refers to the fact that different RRAM devices exhibit different conductance characteristics under identical programming conditions. It arises from fabrication imperfections e.g. oxide thickness fluctuations leading to variable thickness and cross-section area across different RRAM devices [32]. Cycle-to-cycle variation for an RRAM device and device-to-device variation for 1 kB array of RRAM devices are shown in Fig. 5.

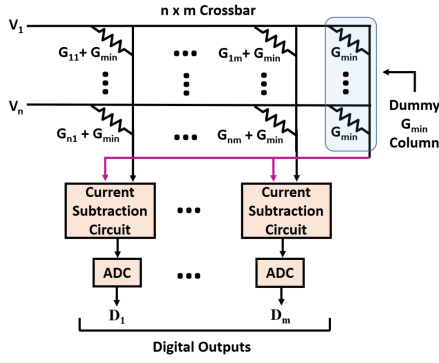


Fig. 6. Current subtraction technique (CST) as described in [25].

Conductance variation causes deviation from the expected current contribution of RRAM devices leading to error prone VMM and degradation of neural network accuracy.

C. Existing Solutions and Their Limitations

CIM architectures employ various bit-slicing schemes to express the full-precision weights using multiple RRAM devices. For instance, ISAAC [23] and PUMA [24] both use balanced bit-slicing (BBS) scheme which divides the full-precision weight into equal chunks having bit-sizes same as the RRAM bit-capacity. For example, in case of 8-bit weight and 2-bit RRAM, BBS produces four slices of 2-bit size. This can be expressed as [2, 2, 2, 2] bits/slice, where the indices go from the most significant bit (MSB) slice on the left to the least significant bit (LSB) slice on the right. However, the sensing margin in BBS is not sufficient to deal with non-zero G_{min} error. Moreover, the underlying crossbar arithmetic results in high accumulated error in the final output when the partial results from the slices are combined, leading to erroneous VMM and degraded neural network accuracy. Furthermore, PANTHER [22], which is an extension of PUMA, leverages a *heterogeneous* bit-slicing (HBS) scheme which allocates more bits for central slices. For example, in case of 8-bit weight and 2-bit RRAM, HBS will lead to [1, 1, 2, 2, 1, 1] bits/slice. HBS also suffers from accuracy degradation due to non-zero G_{min} error as it uses the same weight encoding and underlying crossbar arithmetic as that in BBS. Moreover, HBS is intended for optimizing weight update operations during neural network training and not VMM operations during inference. The current subtraction technique (CST) [25] can be utilized to mitigate the impact of non-zero G_{min} error on BBS and HBS to improve the neural network accuracy. CST adds an offset equal to G_{min} to all conductance states and uses a dummy crossbar column having only G_{min} conductances as shown in Fig. 6. Thus, it can reduce the non-zero G_{min} error impact by subtracting the common off-state currents between the crossbar columns and the dummy column. However, this technique is effective only when the conductance states exhibit values that are close to the ideal values, but in reality, these values can be significantly impacted due to conductance variations. Hence, CST struggles to provide high neural network accuracy in presence of conductance variation. In this work, we improve the neural network accuracy by addressing the non-zero

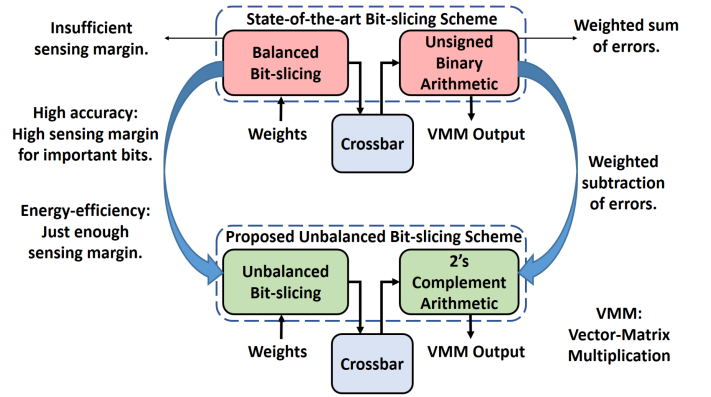


Fig. 7. Overview of state-of-the-art and proposed bit-slicing schemes.

G_{min} error problem in presence of the conductance variation using a novel bit-slicing scheme and crossbar arithmetic.

IV. PROPOSED UNBALANCED BIT-SLICING

A. Overview

A bit-slicing scheme consists of bit-slicing logic and associated crossbar arithmetic. The bit-slicing logic determines how a full-precision neural network weight is split into smaller slices, whereas the crossbar arithmetic governs the way in which the partial outputs from the crossbar columns are combined to obtain the final full-precision output. Along the same lines, an overview of the state-of-the-art as well as the proposed bit-slicing schemes is shown in Fig. 7. For the state-of-the-art bit-slicing scheme, balanced bit-slicing logic splits the full-precision neural network weight into equal-sized (*balanced*) slices and unsigned binary arithmetic is used to combine the column-wise partial outputs. Whereas, in our proposed unbalanced bit-slicing (UBS) scheme, we split the full-precision neural network weight into a mix of equal-sized and unequal-sized (*unbalanced*) slices in such a way that the resulting allocated sensing margins minimize the impact of non-zero G_{min} error. The sensing margin allocation strategy in UBS can be adapted based on whether the goal is to achieve high accuracy or high energy-efficiency. If the target is to achieve high accuracy, UBS delivers high sensing margin to more important bits (MSBs). On the other hand, if energy-efficiency is the primary goal then UBS provides just enough sensing margins to the slices to reduce the hardware resource requirements resulting in a high energy-efficiency while maintaining a sufficiently good accuracy. Moreover, UBS uses 2's complement arithmetic which results in negative scaling factor for the column containing MSB slices when combining the column-wise partial outputs using shift-and-add operations. The full-precision output denoted as D_{acc} is obtained by performing shift-and-add operations on n columns as per (1a), where D_i s and S_i s denote the column-wise partial outputs and scaling factors respectively.

$$D_{acc} = \sum_{i=1}^n S_i \cdot D_i \quad (1a)$$

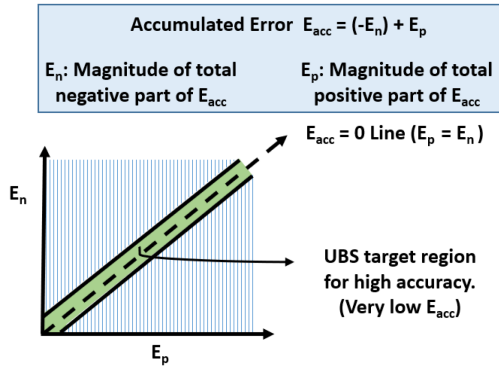


Fig. 8. Error reduction approach for high accuracy.

$$D_{acc} = \sum_{i=1}^n S_i \cdot T_i + \sum_{i=1}^n S_i \cdot E_i \quad (1b)$$

$$E_{acc} = \sum_{i=1}^n S_i \cdot E_i = \left(-S_1 \cdot E_1 + \sum_{i=2}^n S_i \cdot E_i \right) \quad (1c)$$

Expressing $D_i = T_i + E_i$ leads to (1b) for D_{acc} , where T_i is the ideal column output ($G_{min} = 0$ scenario) and E_i is the error due to non-zero G_{min} . The summation over E_i in (1b) gives the accumulated non-zero G_{min} error denoted as E_{acc} which is present in the full-precision output D_{acc} . The contribution of the negatively scaled 2's complement MSB column ($i = 1$) towards E_{acc} can be separated from the other positively scaled columns ($i = 2$ to n) as shown in (1c), where E_1 is the error in MSB column and $-S_1$ is the MSB column scaling factor where $S_1 > 0$ denotes magnitude of the MSB column scaling factor. It is clear from (1c) that the negative MSB column scaling factor reduces the overall accumulated error E_{acc} in the final output due to weighted subtraction of column-wise errors. However, such weighted subtraction will not be perfectly zero resulting in a non-zero accumulated error which can be large enough to cause substantial deviation from the correct VMM computation. The sensing margins provided by the unbalanced slice sizes in UBS ensure that this accumulated error remains small. Thus, sensing margin allocation and 2's complement arithmetic work together to minimize the impact of non-zero G_{min} error for improved neural network accuracy. In the next subsections, we first describe the details of UBS for achieving high accuracy followed by its more energy-efficient variant.

