Unbalanced Bit-slicing Scheme for Accurate Memristor-based Neural Network Architecture

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

DOI
10.1109/AICAS51828.2021.9458443

Publication date
2021

Document Version
Accepted author manuscript

Published in
2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)

Unbalanced Bit-slicing Scheme for Accurate Memristor-based Neural Network Architecture

Sumit Diware* Anteneh Gebregiorgis* Rajiv V. Joshi† Said Hamdioui* Rajendra Bishnoi*
*Computer Engineering Lab, Delft University of Technology, Delft, The Netherlands.
Email: {S.S.Diware, A.B.Gebregiorgis, S.Hamdioui, R.K.Bishnoi}@tudelft.nl
†IBM Research Division, Yorktown Heights, NY, USA. Email: rvjoshi@us.ibm.com

***Abstract*—Emerging memristor-based computing has the potential to achieve higher computational efficiency over conventional architectures. Bit-slicing scheme, which represents a single neural weight using multiple memristive devices, is usually introduced in memristor-based neural networks to meet high bit-precision demands. However, the accuracy of such networks can be significantly degraded due to non-zero minimum conductance (G_{min}) of memristive devices. This paper proposes an unbalanced bit-slicing scheme; it uses smaller slice sizes for more important bits to provide higher sensing margin and reduces the impact of non-zero G_{min}. Moreover, the unbalanced bit-slicing is assisted by 2's complement arithmetic which further improves the accuracy. Simulation results show that our proposed scheme can achieve up to 8.8× and 1.8× accuracy compared to state-of-the-art for single-bit and two-bit configurations respectively, at reasonable energy overheads.**

I. Introduction

Neural network-based cognitive applications like object recognition [1], natural language processing [2] have become an integral part of modern computing systems. Existing computing systems based on von-Neumann architecture e.g. CPUs, GPUs and TPUs lead to degradation of performance and energy efficiency for neural network applications, mainly due to the memory wall [3]. *Computation In-Memory* (CIM) using emerging non-volatile memory technologies such as resistive random access memories (RRAMs) also known as memristors can perform analog computing within the memory itself by replacing digital operations with circuit laws [4], [5]. Hence, CIM can potentially serve as a more efficient alternative to von-Neumann architecture [6]. The bit-capacity of RRAM devices is typically less than that demanded by neural network applications. Therefore, a bit-slicing scheme [7], [8] is commonly employed in CIM architectures; multiple RRAM devices represent a single neural weight and shift-and-add post-processing is performed to combine the partial outputs. However, these architectures suffer from non-zero minimum conductance (G_{min}) error, in which a zero weight in the neural network is represented by a RRAM device with non-zero G_{min} conductance [9]. Such RRAM devices with G_{min} conductance produce non-zero operational currents that severely impact the sensing margin and make the output erroneous.

Limited work has been published on mitigating the impact of non-zero G_{min} error for CIM architectures with bit-slicing. RRAM-based CIM accelerators ISAAC [7] and PUMA [8] both use the same *balanced* bit-slicing (BBS) scheme for *Vector-Matrix Multiplication* (VMM). However, BBS scheme cannot provide good accuracy in presence of non-zero G_{min} error due to limited sensing margin and high accumulative non-zero G_{min} error. PANTHER [10], which is an extension of PUMA [8], proposes *heterogeneous* bit-slicing (HBS) scheme, but it is meant for optimizing weight update operations and not for VMM. Moreover, this scheme is susceptible to accuracy degradation due to non-zero G_{min} error as it uses the same weight encoding and underlying crossbar arithmetic as that of PUMA [8]. *Current subtraction technique* [11] can reduce G_{min} impact on these accelerators but cannot provide good accuracy when conductance variation is considered. It also introduces additional analog components like opamps, resistors which contribute more noise and variations making the computations even more error-prone. Hence, there is a strong need for an effective solution to non-zero G_{min} error.

