Stuck-at-Fault Immunity Enhancement of Memristor-Based Edge AI Systems

Md. Oli-Uz-Zaman, *Student Member, IEEE*, Saleh Ahmad Khan, *Student Member, IEEE*, William Oswald, *Student Member, IEEE*, Zhiheng Liao, and Jinhui Wang, *Senior Member, IEEE*

***Abstract*— Deep Neural Networks (DNNs) are widely used in edge AI. But the complex perception and decision-making demands the overlarge computation and makes the DNN architecture very sophisticated. Memristors have multilevel resistance property that enables faster in-memory DNN computation to remove the bottleneck caused by the von Neumann architecture and CMOS technology. However, the Stuck-at-Fault (SAF) defect of memristor generated from immature fabrication and heavy device utilization makes the memristor-based edge AI commercially unavailable. To mitigate this problem, an Adaptive Mapping Method (AMM) is proposed in this paper. Based on the analysis for the VGG8 model with CIFAR10 dataset, the experiment results show that the AMM is efficient in restoring the inference accuracy up to 90% (the original accuracy without SAF) under SAFs from 0.1% to 50%, where Stuck-at-One (SA1): Stuck-at-Zero (SA0) = 5:1, 1:5, and 1:1. Additionally, the AMM has a significant immunity against the nonlinearity and conductance drift. The AMM improves the accuracy more than 50% in presence of high nonlinearity LTP = 4 and LTD = −4 and the standard conductance drift (10 years at 85 degree centigrade) nearly has no influence on the inference accuracy of the DNN in edge AI with the AMM.**

***Index Terms*— Memristor, deep neural network (DNN), artificial intelligence (AI), edge system, stuck-at-fault (SAF), inference accuracy, nonlinearity, conductance drift.**

I. Introduction

NOWADAYS a DNN (Deep Neural Network) model deployed on edge devices is more popular to fulfil the desire of edge AI (Artificial Intelligence). This is because DNNs have achieved a tremendous success due to their unparallel performance in important applications, such as computer vision, image processing, natural language processing, etc., [1], [2], and meanwhile edge devices are highly effective in the Internet of Things (IoT) systems to perform computation, communication, and storage functions. For example, in the Autonomous Vehicles industry, DNNs along with a large number of edge devices have been exploited for sensing, localization, perception, and decision making [3]. However, with the increasing customer demand, DNN structures become more complex and require huge computational resources. Since the downscaling of the conventional CMOS (Complementary Metal Oxide Semiconductor) technology is coming to the plateau [3], [4], the CMOS-based edge AI hardware are facing insurmountable challenges to deal with such DNN problems: 1) the computing speed of the CMOS-based edge AI has no space to improve and cannot further accelerate AI tasks; 2) the COMS-based edge AI hardware consumes large power during performing the vector matrix multiplication in the DNN model. This situation creates an undesirable standstill towards the further advancement of the edge AI in the IoT industry [5].

As the emerging non-volatile memory, memristors can be a rescuer from this deadlock. Following Professor Chua's concept about memristor from 1971, Hewlett-Packard discovered world's first physical memristor device in 2008 where a Titanium Dioxide (TiO_2) and an oxygen deficient layer (TiO_{2-x}) was sandwiched between two platinum electrodes [6]. So, the memristor is a metal-insulator-metal (MIM) architecture where the insulator layer is responsible for the storage of information. The internal storage layer is dynamically reconfigurable through electrical stimulation. Memory function of the memristor is obtained from this reconfiguration where the resistance of the insulator layer depends on the history of current that has been flown through it. If the power supply is removed, this programmed state is not lost. Besides non-volatility and multilevel resistive state property, memristor exhibits wonderful characteristics like low computational complexity [7], sub-nanosecond switching speed [8], [9], [10], sub-10-nm scalability [11], low energy dissipation of few pJ per bit [8], [12], [13], [14], [15], long write-erase endurance [16] and CMOS-compatibility [17], [18]. Additionally, since processing and the storage units are separate, shutting data back and forth between them causes huge latency and energy consumption, the conventional von Neuman architecture is becoming obsolete for DNN tasks. To solve this bottleneck, Compute-In-Memory (CIM) approach has been considered where the memory would be integrated into processing task to boost the system. The crossbar architecture and multilevel cell storage (multiple bits per cell) of memristors can very efficiently perform the vector matrix multiplication as the CIM, which is the most pivotal operation in the DNN algorithm.

Although memristor exhibits excellent properties, immature fabrication technique and low manufacturing yield prevent

Manuscript received 2 May 2022; revised 16 August 2022; accepted 12 September 2022. Date of publication 16 September 2022; date of cur-rent version 19 December 2022. This work was supported in part by the National Science Foundation under Grant 2218046, Grant 1953544, and Grant 1855646. This article was recommended by Guest Editor S. Yu. *(Corresponding author: Jinhui Wang.)*

Md. Oli-Uz-Zaman, Saleh Ahmad Khan, William Oswald, and Jinhui Wang are with the Department of Electrical and Computer Engineering, University of South Alabama, Mobile, AL 36688 USA (e-mail: jwang@southalabama.edu).

Zhiheng Liao is with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58105 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JETCAS.2022.3207687.

Digital Object Identifier 10.1109/JETCAS.2022.3207687

memristor-based computing system to be commercially available. Hard Faults and Soft Faults are two kind of faults that can be found in memristors. Memristors with soft fault, e.g., Read-One-Disturb (R1D) and Read-Zero-Disturb (R0D) flip the original stored bits during the read operation. This fault can be removed since the resistance of the oxide layer is still changeable. But hard fault, also known as Stuck-at-Fault (SAF), is impossible to remove. SAF denotes a device when the resistance of a memristor freezes at High Resistive State (HRS) or Low Resistive State (LRS) [19]. Since the resistance variation is directly related to the mapped weights, the defective memristor will provide wrong weight and result in the inference error to the output of the DNN.