B. Unbalanced Bit-Slicing Scheme for High Accuracy

1) *Bit-Slicing Logic*: Goal of the bit-slicing logic is to minimize the accumulated error represented by E_{acc} in (1c) that remains after combining the column-wise partial outputs using 2's complement arithmetic. It is clear that the accumulated error can be minimized by achieving a good matching between the magnitudes of negatively scaled error in MSB column and sum of the positively scaled errors in the rest of the columns as illustrated in Fig. 8. If all the column-wise errors (E_i in (1c), $i = 1$ to n) are individually large, then the mismatch between the scaled errors will be large leading to a high accumulated error.

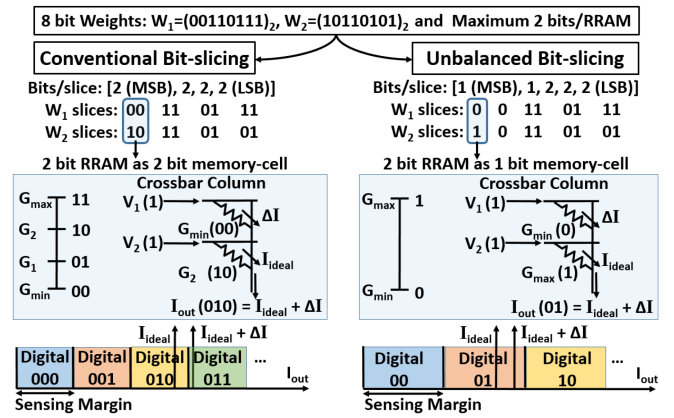


Fig. 9. Illustration of the impact of sensing margin on bit-slicing schemes.

Algorithm 1: UBS Slice Configuration Selection for High Accuracy.

input : CIM system architecture (A), Energy constraint (C), RRAM bit-capacity (B), Bits per full-precision neural weight (N)

output: UBS slice configuration for high accuracy (S)

- 1 FSC \leftarrow fundamental_slice_config(B, N);
- 2 S \leftarrow FSC;
- 3 R \leftarrow compute_resource_req(A, FSC);
- 4 **while** R < C **do**
- 5 config \leftarrow next_possible_UBS_config(S, B, N);
- 6 R \leftarrow compute_resource_req(A, config);
- 7 **if** R < C **then**
- 8 S \leftarrow config;
- 9 **end**
- 10 **end**
- 11 **return** S

On the other hand, a small accumulated error can be obtained if the columns with high scaling factors have low errors. As columns with high scaling factors correspond to MSB slices, our bit-slicing logic aims at reducing the errors in as many MSB slices as possible for achieving low accumulated error and high neural network accuracy. Non-zero G_{min} error at the output of a crossbar column with a certain slice size can be reduced by providing the slices with higher sensing margin. This can be achieved by using an RRAM device with n -bit capacity as an m -bit memory-cell (slice) such that $m < n$. This results in wider separation between the conductance states leading to higher sensing margin and reduced error in the crossbar column output due to non-zero G_{min} as shown in Fig. 9.

As discussed earlier, reducing the individual errors in as many MSB slices as possible will result in better accuracy. Different UBS configurations can be obtained based on how many MSBs are provided with high sensing margin to reduce the individual errors. For instance, using 2-bit RRAMs for 8-bit weights, [1, 1, 2, 2, 2] bits/slice (first 2 MSBs with high sensing margin, total five slices) and [1, 1, 1, 1, 2, 2] bits/slice (first 4 MSBs with high sensing margin, total six slices) are some of the possible configurations. UBS configuration with more number of slices results in better accuracy due to better matching between the positively and negatively scaled errors leading to

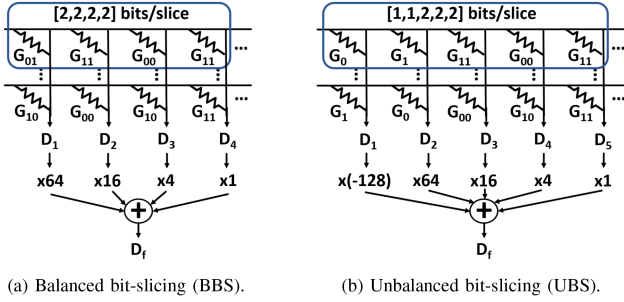


Fig. 10. Accumulation of partial digital outputs in CIM crossbar.

smaller accumulated error. However, it needs more energy due to more analog to digital conversion operations. For minimum energy requirement, an UBS configuration should have:

- Minimum number of slices per weight.
- High sensing margin for MSB slices.
- First MSB slice of 1-bit size for 2's complement arithmetic compatibility (details in Section IV-B2).

Such configuration is called *fundamental slice configuration* (FSC) which is obtained for N -bit weights and m -bit RRAMs as follows: 1 b for the first MSB slice and remaining $N-1$ bits divided into nearly equal chunks of m bits with small-sized chunks assigned to MSB slices. For example, with 8-bit weights and $m = 2$ bits per RRAM, we obtain FSC = [1, $m-1$, m , m , m] = [1, 1, 2, 2, 2] bits/slice which is used in Fig. 9.

UBS provides high accuracy at the cost of additional energy. Algorithm 1 gives UBS slice sizes for optimal accuracy in presence of non-zero G_{\min} error subjected to energy constraint. It starts with FSC having minimum energy requirement and then progressively assigns smaller slices to the next MSBs. Finally, the UBS slice configuration having the highest accuracy (highest number of slices) within the specified energy limit is selected.

2) *Crossbar Arithmetic*: Weights represented in 2's complement format cannot be mapped to a crossbar that uses BBS. This is due to the difficulty in isolating the negative contribution of the MSB from a multi-bit slice in 2's complement format [23]. Hence, BBS converts signed weights into equivalent positive weights using an offset and utilizes unsigned binary arithmetic to combine the column-wise partial outputs. In UBS, we force the MSB slice to always be of 1-bit size for compatibility with 2's complement weight encoding and utilize 2's complement arithmetic to combine the column-wise partial outputs.

Fig. 10 shows the accumulation of partial digital outputs (D_i 's) for 8-bit weights with 2-bit RRAMs using both conventional BBS and proposed UBS. We use [2, 2, 2, 2] bits/slice for BBS and FSC with [1, 1, 2, 2, 2] bits/slice for UBS. We obtain (2a) and (2b) as the expressions for the accumulated output error for BBS and UBS respectively, by using (1c) for dataflow graphs shown in Fig. 10.

$$E_{acc} = 64 \cdot E_1 + 16 \cdot E_2 + 4 \cdot E_3 + E_4 \quad (2a)$$

$$E_{acc} = (-128) \cdot E_1 + 64 \cdot E_2 + 16 \cdot E_3 + 4 \cdot E_4 + E_5 \quad (2b)$$

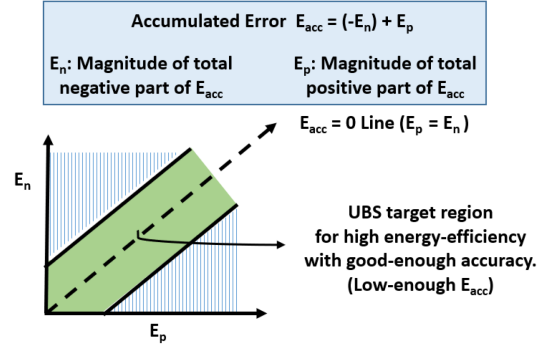


Fig. 11. Error reduction approach for high energy-efficiency.