This paper proposes an *unbalanced* bit-slicing (UBS) scheme for RRAM-based CIM to mitigate the impact of non-zero G_{min} error and conductance variation on VMM. UBS provides higher sensing margin for more important bits; i.e. most significant bits (MSBs) to reduce the impact of non-zero G_{min} error and conductance variation. This suffices for good accuracy due to the robustness of neural networks to minor computational fluctuations. In addition, UBS is supported with 2's complement arithmetic whose inherent differential nature further suppresses the impact of non-zero G_{min} error and conductance variation. The key contributions of this paper are:

- An unbalanced bit-slicing scheme which provisions extra sensing margin for more important bits.
- A method to find unbalanced bit-slice sizes for optimal accuracy in presence of non-zero G_{min} error for given resource constraints and system specifications.
- A holistic solution consisting of unbalanced bit-slicing and 2's complement arithmetic which mitigates non-zero G_{min} error as well as conductance variation impact.

The proposed UBS scheme is evaluated with MNIST and Fashion-MNIST datasets. It achieves up to 8.8× accuracy compared to state-of-the-art with negligible energy overhead for 1 bit per slice. For 2 bits per slice, it provides up to 1.8× accuracy compared to state-of-the-art at an energy overhead of no more than 14.4%.

The rest of this paper is organized as follows. Section II presents the fundamentals of CIM and motivation for the prob-

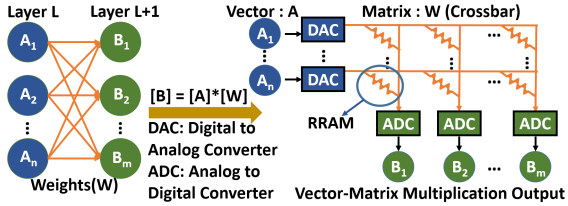


Fig. 1. CIM-based vector-matrix multiplication using RRAM devices.

lem statement. Section III provides details of the proposed bit-slicing scheme, followed by experimental results in Section IV. Finally, Section V concludes the paper.

II. Background

Data storage in the form of RRAM conductance allows leveraging circuit laws (Ohm's law and Kirchhoff's current law) to perform computing within the memory itself. Fig. 1 shows mapping of VMM operation in a neural network onto a CIM architecture. Elimination of data movement and highly parallel analog computation enable CIM to achieve high computational and energy efficiency.

However, RRAM devices in CIM architecture typically have less bit-precision than that demanded by neural networks. To overcome this problem, bit-slicing is utilized in CIM architectures as shown in Fig. 2 [7], [8]. Neural weights and inputs are divided into smaller chunks called slices, which are mapped to conductances and voltages respectively. Column currents resulting from time-multiplexed voltage inputs are converted to digital, shifted and added to get the final output.

As a zero weight in RRAM-based neural network is represented by a non-zero G_{min} conductance, a non-zero output current is produced. This is called *non-zero G_{min} error* which leads to accuracy degradation of up to 88.7% for 1 bit/slice and 45.4% for 2 bits/slice with respect to fixed-point baseline in bit-slicing architectures [7], [8], as will also be shown in Section IV. Avoiding the use of bit-slicing is not possible as single memristor cannot hold 8 or 16 bit neural weight [7]. Hence, non-zero G_{min} error remains a pressing concern.

III. Proposed Unbalanced Bit-slicing Scheme

A. Overview

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. Fig. 3 shows the overview of conventional [7], [8] and the proposed bit-slicing schemes. For conventional scheme, *balanced* bit-slicing

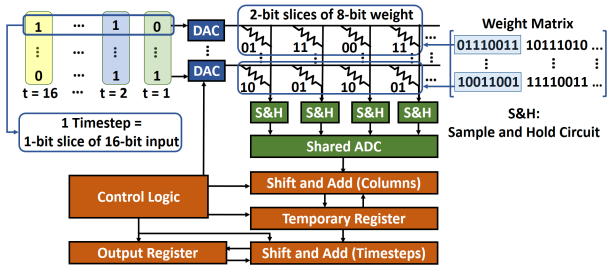


Fig. 2. RRAM-based CIM architecture with bit-slicing.