To increase the immunity against SAF defect, several works have been proposed so far. Previously proposed hardware-based solutions in [20], [21], [22], [23], and [24] have some limitations. They work in the following sequence: Learning, Retraining, and Remapping. In the learning phase, a complex algorithm classifies the synaptic weights into significant and insignificant weights. The classification is based on the influence of the weights on the performance of the network. This process causes high computational cost and hinders the possibility of getting a real time response from the edge device. In retraining phase, the SAF defects in memristor crossbar is mimicked. Here, another complex algorithm compensates the SAF caused computation error by retuning the weights over and over again. Repeated weight initialization and updating are needed. This repetitive process causes additional delay. At last in remapping phase, a complex control circuits prevent those significant weights to be mapped to the defective cells obtained in retraining phase. Those methods were successful to restore the high accuracy up to 20% SAFs. But Random patterns in SAF defect require individual optimization for each memristor array that is impossible when it comes to mass memristor-based edge device production. Then a software-based solution is proposed in [25]. This is a good solution as it does not require any individual optimization. But software-based approaches struggle with enormous amount of latency and power consumption. For example, when the IBM team developed a cortical simulation at the complexity of the cat brain, the response was hundred times slower than the original neuron firing rate [26]. Besides, the power consumption was 1.4 MW while the power consumption of more complex task performed by human is only 10 W [27].

Apart from SAF, memristor also suffers from nonlinear-ity which creates inaccurate weight updating and reduces the inference accuracy of the DNN. The required width or amplitude of input signals is highly desirable for achieving the needed conductance of the memristors. But nonlinearity makes this process challenging and hence degrades the inference accuracy [28], [29]. What is more, when nonlinearity incorporates with the SAF, the DNN model gets completely damaged very quickly.

So, a new technology – Adaptive Mapping Method (AMM) is proposed in this paper to mitigate the influence of SAFs as well as high nonlinearity without existing limitations of previously proposed techniques. Our proposed AMM will eliminate the complex control circuits and additional algorithms for individual optimization. This paper will make the following contributions:

- The weight distribution of the VGG8 model using CAIFER 10 dataset is presented and analyzed.
- Considering the sporadic nature of the SAF (SA1 and SA0) from device to device, when the memristor crossbar arrays suffer from different defected conditions (0.1% to 50%) with different SAF ratios (SA1:SA0 = 5:1, 1:5 and 1:1), the accuracy drop of the DNN model is presented.
- Novel mapping technique – AMM is proposed to restore the high accuracy from different defected conditions (0.1% to 50%) with different SAF ratios (SA1:SA0 = 5:1, 1:5 and 1:1).
- Considering the nonlinearity and conductance drift, the AMM is further verified under LTP=4 and LTD=-4, as well as with the standard conductance drift (LTP: long-term potentiation; LTD: long-term depression).
- The design flow and applicability is summarized.
- The proposed AMM is compared with state-of-the-art.

The rest of the paper is organized as follows. In Section II, an Adaptive Mapping Method (AMM) is introduced in detail. The results and discussions regarding accuracy restoration, chip area estimation, and influence by non-linearity and conductance drift, have been presented in Section III. The Design Flow, Applicability, and Comparison with State-of-the-Art are described in section IV, V, and VI respectively. Finally, the conclusion is drawn in VII.

II. Methodology

A. Stuck-at-Fault (SAF)

When the resistance state of the memristor is stuck at LRS, it is known as SA0 defect. A newly fabricated memristor possesses extremely high resistance. So, a forming process is required at the wafer level to initialize the memristor for regular read/write operations. The forming process is an action of inserting high tester voltage in every memristor for decreasing the resistance level to the normal LRS. This highly sophisticated process should last around 100 us. The delicate insulator layer in the metal-insulator-metal (MIM) structure can severely be compromised during this process. Thus, some memristors would be overly formed because of variations in the unstable taster voltage and the thickness of the insulator layer. The resistance of the overly formed memristors stay at LRS forever and input stimulus pulse(s) fail(s) to change its resistance state.

On the other hand, When the resistance state of the memristor is stuck at HRS, it is known as SA1 defect. As shown in Fig. 1, the word line (WL), bit line (BL), and select line (SL) are the three terminals to access each memristor inside the crossbar. But the broken WL makes memristor cells inaccessible for new write operation. Broken WL creates an open circuit where the resistance is unlimited. When the read circuit tries to read the resistance of those memristors, it always mistakes the cell as in HRS.

In a previous research, it has been found that 9.04% and 1.75% memristor cells are affected by SA1 and SA0 respectively, which is approximately SA1:SA0=5:1 [19]. But

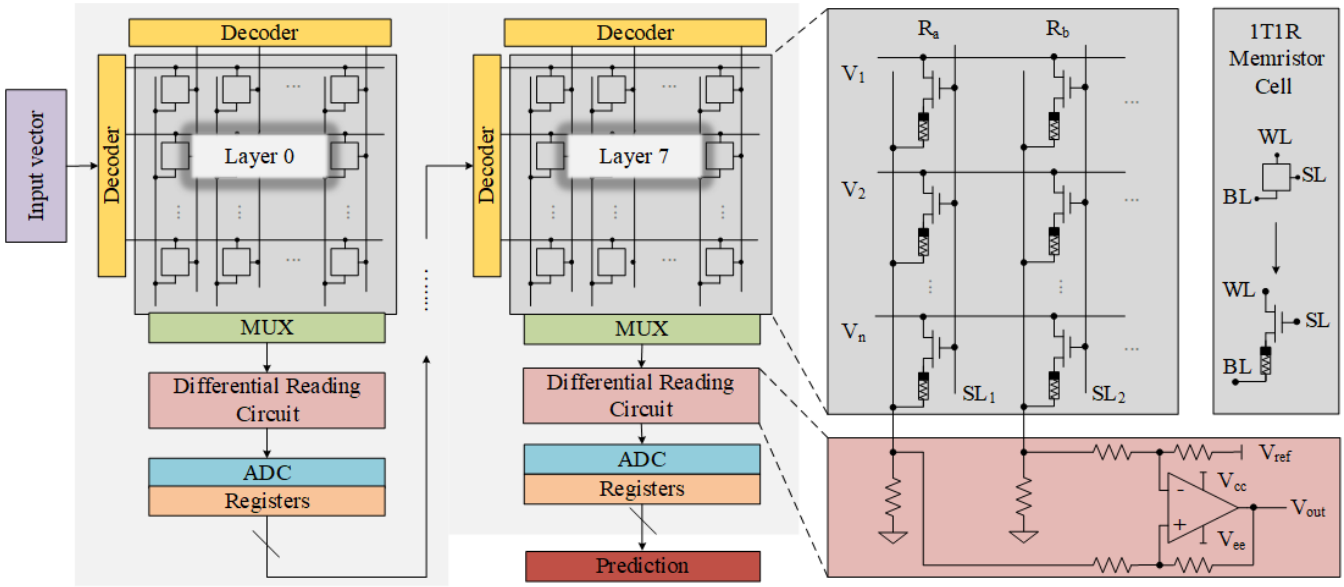

Fig. 1. Hardware implementation for adaptive mapping method (AMM).