As described by these equations, UBS can result in lower accumulated error compared to BBS due to weighted subtraction of column-wise errors in 2's complement arithmetic. This holds true for any UBS configuration, as they all use 1-bit slice for the first MSB to be compatible with 2's complement crossbar arithmetic. Moreover, the impact of non-zero G_{\min} error is further suppressed as the bit-slicing logic (described earlier in Section IV-B1) ensures that the accumulated error obtained after the weighted subtraction remains small.

C. Unbalanced Bit-Slicing Scheme for Energy-Efficiency

UBS leads to more slices per weight compared to BBS due to utilizing RRAMs for less than their bit-capacity for some slices to achieve higher sensing margin. The extra slices in UBS introduce additional analog-to-digital conversion operations resulting in higher energy consumption. Hence, it is necessary to decrease the number of slices per weight in UBS for reducing the energy overhead which can be achieved by using bigger slice sizes. However, the low sensing margin in bigger slice sizes leads to higher accumulated error and reduced accuracy. Thus, bigger slice sizes provide higher energy-efficiency at the expense of reduced accuracy and there exists a potential accuracy versus energy tradeoff.

Neural networks can inherently tolerate some deviation from ideal VMM computations. For instance, correct classification needs the output corresponding to the correct class to be maximum, but it does not matter if it is slightly higher or much higher than the other classes. So, instead of the targeting a very low accumulated error as done earlier in Section IV-B1 to obtain high accuracy, sufficiently good accuracy can still be achieved with a higher accumulated error which is just small enough to get the correct classification output. Thus, we can leverage bigger slices for higher energy-efficiency while incurring reasonable accuracy loss by adjusting the sizes of the bigger slices in such a way that the accumulated error remains small enough for correct classification. This is illustrated in Fig. 11. Hence, an energy-efficient UBS configuration that uses bigger slice sizes must satisfy the following conditions to achieve high energy-efficiency while maintaining sufficiently good accuracy:

- Less number of slices per weight than UBS FSC.

Algorithm 2: UBS Slice Configuration Selection for Energy-Efficiency.

input : CIM system architecture (A), Permissible accuracy loss w.r.t. FSC (P), Energy constraint (C), RRAM bit-capacity (B), Bits per full-precision neural weight (N)

output: UBS slice configuration for energy-efficiency (S)

```

1 FSC  $\leftarrow$  fundamental_slice_config( $B, N$ );
2 FSC_Acc  $\leftarrow$  compute_accuracy( $A, FSC$ );
3 Accmin  $\leftarrow$  FSC_Acc -  $P$ ;
4  $E \leftarrow$  list_energy_efficient_UBS_configs( $FSC, B, N$ );
5 Eligible_configs_list  $\leftarrow$   $\emptyset$ ;
6 Eligible_configs_list.insert( $FSC$ );
7 for config in  $E$  do
8   Accuracy  $\leftarrow$  compute_accuracy( $A, config$ );
9   Energy  $\leftarrow$  compute_energy( $A, config$ );
10  if Accuracy > Accmin & Energy <  $C$  then
11    | Eligible_configs_list.insert( $config$ )
12  end
13 end
14  $S \leftarrow$  minimum_energy_config( $Eligible_configs_list$ );
15 return  $S$ 

```

- Small enough accumulated error for correct classification.

To satisfy the constraint for number of slices, we need to utilize some slice sizes that are bigger than the RRAM bit-capacity. For instance, with 2-bit RRAMs for 8-bit weights, UBS FSC ([1, 1, 2, 2] bits/slice) leads to five slices. If we want the UBS configuration to fit in 4 or less slices, the only option is to keep the first MSB slice as 1-bit (for 2's complement compatibility) and increase the sizes of other slices so that the remaining seven bits fit in three or less slices. This leads to slice configurations such as [1, 2, 2, 3] bits/slice where some slices exceed the 2-bit capacity of our RRAM devices. Slices bigger than RRAM bit-capacity can be implemented by overloading the existing RRAM device. Overloading refers to the process of using n -bit RRAM as an m -bit memory-cell (slice) such that $m > n$, where $n = 2$ and $m = 3$ for our scenario. A non-overloaded RRAM (n -bit RRAM used to hold n or fewer bits) has a certain available sensing margin and no overlap exists between the variation profiles of its conductance states. Overloading an RRAM device leads to a larger number of conductance states within the same conductance range. This results in reduced sensing margin and overlap between the variation profiles of the conductance states. The more we overload a device (more difference between m and n), the lower is the sensing margin and the higher is the variation profile overlap. This can lead to increased errors due to non-zero G_{\min} in presence of conductance variation for a column with overloaded RRAMs.

Overloading RRAM devices to meet the first constraint for number of slices inherently increases the error in overloaded columns. However, we can still use overloading and satisfy the second constraint of small-enough accumulated error by overloading the columns which contribute less towards the accumulated error. The scaling factors for individual output errors (E_i s) in MSB columns are much higher compared to the LSB columns as evident from (2b). Hence, low accumulated error

can be achieved if MSB slices have smaller individual output errors compared to LSB slices. Hence, MSB slices in an energy-efficient UBS configuration should have higher sensing margin i.e. they should be smaller and less overloaded compared to LSB slices. Thus, a possible energy-efficient UBS configuration which can achieve low accumulated error in presence of RRAM overloading is obtained as follows:

- First MSB slice must be of 1-bit size.
- Slices per weight must be less than UBS FSC.
- LSB slices must be bigger or equal to MSB slices.

For 8-bit weights and 2-bit RRAMs (UBS FSC as [1, 1, 2, 2, 2] bits/slice) various possible energy-efficient UBS configurations which satisfy the aforementioned conditions can be listed as: [1, 2, 2, 3] bits/slice, [1, 1, 2, 4] bits/slice, [1, 1, 3, 3] bits/slice, [1, 1, 1, 5] bits/slice, [1, 3, 4] bits/slice, [1, 2, 5] bits/slice, [1, 1, 6] bits/slice and [1, 7] bits/slice. They provide different levels of accuracy while consuming different amounts of energy. Algorithm 2 describes the method of selecting appropriate energy-efficient UBS configuration according to the energy and accuracy constraints. It begins with FSC whose accuracy is taken as the baseline. The permissible accuracy loss with respect to this baseline is also specified as an input. Out of the various possible energy-efficient UBS configurations, the ones having accuracy and energy within the specified limits are shortlisted. Finally, slice configuration having the least energy requirement among the shortlisted ones is selected.

V. SIMULATION SETUP AND EXPERIMENTS

A. Simulation Setup

We have developed a simulation framework in Python based on in-situ multiply-accumulate (IMA) unit in [23] (shown in Fig. 3) which is compatible with UBS, BBS and HBS. The IMA design follows 32 nm CMOS technology [23]. The power and area details for various IMA components are also obtained from [23]. The power consumption of ADCs in the IMA is modelled based on the ADC design presented in [38] which is also referenced in [23]. In order to estimate the power/area for the aforementioned ADC at different bit resolutions while keeping rest of the ADC specifications same, we followed the methodology given in [23]. It involves scaling the power/area of all the ADC components except capacitive DAC (CDAC) linearly with ADC resolution and scaling the power/area of the CDAC exponentially with ADC resolution. The ADC power/area at new bit resolution is then obtained by summing all these scaled component-wise powers/areas. The RRAM device-related simulation parameters are taken from HfO_x-based device presented in [39]. All of our experiments take into account both non-zero G_{\min} error and conductance variation together. We consider 2-bit RRAMs (same as [23], [24]) and 8-bit weights. This leads to [1, 1, 2, 2, 2] bits/slice (FSC obtained using Algorithm 1) for UBS, [2, 2, 2, 2] bits/slice for BBS [23], [24] and [1, 1, 2, 2, 1, 1] bits/slice for HBS [22]. Other slice configurations are specified within the corresponding result figures.