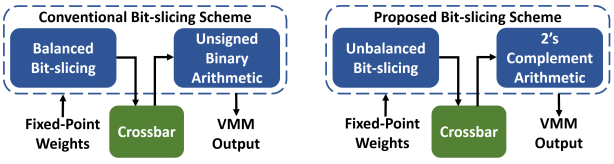


Fig. 3. Overview of conventional and proposed bit-slicing schemes.

(BBS) logic provides low sensing margin resulting in high susceptibility to non-zero G_{min} errors while unsigned binary arithmetic leads to high accumulative error after combining the partial outputs. In the proposed scheme, *unbalanced* bit-slicing (UBS) logic provides higher sensing margin to more important bits, while 2's complement arithmetic leads to reduction in accumulative error after combining the partial outputs; both these effects minimize the impact of non-zero G_{min} error and improve the neural network accuracy.

B. Unbalanced Bit-slicing (UBS) Logic

Using a memristor with n-bit maximum capacity as an m-bit memory-cell (slice) such that $m<n$ results in higher sensing margin and makes the crossbar column output immune to non-zero G_{min} error as shown in Fig. 4. As neural networks can inherently tolerate small deviation from ideal computation, error-free MSBs suffice for high accuracy. Hence, UBS uses less bits ($<n$) for slices corresponding to MSBs for providing high sensing margin and reducing the impact of non-zero G_{min} error compared to conventional bit-slicing.

Different UBS configurations can be obtained based on how many MSBs are provided with high sensing margin. For instance, using 2 bits/RRAM for 8 bit weight, [1,1,2,2,2] bits/slice (5 slices) and [1,1,1,1,2,2] bits/slice (6 slices) are some of the possible configurations. A configuration with more number of slices leads to better accuracy due to more error-free MSBs. However, it needs more area and energy due to more crossbar columns and analog to digital conversion operations. For minimum possible area and energy requirement, an UBS configuration should have: 1) Minimum number of slices with highest sensing margin for MSBs. 2) First MSB slice of 1 bit for 2's complement arithmetic (details in Section III-C). Such configuration is called *fundamental slice configuration* (FSC) which is obtained for m-bits/RRAM as follows: 1 bit for MSB slice, m-1 bits for next slice, and m bits for the remaining

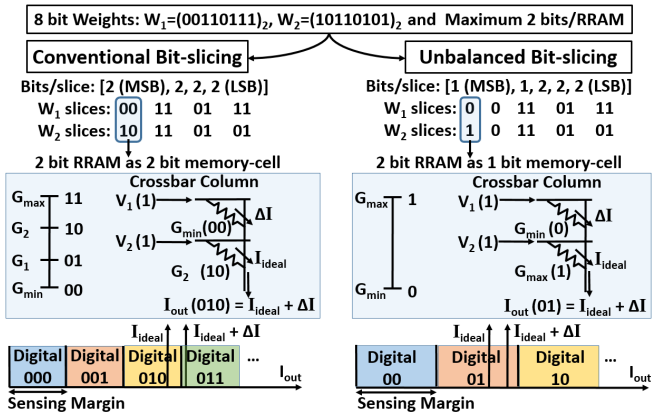


Fig. 4. Impact of sensing margin on bit-slicing schemes.