this is not always the case. This ratio may vary from device to device. It has also been mentioned in [19] that over forming can cause 60% of the memristor cells to be SA0.

Actually, SAF is prevalent in all kinds of RRAM-based memristors. Such as HfO_2 [19], *Ag:a-Si* [30], and TiO_x (memristor in HP lab with 1T1R: One Transistor One memristor) [31], [32] and 1R structure [33]), all have the same mechanism for SA1 and SA0 inside the crossbar array.

B. Weight Distribution

To validate the effectiveness of the memristor based edge AI system, a VGG8 model along with CIFAR10 dataset are used. Fig. 2 shows the overall weight distribution of VGG8 model trained by CIFAR10 dataset. VGG8 model contains 12.97 million synapses to represent weights. Among the 12.97 million synaptic weights, 43.38% is negative, 25.53% is positive, and 31.18% is neutral weight. As listed in Fig. 2, the mean values of the layers are inclined towards zero and the standard deviations implies that those values are clustered closely. Besides, 99% of the weights are situated within $\pm(3 \times 10^{-7})$.

Since the weights of the DNN model is represented in terms of raw conductance value, the weights are in the range of $10^{-7}S$ as shown in Fig. 2. The *Ag:a-Si* memristor [30] used in this research, as shown in Fig. 6, has a very wide conductance range between $0.1\mu S$ to $1\mu S$. Any conductance range among this wide range can be selected to map the weight range [-1, 1]. The selection of the conductance range depends on the weight precision and the selection of the number of conductance level [34].

C. Adaptive Mapping Method (AMM)

After analyzing the weight distribution in Fig. 2, an Adaptive Mapping Method (AMM) is proposed in this paper. SAF causes huge discrepancy between the original weight and mapped weight. Hence, accuracy degrades significantly even with a very small amount of SAFs inside the memristor crossbar array. The AMM maximally avoids the negative impact of SAF cells for systems and bring back the high accuracy. Since the SAF varies from device to device, our proposed AMM works for all the possible cases where the ratio of SA1:SA0 can be 5:1, 1:5, or 1:1.

1) AMM When SA1:SA0 = 5:1: According to conventional mapping, weights from the algorithm ranging from [-1, 1] will be mapped to memristor devices based on the resistance level [LRS, HRS]. The AMM will use the same resistance level [LRS, HRS], but it rearranges the same weights between [0, 1]. When SA1:SA0 = 5:1, most of SAFs are SA1. The algorithm for the AMM is as follows.

$$W_a = \begin{cases} 1 & W \geq 0, \\ 1 - |W| & W < 0, \end{cases} \tag{1}$$

$$W_b = \begin{cases} 1 - W & W > 0, \\ 1 & W \leq 0, \end{cases} \tag{2}$$

where, W_a and W_b are the two positive portions of a single weight stored in two different memristors. The desired weight is extracted from the simple subtraction (W_a-W_b) during the execution.

As shown in Fig. 3, when SA1:SA0 = 5:1, the AMM splits the original weight 0.3 into 1 and 0.7. Those two weights will be mapped in two memristors and an op-amp based subtractor will extract the original weight (1 - 0.7) during the execution. In the conventional mapping, all the weights from the left side of the Fig. 3 directly map to memristors, and there are no "1". So, the conventional mapping faces a challenge as a big discrepancy between the original weight and the mapped weight. But after the AMM, 61.11% cells are mapped to "1", as shown in the right side of Fig. 3. The mapped "1" replace most of SA1s. In this case, the accuracy degradation can be greatly suppressed.

Layer	Mean	STD
0	-3.05×10^{-8}	1.02×10^{-7}
1	-2.81×10^{-8}	1.02×10^{-7}
2	-3.47×10^{-8}	1.06×10^{-7}
3	-3.44×10^{-8}	1.06×10^{-7}
4	-3.48×10^{-8}	1.05×10^{-7}
5	-3.46×10^{-8}	1.07×10^{-7}
6	-3.46×10^{-8}	1.07×10^{-7}
7	-3.58×10^{-8}	1.07×10^{-7}
Overall	-3.45×10^{-8}	1.07×10^{-7}

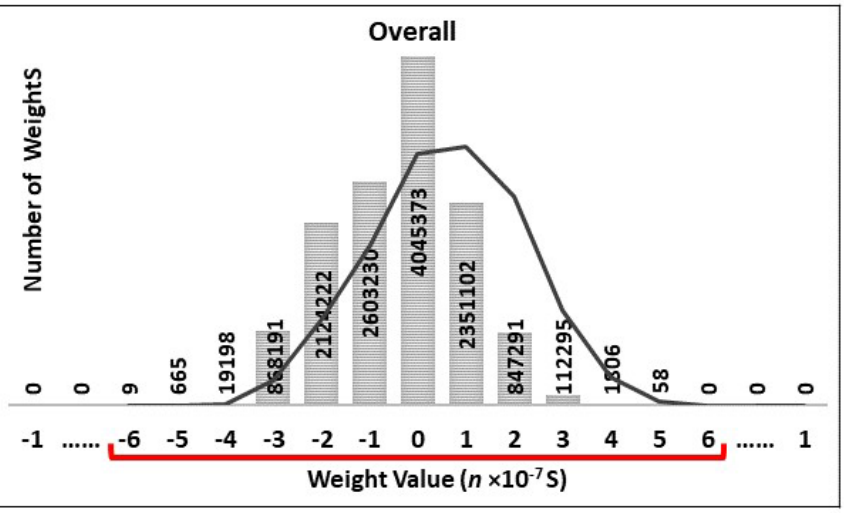

Fig. 2. Overall weight distribution of VGG8 model trained by CIFAR10 dataset.