We evaluate the performance of five datasets shown in Table I on a fully-connected neural network (FC-NN) as well as a

TABLE I
ACCURACY ON THE DATASETS CONSIDERED FOR SIMULATION

Dataset	Baseline Accuracy	
	Fully-connected Neural Network (FC-NN)	Convolutional Neural Network (CNN)
MNIST [33]	97.85%	98.75%
FMNIST [34]	88.57%	88.69%
KMNIST [35]	87.99%	92.47%
RMNIST [36]	87.94%	93.75%
EMNIST [37]	82.26%	85.40%

convolutional neural network (CNN). All of our datasets consist of 28×28 images. The details of the used neural networks are as follows:

- **FC-NN:** It consist of an input layer of 784 neurons followed by two hidden layers with 100 and 50 neurons. It has 47 output neurons for EMNIST [37] dataset and 10 output neurons for all other datasets. Thus, FC-NN can be expressed as 784-100-50-(47 or 10). The activation function used is ReLU.
- **CNN:** Its structure is similar to LeNet-5 [33] with two convolution layers having 6 kernels and 16 kernels of size 5×5 , each followed by a max-pooling layer. Flattened output from the second max-pooling layer connects to fully-connected layers with 120 neurons and 84 neurons. The output layer has 47 neurons for EMNIST [37] dataset and 10 neurons for all other datasets. So, we can represent the CNN as 6c5-maxpool-16c5-maxpool-flatten-120-84-(47 or 10), where mCn means m kernels of size $n \times n$. ReLU is used as activation function.

Both neural networks are trained using PyTorch [40] with Adam [41] optimizer and inference on CIM hardware is simulated using our Python-based framework. The Python-based simulation framework is validated by comparison with SPICE simulation. All the results are shown in terms of relative accuracy. It is calculated by expressing the accuracy obtained in Python-based hardware simulation for a given dataset as a percentage of its ideal (software) baseline accuracy in Table I. Thus, relative accuracy acts as a measure of how much software accuracy is preserved in CIM hardware in presence of non-zero G_{\min} error and conductance variation.

B. Experiments Performed

1) *Accuracy Comparison for Various Bit-Slicing Schemes:* The goal of this experiment is to demonstrate the effectiveness of UBS across various datasets and neural networks, when compared to state-of-the-art bit-slicing schemes and mitigation techniques. We compare the classification accuracy of UBS against BBS, HBS, BBS augmented with CST and HBS augmented with CST. This comparison is performed across five different datasets using FC-NN as well as CNN.

2) *Accuracy Versus Energy Tradeoff for UBS:* In this experiment, we compare the classification accuracy and energy consumption of UBS FSC with other energy-efficient UBS

configurations presented in Section IV-C to see if they can achieve reasonable accuracy while consuming less energy than UBS FSC. This is a key experiment, as a slicing scheme or configuration with high energy-efficiency but poor classification accuracy is not useful. It also sheds the light on how to select the best UBS configuration for energy-efficiency which satisfies a given accuracy constraint and vice versa. HBS and BBS are also included in this experiment to show how they fare against energy-efficient UBS configurations in terms of both accuracy and energy. This evaluation is performed across five datasets on FC-NN and CNN.

3) *Scalability of UBS:* This experiment aims to demonstrate the applicability of UBS to complex datasets and large neural networks. We use CIFAR-10 [42] dataset on VGG-16 [43] neural network for this purpose. As the original VGG-16 network is intended for 224×224 RGB images, we adapt it for 32×32 RGB images in CIFAR-10 by making a slight change in the final fully-connected layers. The resulting network can be represented as follows (nCm means n convolution filters of m x m size): Input - 64c3 - 64c3 - Maxpool - 128c3 - 128c3 - Maxpool - 256c3 - 256c3 - 256c3 - Maxpool - 512c3 - 512c3 - 512c3 - Maxpool - 512c3 - 512c3 - 512c3 - Maxpool - 512 - 10. This network achieves a software baseline accuracy of 89%. We then compare the classification accuracy of various bit-slicing schemes on this network for CIFAR-10 dataset.

VI. SIMULATION RESULTS

A. Accuracy Comparison for Various Bit-Slicing Schemes

UBS achieves the highest accuracy while HBS achieves the lowest accuracy among BBS, HBS and UBS for both FC-NN and CNN as shown in Figs. 12 and 13. The intuition behind this can be explained as follows. Non-zero G_{\min} error at the output of i^{th} crossbar column can be expressed as:

$$E_i = \frac{N_i \cdot \delta}{S} \quad (3)$$

where N_i is the number of RRAMs (in the i^{th} column) having G_{\min} conductance, δ is the error due to a single G_{\min} conductance and S is the sensing margin for such an architecture design. The column-wise errors E_i 's result in final accumulated error E_{acc} as per (2a) and (2b) for BBS and UBS respectively. UBS intends to reduce the error E_i compared to BBS by providing larger S (see 3) using smaller slice sizes for some important columns like MSBs. However, this also leads to higher N_i for some UBS columns compared to their BBS counterparts as smaller slice sizes produce more digital zero chunks which get mapped to G_{\min} , contributing towards increase in E_i . Nevertheless, weighted sum in (2a) leads to high accumulated error for BBS despite having smaller N_i 's. On the other hand, the impact of higher N_i 's on the accumulated error in UBS is severely diminished thanks to weighted subtraction due to 2's complement arithmetic as shown in (2b). The sensing margin allocation in UBS further reduces the impact of accumulated error that remains after the weighted subtraction. Hence, UBS has a very small accumulated error and thereby

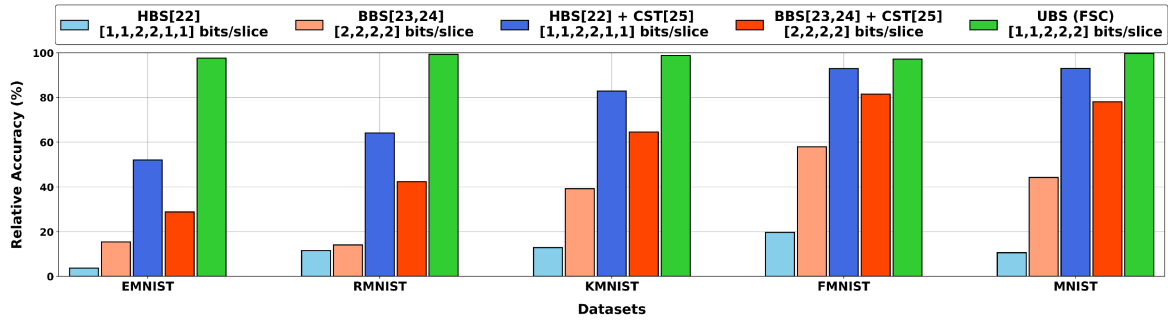


Fig. 12. Accuracy comparison for various bit-slicing schemes on fully-connected neural network across different datasets.

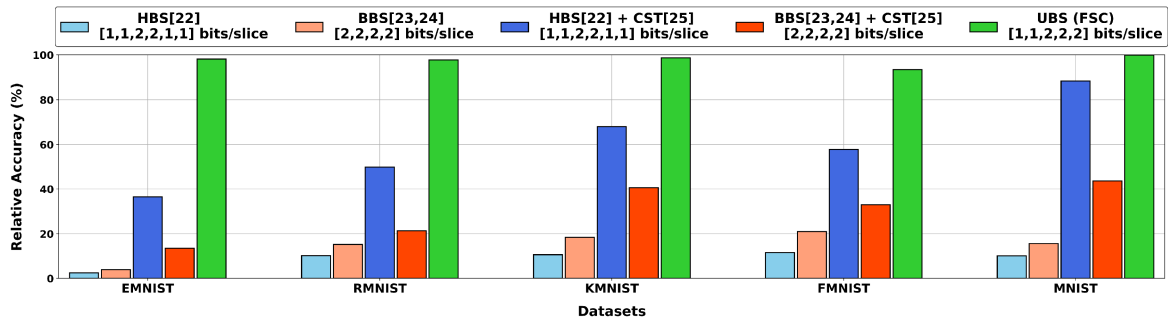


Fig. 13. Accuracy comparison for various bit-slicing schemes on convolutional neural network across different datasets.

provides much higher accuracy compared to BBS. Even though HBS provides high sensing margin for some slices, increase in N_i 's coupled with weighted accumulation of errors (similar to (2a)) due to its unsigned binary arithmetic leads to higher accumulated error and lower accuracy compared to both BBS and UBS.