Algorithm 1: Optimal slice size computation

input : System architecture A, RRAM maximum bit-capacity R, area and energy budget B_{max}
output: Slice configuration S

```
F ← fundamental_slice_config(A, R);
M ← compute_resource_req(A, F);
S ← F;
if M < B_max then
    while M < B_max do
        J ← list_possible_next_MSB_configs(A, R);
        T ← list_configs_within_budget(A, J, B_max);
        S ← maximum_accuracy_config(T);
        M ← compute_resource_req(A, S);
    end
end
return S
```

slices. E.g., with 8-bit weights and *m*=2 bits/RRAM, FSC = [1,*m*-1,*m*,*m*,*m*]=[1,1,2,2,2] bits/slice; this is used in Fig. 4.

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

C. Crossbar Arithmetic

Conventional BBS cannot work with 2's complement arithmetic due to the difficulty in isolating the contribution of MSB from a multi-bit slice in 2's complement format [7]; it converts signed weights to positive weights using an offset and utilizes unsigned binary arithmetic to combine partial outputs. However, UBS can use 2's complement arithmetic as it can isolate MSB contribution by forcing MSB slice to be 1 bit only. Fig. 5 shows the accumulation of partial digital outputs (D_i's) for 8 bit weights with maximum 2 bits/RRAM using both conventional BBS and proposed UBS. Conductance subscripts indicate binary slice value. Let D_i=$T_i + E_i$, where T_i is the ideal value and E_i the error due to non-zero G_{min}. Similarly, D_f=$T_f + E_f$ for final output. Eq. 1a below gives E_f for conventional BBS, while Eq. 1b gives it for the proposed UBS.

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

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

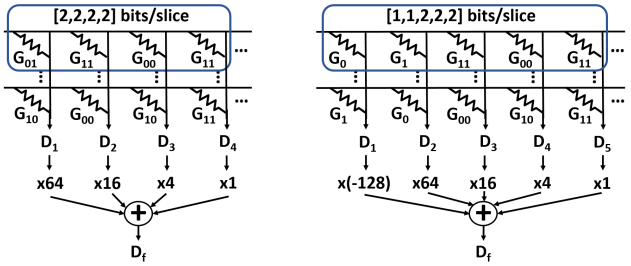
Clearly, unsigned binary arithmetic leads to higher output error due to weighted sum of errors while 2's complement arithmetic reduces output error due to weighted subtraction of errors.

D. Error Analysis for Different Bit-slicing Schemes

Non-zero G_{min} error at the output of i^{th} crossbar column such as that of Fig. 5 can be expressed as:

$$E_i = \frac{N_i{\cdot}\delta}{S} \tag{2}$$

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. Integrating these column-wise errors E_i's results in final error E_f as per Eq. 1a and Eq. 1b for BBS and UBS, respectively. UBS intends to reduce the error E_i compared to BBS by providing larger S (see Eq. 2) using smaller slice sizes for some important columns like MSBs as shown in Fig 4. 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 Eq. 1a can lead to high accumulated E_f for BBS despite having smaller N_i's. On the other hand, the impact of higher N_i's on E_f in UBS can be severely diminished thanks to weighted subtraction (due to 2's complement arithmetic) as shown in Eq. 1b. The higher MSB sensing margin in UBS can further suppress the impact of residual E_f that remains after the weighted subtraction, potentially leading to much less resultant E_f in UBS than BBS and subsequently resulting in better neural network accuracy. This also shows that if 2's complement arithmetic and large sensing margin are not present together in UBS, either increase in N_i's or residual E_f will remain uncompensated leading to poor neural network accuracy. Even though HBS [10] also provides a larger sensing margin S for some slices like UBS, increase in N_i's coupled with weighted accumulation of errors (see Eq. 1a) due to its unsigned binary arithmetic can potentially lead to poor neural network accuracy compared to both BBS and UBS.



(a) BBS. (b) UBS.

Fig. 5. Accumulation of partial digital outputs in a crossbar.