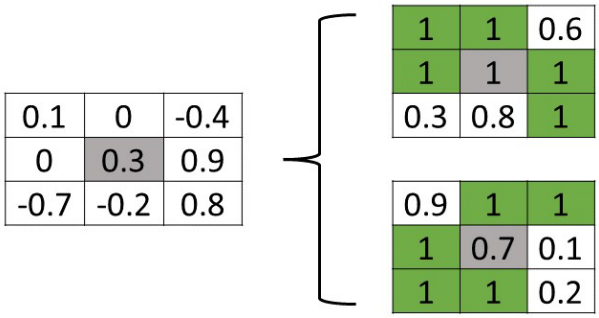

Fig. 3. Adaptive mapping method (AMM) When SA1:SA0 = 5:1.

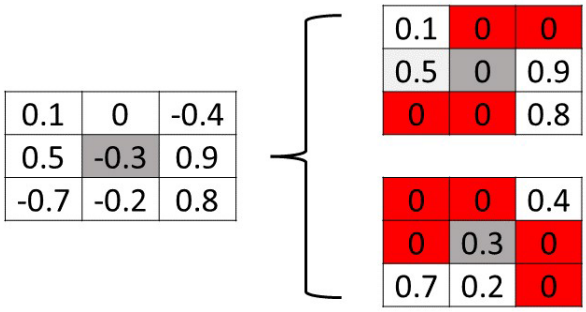

Fig. 4. Adaptive mapping method (AMM) when SA1:SA0 = 1:5.

2) AMM When SA1:SA0 = 1:5: When SA1:SA0 = 1:5, SA0s are dominant in the memristor crossbar array. The algorithm for the AMM is as follows.

$$W_a = \begin{cases} W & W \geq 0, \\ 0 & W < 0, \end{cases} \tag{3}$$

$$W_b = \begin{cases} 0 & W > 0, \\ |W| & W \leq 0, \end{cases} \tag{4}$$

Same as before, each weight is split into two positive weights W_a and W_b, and be mapped to two different memristors. During the execution, the subtractor subtracts the two weights (W_a-W_b) and bring back the original algorithmic weight.

Fig. 4 shows that negative weight -0.3 is split into 0 and +0.3 and is mapped in two memristors. At the end the op-amp based subtractor subtract (0-0.3), the original weight (-0.3) is brought back. Since the conventional mapping takes place between [−1, 1] according to the resistance state [LRS, HRS], memristors stuck at LRS always report -1. But, the range of weight values are changed to [0, 1] with respect to the same resistance level [LRS, HRS] after applying the AMM. So the newly programmed memristors will provide "0" when it is stuck at LRS. In the conventional mapping, the weights from the left side of the Fig. 4 is directly mapped to the crossbar array where only a single "0" can be seen. But as shown in the right side of the Fig. 4, 55.56% weights are mapped to "0" according to the AMM. So, the huge amount of "0" will surely replace SA0s and work towards decreasing the discrepancy between the original weight and the mapped weight, therefore improving inference accuracy of the DNN model with SAFs.

3) AMM When SA1:SA0 = 1:1: In previous conditions, the AMM works in such a way that most of weights inside the crossbar array are mapped to either HRS or LRS based on the dominance of the SAF. But sometimes SA1 and SA0 are equal and happen simultaneously, for example, SA1:SA0 = 1:1. To handle this situation, the AMM uses the same approach of splitting a single weight into two positive weights. But it creates enormous amount of "1" as well as "0" at the same time so that most of the SA1 and SA0 are replaced by those newly mapped "1" and "0". This condition follows the following algorithm.

$$W_a = \begin{cases} W & W > 0, \\ 0 & W < 0, \\ 1 & W = 0, \end{cases} \tag{5}$$

$$W_b = \begin{cases} 0 & W > 0, \\ |W| & W < 0, \\ 1 & W = 0, \end{cases} \tag{6}$$

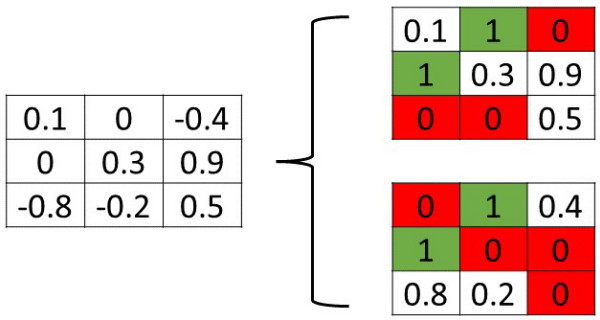

Fig. 5. Adaptive mapping method (AMM) when SA1:SA0 = 1:1.

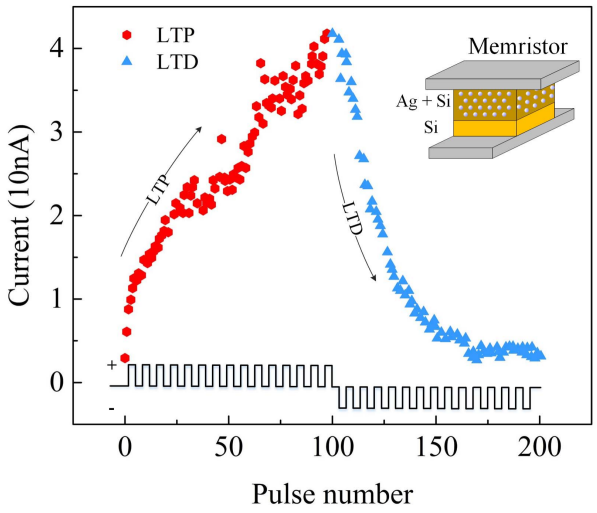

Fig. 6. *Ag:a-Si* based memristor device.