CST [25] mitigation for BBS and HBS relies on current subtraction using a common dummy column for the entire crossbar. Such current subtraction is not perfect due to conductance variation and yields non-zero current residues. These current residues get accumulated across the columns leading to errors in the digital output after analog-to-digital conversion. These errors are further amplified during weighted sums to combine the column-wise partial digital outputs in HBS and BBS, leading to a high overall accumulated error. UBS does not rely on a common column and leverages weighted subtraction across various groups of columns. Hence, UBS outperforms HBS and BBS even when they are augmented with CST as shown in Figs. 12 and 13.

The final accumulated error at the output layer is high in CNN compared to FC-NN as our considered CNN has more layers than FC-NN. Thus, the impact of non-zero G_{\min} error becomes more severe in CNN. Hence, the accuracy for BBS and HBS (with as well as without CST) reduces on CNN when compared to that on FC-NN. On the other hand, higher number of layers in CNN result in negligible impact on UBS accuracy as it always inherently leads to low error for the output in each layer. This is evident in Fig. 12 and Fig. 13. Thus, UBS outperforms BBS and HBS (both with and without CST) across various datasets as well as types of neural networks.

B. Accuracy Versus Energy Tradeoff for UBS

We only consider in-situ multiply-accumulate (IMA) unit in [23] for energy comparison as bit-slicing schemes and configurations have no impact on its other components. Different bit-slicing schemes and configurations lead to changes in total number of crossbar columns as well as the number of bits per RRAM device (slice size) for a given column. These changes affect various IMA components such as analog-to-digital converters (ADCs), crossbar arrays, sample-and-hold circuits etc. However, the contribution of ADCs towards the overall IMA energy is significantly higher compared to other components [23]. Hence, we focus on the impact of bit-slicing schemes and configurations on the energy of ADCs which can be directly correlated to the IMA energy. A change in the number of crossbar columns leads to a change in total number of ADCs in the IMA to maintain constant throughput. A change in the number of bits per RRAM device for a column causes a change in the ADC resolution for correct analog-to-digital conversion. Thus, the overall IMA energy for a bit-slicing scheme or configuration is determined by the combined effect of the number of required ADCs and their resolutions.

Both BBS and HBS are augmented with CST for comparison in Figs. 14 and 15 as their accuracy is higher with CST (discussed in Section VI-A). UBS FSC ([1, 1, 2, 2, 2] bits/slice), BBS + CST ([2, 2, 2, 2] bits/slice) and HBS + CST ([1, 1, 2, 2, 1, 1] bits/slice) in Figs. 14 and 15 all have slices which are either 1-bit or 2-bit in size. Thus, ADC resolutions used in these schemes are quite similar. Hence, their IMA energy performance is mainly governed by total number of ADCs. The total number of ADCs required for UBS FSC is less than that in HBS + CST as it has

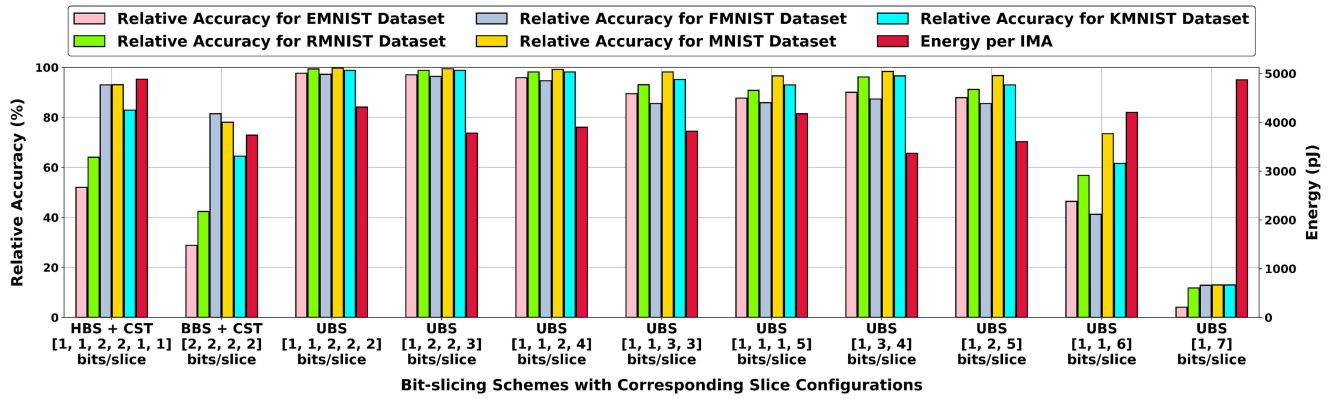


Fig. 14. Accuracy versus energy tradeoff demonstration for various bit-slicing schemes on fully-connected neural network across different datasets.

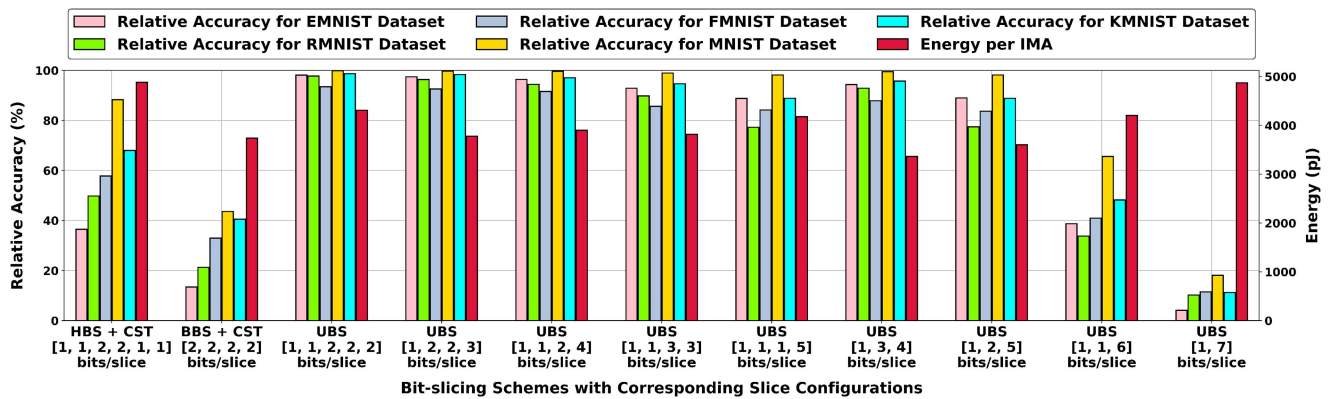


Fig. 15. Accuracy versus energy tradeoff demonstration for various bit-slicing schemes on convolutional neural network across different datasets.

less number of slices per weight. Hence, UBS FSC results in less energy than HBS + CST. On the other hand, UBS FSC has more slices per weight than BBS + CST which results in more number of ADCs and higher energy when compared to BBS + CST. This is shown in Figs. 14 and 15.

The energy overhead for UBS FSC can be lowered at the expense of reduced accuracy by decreasing the total number of slices. As discussed in Section IV-C, the possible energy-efficient UBS configurations can be listed as: [1, 2, 2, 3] bits/slice, [1, 1, 2, 4] bits/slice, [1, 1, 3, 3] bits/slice, [1, 1, 1, 5] bits/slice, [1, 3, 4] bits/slice, [1, 2, 5] bits/slice, [1, 1, 6] bits/slice and [1, 7] bits/slice. Two opposite effects regarding the energy consumption come into picture when we move from UBS FSC to energy-efficient UBS configurations:

- Energy-efficient UBS configurations have less number of slices than UBS FSC. Hence, they need less crossbar columns and less ADCs compared to UBS FSC. This contributes towards reduction in energy requirements.
- Energy-efficient UBS configurations use bigger slice sizes compared to UBS FSC. Hence, they need higher resolution ADCs which demand more energy. This contributes towards increase in energy requirements.