IV. Simulation Results

A. Simulation Setup

We have developed a Python-based simulation framework using in-situ multiply-accumulate (IMA) unit similar to [7] (shown in Fig. 2) which is compatible with UBS, BBS and HBS. Table I shows the simulation setup details. We consider two scenarios for our simulation: a) maximum 1 bit/slice for which we use [1,1,1,1,1,1,1,1] bits/slice for each of the three configurations UBS, BBS and HBS; b) maximum 2 bits/slice for which we use [1,1,2,2,2] bits/slice (FSC) for UBS, [2,2,2,2] bits/slice for BBS [7], [8] and [1,1,2,2,1,1,] bits/slice for HBS [10]. Note that relative accuracy is computed using ideal

TABLE I
Simulation Setup Details.

Network	Fully-Connected Network, 2 Hidden Layers
No. of Neurons	784-100-50-10
Datasets	MNIST [12], Fashion-MNIST (FMNIST) [13]
Training	Software: Floating Point (Gradient Descent)
Inference	Crossbar: Fixed point Weights:8-bit (6 fraction), Inputs:16-bit (10 fraction)

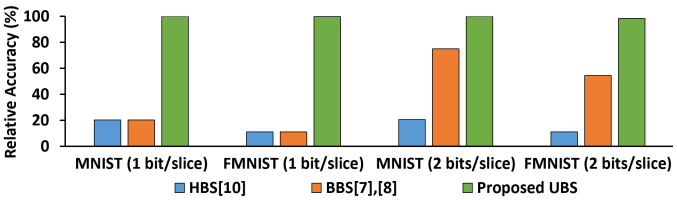


Fig. 6. Neural network accuracy for conductance on/off ratio 40.

fixed-point accuracy of 97.73% and 88.34% for MNIST and FMNIST datasets, respectively.

B. Impact of Bit-slicing Schemes

Fig. 6 shows the simulation results for the three designs under consideration; UBS achieves up to 8.8× and 1.8× accuracy compared to BBS for 1 bit/slice and 2 bits/slice scenarios, respectively. HBS achieves poor accuracy compared to BBS for 2 bits/slice indicating that the increase in N_i's dominates the increase in S for HBS (discussed in Section III-D). HBS and BBS both end up with identical accuracy for 1-bit/slice due to same arithmetic and slice size. Fig. 7 confirms the necessity of *combining* 2's complement arithmetic with high sensing margin to realize the best accuracy for UBS.

C. Impact of Conductance On/off Ratio

UBS outperforms BBS and HBS across different conductance on/off ratios as shown Fig. 8. For FMNIST dataset and conductance on/off ratio 30, UBS recovers 96.6% while BBS recovers only 46.8% of baseline accuracy. For the simpler MNIST dataset, accuracy benefits are even higher.

D. Impact of Conductance Variation

Current subtraction technique (CST) [11] reduces the impact of non-zero G_{min} error on BBS and HBS but becomes less effective at higher variation. As shown in Fig. 9, UBS provides better accuracy at higher variation due to greater MSB sensing margin and 2's complement weighted subtraction reducing errors due to both variation and non-zero G_{min}. High variation tolerance of UBS can also reduce the number of write-verify cycles due to wider permissible target current range resulting in energy saving for memristor programming.

E. Hardware Performance Evaluation

Table II shows IMA performance for BBS [7], [8] and proposed UBS. For 1 bit/slice, energy overhead for UBS is negligible as it needs only a few 2's complement circuits and shift-and-add operations which are inexpensive in hardware [7]. For 2 bits/slice, UBS achieves 80.1% higher accuracy at 14.4% more energy than BBS, resulting in 57.5% higher Figure-of-Merit (FoM) which emphasizes its effectiveness.