Fig. 5 shows a demonstration of the AMM when SA1:SA0 = 1:1. Newly mapped crossbar achieves 22.22% cells to be mapped to HRS and 38.89% cells to be mapped to LRS.

D. Hardware Implementation

Hardware implementation of the AMM is shown in Fig. 1 [35]. According to Equation 1, 2, 3, 4, 5, 6, the AMM will create two positive weights W_a and W_b from the original weight W of any polarity. W_a and W_b is mapped to two memristors based on resistance state R_a and R_b [35], [36].

III. Result and Discussion

A physical 40 nm *Ag(Silver):a-Si* (amorphous Silicon) memristor device, as shown in Fig. 6 [30], is manufactured and tested thoroughly for extrapolating the real-world behavior of the memristor-based edge AI systems. Characteristics of the *Ag(Silver):a-Si* memristor is incorporated into DNN+NeuroSim platform for evaluations. DNN+NeuroSim is an integrated framework that emulates neural networks (DNN) inference performance on the memristor-based hardware [34]. An 8-layer DNN model VGG8 for CIFAR10 dataset is utilized.

A. Weight Distribution After Applying AMM

To enhance the immunity against SAFs when SA1:SA0 = 5:1, after applying the AMM, the initial weight distribution

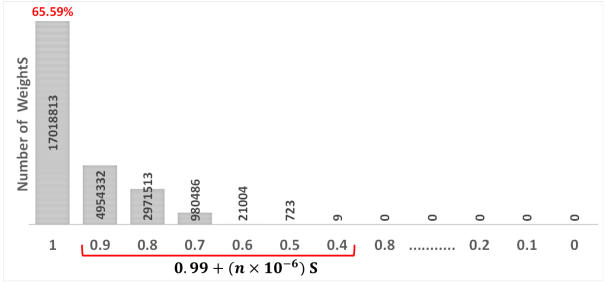

Fig. 7. Weight distribution of VGG8 model after applying AMM for SA1:SA0 = 5:1.

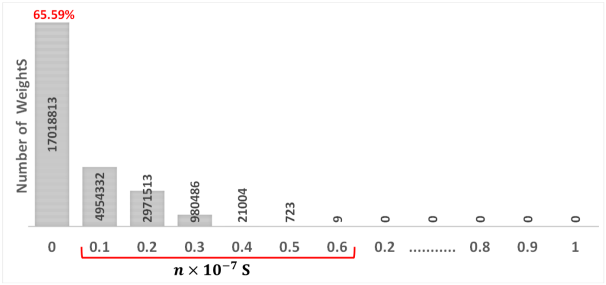

Fig. 8. Weight distribution of VGG8 model after applying AMM for SA1:SA0 = 1:5.

of the VGG8 model, as shown in Fig. 2, is altered into a different shape, as shown in Fig. 7 [8]. Algorithm 1 shows the pseudocode that enabled this feat. When SA1 is dominant, the algorithm maps 69.96% weights to "1" or HRS. As a result, most of the newly mapped "1" replaces the defective cells that are stuck at HRS (SA1). Similarly, the other 30.04% cells are mapped to sub-1 regions (less than 1 but greater than or equal 0.9). If those sub-1 values are mapped to the SA1 defective cell, the deviation between the mapped weight and the weight from the algorithm is very insignificant. Accordingly, it results in very low accuracy loss in extreme SAF conditions.

Similarly, if the SA0 is dominant inside the crossbar, that is SA1:SA0 = 1:5, through the Algorithm 2, the AMM maps 65.71% of the weights to the "0" or LRS, and the rest of the 34.29% weights are also mapped to near zero regions, as shown in Fig. 8. Most of the newly mapped "0" replaces the defective cells that are stuck at LRS (SA0). Therefore, memristors affected by the SA0 act like a defect free device and do not contribute much to the degradation of the inference accuracy of the DNN.

However, when both parts of the SAF are dominant, Algorithm 3 assigns a large number of weights to both LRS and HRS. As shown in Fig. 9, 34.41% weights are converted to "0" or LRS and 31.82% weights are converted to "1" or HRS in case of SA1:SA0 = 1:1. The rest of the weights are very small, and most of them almost equal to zero which are not so much affected by the SA0 defects. By mapping enormous weights to "0" or "1", the AMM creates significant immunity against SA1:SA0 = 1:1.

B. Accuracy With AMM Under Different SAF Conditions

1) Accuracy Restoration When SA1:SA0 = 5:1: The original inference accuracy of the DNN model achieved by the

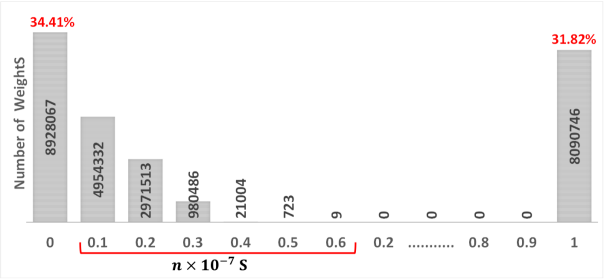

Fig. 9. Weight distribution of VGG8 model after applying AMM for SA1:SA0 = 1:1.

TABLE I
ACCURACY DEGRADATION AND RESTORATION WITH DIFFERENT SAF CONDITIONS WHEN SA1:SA0 = 5:1

SAF (SA1 & SA0)	Accuracy No SAF	Accuracy Before AMM	Accuracy After AMM
0.1%	90%	90%	90%
0.2%		89%	90%
0.5%		80%	90%
1%		42%	90%
2.5%		10%	90%
5%		10%	90%
7.5%		10%	90%
10%		10%	89%
15%		10%	89%
20%		10%	88%
30%		10%	87%
40%		10%	81%
50%		10%	80%

ideal memristor-based edge AI system is 90% without any SAFs. To investigate the deteriorating impact of SAF, SAFs from 0.1% to 50% is introduced. As listed in Table I, before the AMM is used, the inference accuracy decrease starts as 0.2% SAFs. From 2.5% SAFs, the DNN model becomes completely damaged and provides only 10% accuracy which is like random guessing. The AMM can restore the inference accuracy to 90% when the SAF is less than 10%. The AMM is also super-efficient even with extreme conditions. At 50% SAFs, it restores 80% accuracy. According to Table I, when SA1:SA0 = 5:1, SAFs from 0.1% to 50%, inference accuracies of DNNs are improved between 70% and 80%.