The overall energy consumption depends on which of these two effects dominates for a given slice configuration. It can be seen in Figs. 14 and 15 that [1, 3, 4] bits/slice consumes the

least energy among all schemes and configurations (even less than BBS). This indicates that the impact of number of slices (number of ADCs) is dominant over the impact of slice sizes (ADC resolution) for [1, 3, 4] bits/slice.

The energy-efficient UBS configuration in which bit-sizes of the adjacent slices are similar, leads to better balancing between the scaled column-wise errors. This results in smaller accumulated error and high accuracy. Hence, [1, 2, 2, 3] bits/slice gives the best accuracy among all the listed energy-efficient UBS configurations. This is because the maximum bit-size difference between its adjacent slice-sizes is 1, which is the smallest among all the listed energy-efficient UBS configurations. Similarly, among configurations with 3 slices, [1, 3, 4] bits/slice gives the highest accuracy as the maximum difference between its adjacent slices is 2, while it is 3 and 5 for [1, 2, 5] bits/slice and [1, 1, 6] bits/slice respectively. Note that even though energy-efficient UBS configurations achieve slightly less accuracy compared to UBS FSC having [1, 1, 2, 2, 2] bits/slice, they still achieve much higher accuracy compared to BBS and HBS (both with CST) as evident in Figs. 14 and 15. The configurations [1, 1, 6] bits/slice and [1, 7] bits/slice result in significantly poor accuracy compared to all other energy-efficient UBS configurations as depicted in Figs. 14 and 15. Hence, it is not recommended to overload a 2-bit RRAM device for holding more than 5 bits. The analysis in Figs. 14 and 15 can be used to find the UBS

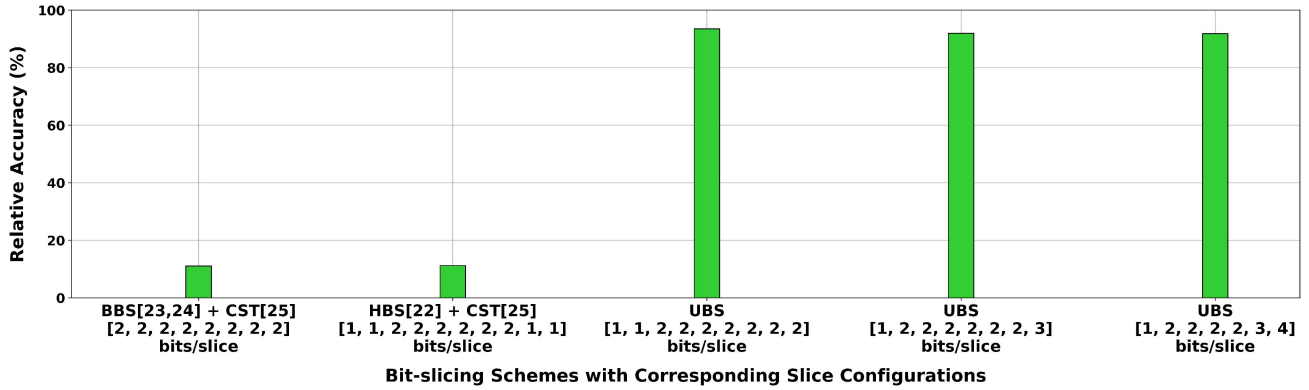


Fig. 16. Accuracy comparison for various bit-slicing schemes on VGG-16 neural network for CIFAR-10 dataset.

TABLE II

SUMMARY OF THE PERFORMANCE METRICS PER IMA, UNIT INDICATED IN BRACKETS WITH EACH METRIC. ACCURACY VALUES FOR EMNIST DATASET ON CNN ARE REPORTED. CORRECT OPERATIONS PER UNIT ENERGY = (TOTAL OPERATIONS \times ACCURACY/100)/ENERGY, UNIT: GIGA OPERATIONS PER JOULE (GOP/J)

Performance Metric	UBS [This work]			BBS [23, 24] + CST [25]	HBS [22] + CST [25]
	[1,1,2,2,2] bits/slice	[1,2,2,3] bits/slice	[1,3,4] bits/slice	[2,2,2,2] bits/slice	[1,1,2,2,1,1] bits/slice
Energy (pJ)	4311	3778	3365	3738	4883
Net energy-efficiency (GOPS/W)	237.5	271.0	304.3	273.9	209.7
Accuracy (%)	83.77	83.23	80.54	11.46	31.19
Correct operations per unit energy (GOP/J)	198.9	225.5	245.1	31.4	65.4
Area (μm^2)	23324	23121	26985	21765	24883

configuration having the lowest energy consumption for a given accuracy constraint as well as the UBS configuration with the highest accuracy for a given energy budget.

C. Scalability of UBS

The accuracy comparison of various bit-slicing schemes on VGG-16 network using CIFAR-10 dataset is shown in Fig. 16. We have considered HBS and BBS empowered with CST for this comparison because they achieve their highest possible accuracy with CST as evident in Figs. 12 and 13. We use 16-bit weights as CIFAR-10 is a complex dataset and VGG-16 is a large network [23]. Considering 2 bits per RRAM as in Section V-A, we get [1, 1, 2, 2, 2, 2, 2, 2, 2] bits/slice for UBS, [2, 2, 2, 2, 2, 2, 2, 2] bits/slice for BBS+CST and [1, 1, 2, 2, 2, 2, 2, 2, 1, 1] bits/slice for HBS+CST. To improve the energy efficiency of UBS with 9 slices ([1, 1, 2, 2, 2, 2, 2, 2, 2] bits/slice), we follow the same logic as in Section VI-B to reduce the number of slices. This leads to [1, 2, 2, 2, 2, 2, 2, 3] bits/slice as the best (most accurate) configuration having 8 slices and [1, 2, 2, 2, 2, 3, 4] bits/slice as the best (most accurate) configuration having 7 slices. These slice configurations are also included for the comparison in Fig. 16. It is evident from Fig. 16 that UBS outperforms HBS+CST and BBS+CST on VGG-16 with CIFAR-10 dataset in terms of accuracy. This can be attributed to the mitigation provided by the sensing margin allocation which is also supported by 2's complement crossbar arithmetic. Moreover, energy-efficient UBS configurations ([1, 2, 2, 2, 2, 2, 3] bits/slice and [1, 2, 2, 2, 2, 3, 4] bits/slice) incur a very minor accuracy loss compared to the UBS with 9 slices ([1, 1, 2, 2, 2, 2, 2, 2] bits/slice) as they sacrifice sensing margin for LSBs to reduce the number of slices.

D. Summary of Performance Metrics and Discussion

The performance metrics per IMA for various bit-slicing schemes are summarized in Table II. We consider both BBS and HBS augmented with CST so that their most accurate versions are considered for comparison. Out of the various energy-efficient UBS configurations discussed in Section VI-B, [1, 2, 2, 3] bits/slice and [1, 3, 4] bits/slice are selected as they provide the highest accuracy and the lowest energy among UBS configurations having 4 slices per weight and 3 slices per weight respectively. UBS FSC ([1, 1, 2, 2, 2] bits/slice) provides $7.3\times$ accuracy at the expense of 15.3% energy overhead compared to BBS, while it achieves $2.7\times$ accuracy and consumes 11.7% less energy when compared to HBS. The UBS energy overhead with respect to BBS can be reduced from 15.3% to mere 1.1% by using UBS with [1, 2, 2, 3] bits/slice. We can even achieve 10% energy savings compared to BBS by leveraging UBS with [1, 3, 4] bits/slice.

The net energy-efficiency represents the number of operations performed per unit energy consumption and is expressed as giga operations per second per watt (GOPS/W) in Table II. The energy-efficiency comparison can be correlated to the energy comparison discussed earlier by taking into account the fact that these two have inverse relationship. For instance, the more energy-efficient scheme is the one that consumes less energy or has higher GOPS/W.