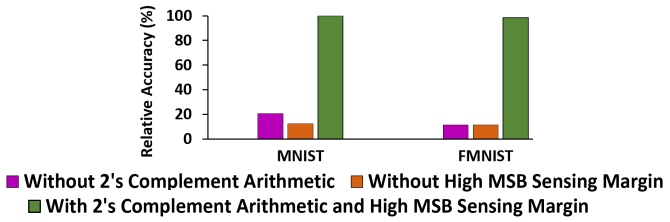
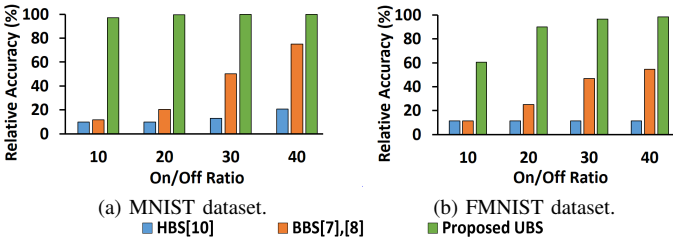
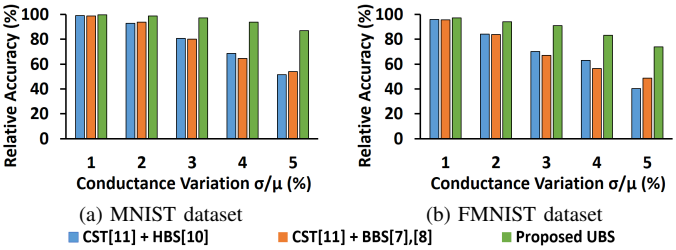


Fig. 7. Accuracy with [1,1,2,2,2] bits/slice for UBS.



(a) MNIST dataset. (b) FMNIST dataset.

Fig. 8. Impact of conductance on/off ratio.



(a) MNIST dataset (b) FMNIST dataset

Fig. 9. Impact of conductance variation.

TABLE II
IMA PERFORMANCE METRICS.

Metric	State-of-the-Art [7], [8]	Proposed Unbalanced Bit-slicing
Area	13120 μm^2	12640 μm^2
Power	24.1 mW	23.5 mW
FoM*	205.1 GOPS/W	323.1 GOPS/W

* Figure-of-Merit (FoM) = Accuracy×Energy efficiency, where energy efficiency is expressed in *giga operations per second per watt* (GOPS/W).

V. Conclusion

This work has shown that by integrating some smart features in the micro-architecture (e.g., unbalanced bit slicing) of memristor based neural networks, one can deal with accuracy degradation and non-idealities (e.g., variability) of the used device technologies. Exploring these mechanisms in combination with others related to the way the mapping is done on the crossbar (e.g., calculation) and the way the design of peripheral circuits is performed could provide significant accuracy improvement even in the presence of non-idealities.

References

[1] C. Szegedy *et al.*, "Going Deeper with Convolutions," in *CVPR*, 2015.
[2] R. Collobert *et al.*, "Natural Language Processing (Almost) from Scratch," *JMLR*, 2011.
[3] S. Hamdioui *et al.*, "Memristor For Computing: Myth or Reality?" in *DATE*, 2017.
[4] S. Hamdioui *et al.*, "Memristor Based Computation-in-Memory Architecture for Data-Intensive Applications," in *DATE*, 2015.
[5] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature Materials*, 2019.
[6] A. Sebastian *et al.*, "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, 2020.
[7] A. Shafiee *et al.*, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *ISCA*, 2016.
[8] A. Ankit *et al.*, "PUMA: A Programmable Ultra-efficient Memristor-based Accelerator for Machine Learning Inference," in *ASPLOS*, 2019.
[9] P. Chen and S. Yu, "Technological Benchmark of Analog Synaptic Devices for Neuroinspired Architectures," *IEEE Design and Test*, 2019.
[10] A. Ankit *et al.*, "PANTHER: A Programmable Architecture for Neural Network Training Harnessing Energy-Efficient ReRAM," *TC*, 2020.
[11] P. Chen *et al.*, "Mitigating Effects of Non-ideal Synaptic Device Characteristics for On-chip Learning," in *ICCAD*, 2015.
[12] Y. Lecun *et al.*, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, 1998.
[13] H. Xiao *et al.*, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv*, 2017.