2) Accuracy Restoration When SA1:SA0 = 1:5: Similarly, when SA1:SA0 = 1:5, conventional mapping degrades the inference accuracy quicky. As listed in Table II, with as small as 1% SAFs, the DNN model becomes completely damaged and shows only 10% accuracy.

The AMM quickly recovers the original inference accuracy (90%) when SAF is smaller than or equal 10%. Table II also demonstrates that, for the AMM with SA1:SA0 = 1:5, the maximum and minimum accuracy improvements are 80% and 72%, respectively.

3) Accuracy Restoration When SA1:SA0 = 1:1: Unlike the other two conditions, even the conventional mapping can create a relatively strong immunity with low ratio SAFs when SA1:SA0 = 1:1. As listed in Table III, the conventional mapping can achieve high accuracy up to 7.5% SAFs without the AMM. As shown in Fig. 2, 31.18% weights of the DNN model are neutral and the overall mean of the eight layers

TABLE II
ACCURACY DEGRADATION AND RESTORATION WITH DIFFERENT SAF CONDITIONS WHEN SA1:SA0 = 1:5

SAF (SA1 & SA0)	Accuracy No SAF	Accuracy Before AMM	Accuracy After AMM
0.1%	90%	88%	90%
0.2%		83%	90%
0.5%		12%	90%
1%		10%	90%
2.5%		10%	90%
5%		10%	90%
7.5%		10%	90%
10%		10%	90%
15%		10%	89%
20%		10%	88%
30%		10%	87%
40%		10%	85%
50%		10%	82%

TABLE III
ACCURACY DEGRADATION AND RESTORATION WITH DIFFERENT SAF CONDITIONS WHEN SA1:SA0 = 1:1

SAF (SA1 & SA0)	Accuracy No SAF	Accuracy Before AMM	Accuracy After AMM
0.1%	90%	90%	90%
0.2%		90%	90%
0.5%		90%	90%
1%		90%	90%
2.5%		90%	90%
5%		88%	89%
7.5%		83%	89%
10%		54%	89%
15%		10%	88%
20%		10%	87%
30%		10%	83%
40%		10%	73%
50%		10%	66%

TABLE IV
COMPARISON OF ACTUAL DEFECT AND VISIBLE DEFECT AFTER APPLYING ADAPTIVE MAPPING METHOD (AMM) WHEN SAF = 10%

Number of Iterations	Actual Defect	Visible Defect
1	10%	4.02%
2		4.03%
3		4.02%
4		4.03%
5		4.04%
6		3.99%
7		4.05%
8		4.00%
9		4.00%
10		4.01%
11		3.98%
12		4.03%

are inclined towards zero. When the conventional mapping deals with SA1:SA0 = 1:1, the circuit nearly reads equal number of "1" and "-1" from the crossbar array based on the resistance level [LRS, HRS]. After the execution of the DNN model, those "1" and "-1" cancel out each other's adverse impact and bring back the mean value closer to zero again. However, the inference accuracy of the conventional mapping drops fast when the SAFs are greater than 7.5%. It even drops to 10% accuracy at 15% SAFs. With the AMM, the DNN model achieves approximately 90% accuracy even

TABLE V
ACCURACY RESTORATION USING ADAPTIVE MAPPING METHOD IN PRESENCE OF SIGNIFICANT NON-LINEARITY

SAF	Accuracy Before AMM (LTP = 4, LTD = -4)			Accuracy After AMM (LTP = 4, LTD = -4)		
	SA1:SA0 = 5:1	SA1:SA0 = 1:5	SA1:SA0 = 1:1	SA1:SA0 = 5:1	SA1:SA0 = 1:5	SA1:SA0 = 1:1
0.1%	61%	56%	71%	71%	71%	71%
1%	10%	10%	70%	70%	70%	70%
2.5%	10%	10%	65%	70%	69%	65%
20%	10%	10%	10%	64%	67%	60%
50%	10%	10%	10%	50%	50%	24%

with significantly high ratio SAF. According to Table III, the AMM can improve the inference accuracy up to 78%. The reason behind this significant improvement is shown in Table IV. Here a practical scenario, SA1:SA0 = 1:1 with 10% SAF, is taken as an example. After the AMM, approximately 6% of defective cells are successfully replaced by the newly mapped "0" and "1". So, the 10% SAFs acts like a 4% SAFs. Accordingly, as listed in Table III, the AMM restores 89% accuracy where conventional mapping provides only 54% accuracy at 10% SAFs under SA1:SA0 = 1:1. Table III also shows that conventional mapping achieves 88% accuracy with 5% SAFs which validates the result of achieving 89% accuracy when the AMM deals with 10% SAFs (the visible defect is 4%, similar with 5%).

Moreover, the scenario in Table IV is the worst-case scenario. After the AMM is applied, the range of weight is squeezed within [0,1] instead of [-1,1] and 34.41% weights are mapped to near zero region, as shown in Fig. 9. The AMM not only reduces the weight range but also creates a huge near zero regions that helps compensate the accuracy loss caused by the SA0 defective cells.

However, the AMM struggle to achieve very high accuracy with extreme 50% SAF, when SA1:SA0 = 1:1. This is because, when the SAF is 50%, SA1 is 25% and SA0 is 25%, but as shown in Fig. 9, the AMM maps 34.41% cells to "0" and 31.82% cells to "1". SA1 and SA0 defective cells cannot replace all the mapped "1" and "0" respectively, which may possibly entail relatively low inference accuracy.

Overall, AMM creates high precision when it reduces the weight range from $[-1, 1]$ to $[0, 1]$. If the conventional mapping utilizes n level of conductance or resistance to map the weights within $[1, -1]$, AMM changes the range into $[0, 1]$ whereas the number of levels n is kept the same. As a result the conductance or resistance becomes more precise and the change of weights also becomes more precise. The highly precise weight values contribute to improve the accuracy of the DNN model.