Metrics like energy and energy-efficiency do not take into account the fact that even though BBS appears to be energy-efficient, it is spending that energy in performing incorrect computations as reflected in its poor accuracy. Hence, we define a metric called “correct operations per unit energy” which takes into account both energy and computational correctness. The number of correct operations can be computed as a product of

accuracy (in fraction format, not as a percentage) and total operations. Correct operations per unit energy is then simply obtained by dividing the number of correct operations by total energy consumption. UBS FSC ([1, 1, 2, 2, 2] bits/slice) provides $6.3\times$ and $3\times$ correct operations per unit energy compared to BBS and HBS respectively. [1, 2, 2, 3] bits/slice provides $7.2\times$ and $3.4\times$ correct operations per unit energy compared to BBS and HBS respectively, at the cost of just 0.54% accuracy loss compared to UBS FSC. [1,3, 4] bits/slice results in $7.8\times$ and $3.7\times$ correct operations per unit energy compared to BBS and HBS respectively, at the expense of 3.23% accuracy loss compared to UBS FSC. This shows that the number of correct computations performed per unit energy consumption is much higher for UBS and it increases further with the use of energy-efficient UBS configurations for a very small decrease in accuracy.

UBS FSC ([1, 1, 2, 2, 2] bits/slice) requires 7.2% more area with respect to BBS. UBS with [1, 2, 2, 3] bits/slice and [1, 3, 4] bits/slice incur area overheads of 6.2% and 24% respectively when compared to BBS. Thus, UBS with [1, 2, 2, 3] bits/slice is the best choice if the target is to achieve high energy-efficiency with accuracy closest to UBS FSC and minimum area overhead with respect to BBS. On the other hand, [1, 3, 4] bits/slice is the best choice if we are willing to trade further accuracy and area for even higher energy-efficiency.

Please note that the bit-sizes for weights in different layers of the neural network can be selected independent of UBS. The bit-sizes for weights in each layer are given as a design constraint to UBS (like number of bits per RRAM) and UBS tries to deliver the best possible performance for these given bit-sizes. For designs which use the same bit-size for weights in every layer, all layers should be provided with the same slice configuration based on the accuracy versus energy-efficiency tradeoff. For instance, with 16-bit weights and 2-bit RRAMs, [1, 1, 2, 2, 2, 2, 2, 2] bits/slice should be used for all layers if accuracy is the primary concern and [1, 2, 2, 2, 2, 3, 4] bits/slice should be used for all layers if energy-efficiency is the major concern. If the design uses different bit-sizes for weights in different layers, the same strategy can be extended to layers that use the same bit-size. For instance, consider a network where half the layers use 8-bit weights, and half the layers use 16-bit weights. If accuracy is the primary target, then one should use [1, 1, 2, 2, 2] bits/slice for 8-bit layers and [1, 1, 2, 2, 2, 2, 2, 2] bits/slice for 16-bit layers. On the other hand, for energy-efficiency, one should use [1, 3, 4] bits/slice for 8-bit layers and [1, 2, 2, 2, 2, 3, 4] bits/slice for 16-bit layers. Moreover, UBS is designed to make the vector-matrix multiplication error free. Provided that the core computation involved in a layer is vector-matrix multiplication or matrix-matrix multiplication, any such layer can reap the benefits of UBS by following the slice configuration allocation logic discussed earlier. Hence, UBS is applicable to any type of deep learning layer as almost all types of layers in deep-learning involve matrix-matrix multiplication as the core computation.

VII. CONCLUSION

We propose an unbalanced bit-slicing scheme to mitigate the impact of non-zero minimum conductance of RRAM devices

on CIM architectures. The proposed scheme achieves this by appropriate utilization of sensing margin and leveraging the weighted subtraction effect of 2's complement crossbar arithmetic. Two different sensing margin allocation strategies are proposed based on whether the final goal is to achieve high accuracy or energy-efficiency. The proposed scheme provides up to $7.3\times$ accuracy and up to $7.8\times$ correct operations per unit energy consumption compared to state-of-the-art. Such high accuracy and energy-efficiency can benefit a wide spectrum of AI applications like autonomous vehicles, wearable healthcare etc. This work has shown that by introducing smart techniques such as the proposed unbalanced bit-slicing scheme in RRAM-based CIM architectures, one can deal with device-level non-idealities like non-zero minimum RRAM conductance.

REFERENCES

- [1] Google, Facebook and microsoft are remaking themselves around AI. [Online]. Available: <https://www.wired.com/2016/11/google-facebook-microsoft-remaking-around-ai/>
- [2] From 2016: Why deep learning is suddenly changing your life. [Online]. Available: <https://fortune.com/longform/ai-artificial-intelligence-deep-machine-learning/>
- [3] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, 2017.
- [4] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1–9.
- [5] Intel processor family. [Online]. Available: <https://www.intel.in/content/www/in/en/products/processors/core.html>
- [6] NVIDIA turing architecture GPUs. [Online]. Available: <https://www.nvidia.com/en-in/geforce/turing/>
- [7] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Architecture (ISCA)*, 2017, pp. 1–12.
- [8] S. Hamdioui et al., "Memristor for computing: Myth or reality?" in *Proc. Design, Automat. Test Europe Conf. Exhib.*, 2017, pp. 722–731.
- [9] A. Sebastian et al., "Memory devices and applications for in-memory computing," *Nature Nanotechnol.*, vol. 15, pp. 529–544, 2020.
- [10] F. Cai et al., "A fully integrated reprogrammable memristor-cmos system for efficient multiply-accumulate operations," *Nature Electron.*, vol. 2, no. 7, pp. 290–299, 2019.
- [11] L. Ni et al., "Distributed in-memory computing on binary rram crossbar," *ACM J. Emerg. Technol. Comput. Syst. (JETC)*, vol. 13, no. 3, pp. 1–18, 2017.
- [12] S. Borkar et al., "Design and reliability challenges in nanometer technologies," in *Proc. 41st Annu. Des. Automat. Conf.*, 2004, pp. 75–75.
- [13] N. Z. Haron and S. Hamdioui, "Why is CMOS scaling coming to an END?," in *Proc. 3rd Int. Des. Test Workshop*, 2008, pp. 98–103.
- [14] *International Roadmap for Devices and Systems*. [Online]. Available: standards.ieee.org/develop/indconn/irds/index.html
- [15] S. Hamdioui et al., "Memristor based computation-in-memory architecture for data-intensive applications," in *Proc. Des., Automation & Test in Europe Conference Exhibition (DATE)*, 2015, pp. 1718–1725.
- [16] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature Mater.*, vol. 18, pp. 309–323, 2019.
- [17] L. Ni et al., "An energy-efficient digital rram-crossbar-based cnn with bitwise parallelism," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 3, pp. 37–46, Dec. 2017.
- [18] A. P. James, "A hybrid memristor-CMOS chip for AI," *Nature Electron.*, vol. 2, no. 7, pp. 268–269, 2019.
- [19] P.-Y. Chen and S. Yu, "Technological benchmark of analog synaptic devices for neuro-inspired architectures," *IEEE Des. Test*, vol. 36, no. 3, pp. 31–38, Jun. 2019.
- [20] P.-Y. Chen et al., "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018.
- [21] A. Siemon et al., "Memristive device modeling and circuit design exploration for computation-in-memory," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2019, pp. 1–5.