C. Accuracy Restoration With Non-Linearity Property

Apart from SAFs, memristor is additionally afflicted by non-linearity that generates improper weight updating and degrades the inference accuracy of the DNN model. Normally, memristor changes conductance/resistance according to current fluxing through it. The weight of the synapse is represented by the resistance/conductance of the memristor, which must be updated frequently during the training and inference process as specified by learning algorithms. Weight

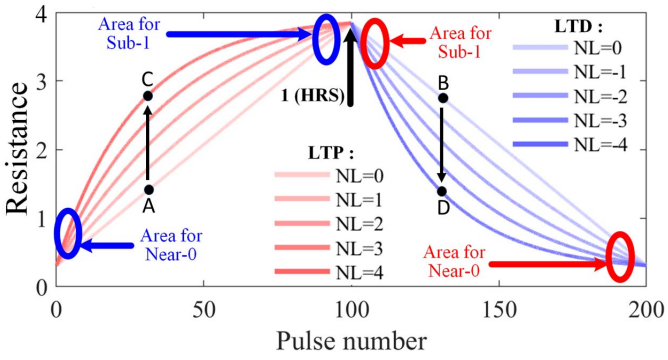

Fig. 10. Non-linearity of memristor cells.

TABLE VI
IMPACT OF DRIFT ON AMM

Drift	SA1:SA0 = 5:1	SA1:SA0 = 1:5	SA1:SA0 = 1:1
Drift to HRS	89%	90%	90%
Drift to LRS	89%	90%	88%
Drift to Middle	90%	90%	90%
Random Drift	89%	89%	89%

increment (or long-term potentiation, LTP) and decrement (or long-term depression, LTD) should ideally be proportionate to the number of writing pulses [28]. However, physical restrictions such as inherent drift and diffusion dynamics of the ions/vacancies cause the inaccurate weight updating with respect to the input stimulus pulse(s) [37], [38]. Ideally, in the weight updating process, the change in the resistance of an ideal synapse device is proportional to number of stimulus pulses. In Fig. 10, the curves (dark) represent the actual resistance value of a memristor device with respect to the number of input pulses where the pulses possess the same duty cycle and the same amplitude, and the straight line (light) represents the hypothetical resistance value of the ideal case. For instance, as shown in Fig. 10, with LTP = 0 and LTD = 0, two ideal memristors produce two resistive states A and B, respectively. However, with strong nonlinearity LTP = 4, LTD = −4, an abrupt incline and decline of the resistive state is obtained, which are labeled as C and D, respectively. This inaccurate weight updating directly impact the overall performance of the DNN model.

The nonlinearity of LTP = 1.75 and LTP = -1.46 is taken into account in all of the results from Tables I, II, and III. However, the conventional mapping totally fails with the inclusion of very high nonlinearity. For example, as shown

TABLE VII

AREA COMPARISON BETWEEN BEFORE AND AFTER ADAPTIVE MAPPING METHOD

Items	Before AMM	After AMM
Total Compute-In-Memory (memristor) array	$3.65072 \times 10^5 \mu m^2$	$7.30144 \times 10^5 \mu m^2$
Total IC Area on chip (Global and Tile/PE local)	$1.20372 \times 10^7 \mu m^2$	$1.20372 \times 10^7 \mu m^2$
Total ADC Area on chip	$5.40220 \times 10^7 \mu m^2$	$5.40220 \times 10^7 \mu m^2$
Total Accumulation Circuits on chip (Adders, shift Adds accumulation units)	$1.05098 \times 10^7 \mu m^2$	$1.05098 \times 10^7 \mu m^2$
Other Peripheries (decoders, mux, switch matrix, buffers, pooling, and activation units)	$9.57562 \times 10^7 \mu m^2$	$9.57562 \times 10^7 \mu m^2$
Weight Gradient Calculation	$9.66804 \times 10^6 \mu m^2$	$9.66804 \times 10^6 \mu m^2$
Differential Reading Circuit (Op amp based subtractors)	0	$3.01333 \times 10^2 \mu m^2$
Total Chip Area	$9.61777 \times 10^7 \mu m^2$	$10.82152 \times 10^7 \mu m^2$ (0.38% Overhead)

TABLE VIII

APPLICABILITY OF ADAPTIVE MAPPING METHOD (AMM)

Worst Case	SA1:SA0 = 5:1	SA1:SA0 = 1:5	SA1:SA0 = 1:1
When all the weights are clustered towards HRS (1)	Here all the 1 will inherently create the strong SAF immunity.	By following Algorithm 2, 50% weights can be mapped to LRS, which will provide a huge SAF immunity.	By following Algorithm 3, 50% weights mapped to LRS and 50% weights to HRS, which will create a huge SAF immunity.
When all the weights are clustered towards LRS (-1)	By following Algorithm 1, 50% weights can be mapped to HRS, which will provide a huge SAF immunity.	By following Algorithm 2, 50% weights can be mapped to LRS, which will provide a huge SAF immunity.	By following Algorithm 3, 50% weights mapped to LRS and 50% weights to HRS, which will create a huge SAF immunity.

TABLE IX

STATE-OF-THE-ART

State-Of-Art	Parameters							
	No Intricate Algorithm	No Complex Read Circuit	No Separate Customization	No Energy Overhead	No Latency Overhead	Consideration of all Possible SAF Ratio	High Accuracy Restoration on all Possible SAF Ratio	Immunity Against Nonlinearity and Drift
[8]	√	√	√	√	√	×	×	√
[21]	×	×	×	×	×	×	×	×
[20]	×	×	×	×	×	×	×	×
[24]	×	√	×	×	×	×	×	×
[23]	×	×	×	×	×	×	×	×
[22]	×	×	×	×	×	×	×	×
[41]	×	√	√	×	×	×	×	×
[42]	×	√	×	×	×	×	×	×
[43]	×	√	×	×	×	×	×	×
[44]	×	√	×	×	×	×	×	×
[45]	√	√	×	×	×	×	×	×
This Work	√	√	√	√	√	√	√	√

in Tables I, II, and III, when 0.1% SAF appears with a ratio of SA1:SA0 = 5:1/1:5/1:1, the accuracy before applying the AMM is 90%/88%/90% respectively, but the accuracies are 61%/56%/71% under the influence of high nonlinearity, as shown in Table V.