- [22] A. Ankit et al., "PANTHER: A. programmable architecture for neural network training harnessing energy-efficient ReRAM," *IEEE Trans. Comput.*, vol. 69, no. 8, pp. 1128–1142, Aug. 2020.
- [23] A. Shafiee et al., "ISAAC: A. convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Architecture (ISCA)*, 2016, pp. 14–26.
- [24] A. Ankit et al., "PUMA: A. programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proc. Twenty-Fourth Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2019, pp. 715–731.
- [25] P.-Y. Chen et al., "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2015, pp. 194–199.
- [26] S. Diware et al., "Unbalanced bit-slicing scheme for accurate memristor-based neural network architecture," in *Proc. IEEE 3rd Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, 2021, pp. 1–4.
- [27] S. Shukla et al., "A scalable multi-teraops core for ai training and inference," *IEEE Solid-State Circuits Lett.*, vol. 1, no. 12, pp. 217–220, Dec. 2018.
- [28] A. Hardtdegen et al., "Improved switching stability and the effect of an internal series resistor in hfo2/tiox bilayer reram cells," *IEEE Trans. Electron Devices*, vol. 65, no. 8, pp. 3229–3236, Aug. 2018.
- [29] W. Kim et al., "Multistate memristive tantalum oxide devices for ternary arithmetic," *Sci. Rep.*, 2016, vol. 6, no. 1, pp. 1–9, 2016.
- [30] G. Pedretti et al., "Conductance variations and their impact on the precision of in-memory computing with resistive switching memory (RRAM)," in *Proc. IEEE Int. Rel. Phys. Symp.*, 2021, pp. 1–8.
- [31] G. Charan et al., "Accurate inference with inaccurate RRAM devices: A joint algorithm-design solution," *IEEE J. Explor. Solid-State Computat.*, vol. 6, no. 1, pp. 27–35, Jun. 2020.
- [32] L. Chen et al., "accelerator-friendly neural-network training: Learning variations and defects in rram crossbar," in *Proc. Design, Automat. Test Europe Conf. Exhib. (DATE)*, 2017, pp. 19–24.
- [33] Y. Lecun et al., "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [34] H. Xiao et al., "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv:1708.07747*.
- [35] T. Clanuwat et al., "Deep learning for classical japanese literature," *arXiv:1812.01718*.
- [36] MNIST Variations. [Online]. Available: https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits
- [37] G. Cohen et al., "EMNIST: Extending MNIST to handwritten letters," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 2921–2926.
- [38] L. Kull et al., "A 3.1 mW 8b 1.2 GS/s single-channel asynchronous SAR ADC with alternate comparators for enhanced speed in 32 nm digital SOI CMOS," *IEEE J. Solid-State Circuits*, vol. 48, no. 12, pp. 3049–3058, Dec. 2013.
- [39] A. Prakash and H. Hwang, "Multilevel cell storage and resistance variability in resistive random access memory," *Phys. Sci. Rev.*, vol. 1, no. 6, 2016, Art. no. 20160010.
- [40] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8026–8037.
- [41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations, ICLR*, 2015.
- [42] The CIFAR-10 dataset. [Online]. Available: <https://www.cs.toronto.edu/kriz/cifar.html>
- [43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Representations, ICLR*, 2015.



Sumit Diware received the master's degree in VLSI design tools and technology from the Indian Institute of Technology Delhi, New Delhi, India, in 2018. He is currently working toward the Ph.D. degree with Computer Engineering Laboratory, Delft University of Technology, Delft, The Netherlands. His research interests include computation-in-memory and neuromorphic architectures for AI.



Abhairaj Singh received the master's degree (*Cum Laude*) in electrical engineering from the Delft University of Technology, Delft, The Netherlands, in 2019. He is working toward the Ph.D. degree with the Computer Engineering Laboratory in Delft. He was a SRAM/ROM/RF Circuit Design Engineer with ARM Technologies for more than four years. He was on memory-based circuit techniques, verification methodologies, design automation, assist circuits, CMOS sensors, low-power (IoT) techniques and silicon debug. He holds one patent during his time at ARM. His research interests include circuit design for hardware AI, computation-in-memory, and emerging technologies.



Anteneh Gebregiorgis (Member, IEEE) received the Ph.D. degree in computer science from the Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2019. He is currently a Postdoctoral Researcher with the Computer Engineering Laboratory (CE-Lab) of Delft University of Technology, Delft, The Netherlands. His research interests include emerging technologies, artificial intelligence (AI), computation in-memory (CIM), neuromorphic computing, architectures for ultra-low power design, reliability analysis, and variability assessment of VLSI devices.



Rajiv V. Joshi (Fellow, IEEE) received the B.Tech. degree from the Indian Institutes of Technology Bombay, Mumbai, India, the M.S. degree from the Massachusetts Institute of Technology, Cambridge, MA, USA, and the Dr. Eng. Sc. degree from Columbia University, New York, NY, USA. He is currently a Research Staff Member and Key Technical Lead with T. J. Watson research center, IBM. He has led successfully predictive failure analytic techniques for yield prediction and also the technology-driven SRAM at IBM Server Group. His statistical techniques are tailored for machine learning and AI. He developed novel memory designs which are universally accepted. He commercialized these techniques. He has authored or coauthored more than 200 papers. He has given more than 45 invited/keynote talks and given several Seminars. He holds 60 invention plateaus and has more than 250 U.S. patents and more than 400 including international patents. He was the recipient of the three Outstanding Technical Achievements (OTAs), three Highest Corporate Patent Portfolio awards for licensing contributions, NY IP Law association "Inventor of the Year" Award in February 2020. He is awarded prestigious IEEE Daniel Noble Award for 2018, Best Editor Award from IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION Journal, and 2015 BMM Award. He is inducted into New Jersey Inventor Hall of Fame in August 2014. He is a Member of the IBM Academy of technology and a master inventor. He was a Distinguished Lecturer of the IEEE CAS and EDS society. He is currently a Distinguished Lecturer of the CEDA. He is ISQED and World Technology Network Fellow and distinguished alumnus of IIT Bombay.



Said Hamdioui (Senior Member, IEEE) received the M.S.E.E. and Ph.D. degrees (both with Hons.) from the Delft University of Technology, Delft, The Netherlands. He is currently a Chair Professor of Dependable and Emerging Computer Technologies, Head of the Quantum and Computer Engineering Department, and is Head of the Computer Engineering Laboratory (CE-Lab) of the Delft University of Technology, The Netherlands. He is also Co-Founder and CEO of Cognitive-IC, a start-up focusing on hardware dependability solutions.. He spent about seven years

within industry, including Microprocessor Products Group with Intel Corporation (California, USA), IP and Yield Group with Philips Semiconductors R&D (Crolles, France) and DSP design group with Philips/ NXP Semiconductors, Nijmegen, The Netherlands. His research interests include two domains: dependable CMOS nano-computing, including Testability, Reliability, Hardware Security and emerging technologies and computing paradigms, including memristors for logic and storage, in-memory-computing for big-data applications. He is currently involved in different national and EU projects. Hamdioui owns two patents, has authored or coauthored one book and contributed to other two, and coauthored more than 200 conference and journal papers. He has consulted for many companies, such as Intel, ST, Altera, Atmel, Renesas, which include memory testing and has collaborated with many industry/research partners in the field of dependable nano-computing and emerging technologies. He is strongly involved in the international community as a Member of organizing committees or a Member of the technical program committees of the leading conferences. He delivered dozens of keynote speeches, distinguished lectures, and invited presentations and tutorial at major international forums/conferences/schools and at leading semiconductor companies. Hamdioui is a Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, and he is on the Editorial Board of the *IEEE Design & Test*, *Elsevier Microelectronic Reliability Journal*, and of the *Journal of Electronic Testing: Theory and Applications*. He is also Member of AENEAS (Association for European Nano Electronics Activities)/ENIAC Scientific Committee Council.



Rajendra Bishnoi received the Ph.D. degree in computer science from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2017. He is currently an Assistant Professor with the Computer Engineering Laboratory, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology (TU-Delft), Delft, The Netherlands. Before joining TU-Delft, he was a Research Leader for the MRAM Group in the Chair of Dependable Nano Computing, KIT for more than two years. From 2006 to 2012, he was a Design Engineer

at Freescale (NXP), where he was a part of the Technical Solution Group in memory and SoC flow. He was the recipient of the EDAA Outstanding Dissertation Award for the year 2017. His research interests include hardware AI and computation-in-memory and emerging technologies.