The AMM is efficient in restoring the high accuracy even with high nonlinearity. As listed in Table V, when SAF is 1% for SA1:SA0 = 1:5, 1:1, and 5:1, the AMM recovers 70% inference accuracy which was 10% before. Similarly when SAF is 20%, the AMM achieves over 60% accuracy.

This is because the AMM maps majority of the weights at the LRS or HRS where the influence of the non-linearly is absent, as shown in Fig. 10. Hence, the nonlinearity cannot cause any negative impact. As shown in Fig. 7, 8, and 9, when SA1:SA0 = 5:1, 99% weights are mapped to "1" or HRS and sub-1 region; when SA1:SA0 = 1:5, 99% weights are mapped to "0" or LRS and sub-0 region; when SA1:SA0 = 1:1, 99% weights are mapped to HRS and LRS; they are all not affected by the non-linearity.

Since in each case, the tendency of the AMM is to map weights to the HRS or LRS are almost always equal about 65% (as shown in Figs. 7, 8, and 9), the AMM has a similar effectiveness for three conditions SA1:SA0 = 1:5, 1:1, and 5:1. Accordingly, Table V listed a similar accuracy restoration after applying the AMM for three conditions.

D. Retention

Retention is defined as the ability of the memristor device to retain its programmed state over a long period of time [39]. Typically the retention ability of a memristor is more than 10 years at 85 degree centigrade. As shown in Table VI, four conductance drift scenarios have been discussed for the retention analyzed: Drift to HRS, Drift to LRS, Drift to Middle, and Random Drift.

Here 10% SAF along with different drift has been considered in Table VI. The AMM can successfully restored the high inference accuracy even with the different drift conditions. The reason is that we split a single weight into two numbers and store it in two memristors. In some cases, the two weights are affected by the same amount of drifts, and therefore after the subtraction, the same difference is obtained before/after drift. Finally, the DNN can get the desired weight during execution.

E. Chip Area Estimation

The memristor based edge DNN chip is made up of a number of tiles, a global buffer, neural functional computation

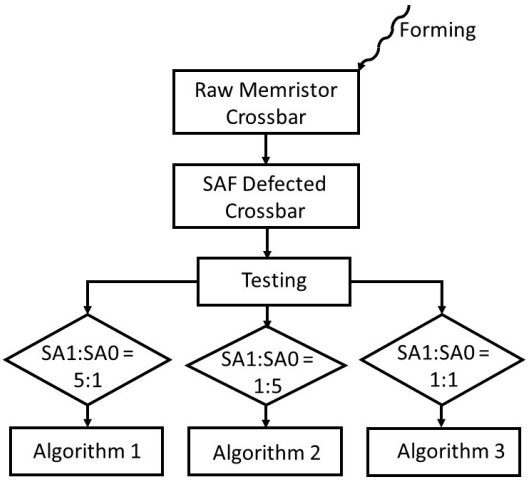

Fig. 11. Flow chart for choosing algorithm.

units such as accumulating units, activation units, pooling units, as well as computation units for weight gradient. In each tile, there are several processing elements (PEs), tile buffers for loading neural activations, accumulation modules for adding partial sums from PEs and output buffers. The total size of a memristor based DNN chip is shown in Table VII.

To improve the accuracy of the DNN model, the AMM requires twice of the number of memristors as the DNN without the AMM. Additionally, the AMM requires one op-amp based subtractor for each two columns of memristor. A 5000×5000 crossbar architecture is enough to store 25 million weights. It demands 2500 op-amps. As shown in Table VII, considering the silicon area of op-amp based subtractors in 32 nm [40] technology, the total chip area with AMM is 0.38% larger than the DNN without the AMM. It can be negligible, considering the great contribution of the AMM on the accuracy and immunity to SAFs.

IV. Design Flow

The design flow of the proposed method is shown in Fig. 11. Forming and heavy device utilization causes the SAF inside the memristor crossbar array. Before we decide which algorithm will be used for AMM, we will firstly test the simple chip to find dominator (SA1 or SA0). The testing platform in [38] or [19] is available. Based on such a dominator, Algorithm 1, 2, or 3 will be selected. Additionally, these algorithms are useful irrespective of the type of the memristor device.

V. Applicability

AMM method can act as a universal method. The method of splitting one weight into two memristors achieves tremendous accuracy restoration. From the analysis of the weights as shown in Fig. 2, it indicates when the weights are clustered towards the weight range [-1, 1], choosing algorithms 1, 2, and 3 can help realize high accuracy. But even if the weight distribution is different with Fig. 2, AMM still has good chance to restore the high accuracy. The worst-case scenario is shown in Table VIII including all the weights are clustered towards HRS or all the weights are clustered towards LRS.

Algorithm 1 AMM When SA1:SA0 = 5:1

Weight Matrix W, Matrix Element Index i, Total Matrix Elements N, Percentage of Stuck-at-1 Fault $SA1$, Percentage of Stuck-at-0 Fault $SA0$.
for $i = 0, 1, 2, \ldots, N-1$
  if $(W_i \geq 0)$
    $W_i^+ = 1$ #1st weight matrix
    $W_i^- = 1 - W_i$ #2nd weight matrix
  else
    $W_i^+ = 1 - |W_i|$
    $W_i^- = 1$
  end if
end for
Append W_i^+ value to a matrix, Mtx_p;
Append W_i^- value to matrix Mtx_n;
#pseudocode for SAF inside the weight matrix
for j = 0,1,2,…,N-1 in Mtx_p
  R_1 = random()×100% #Random num generation
  if $(R_1 \leq SA1)$
    $Mtx_p(j) = 1$
    else
    $Mtx_p(j) = Mtx_p(j)$
  end if
end for
for j = 0,1,2,…,N-1 in Mtx_p
  R_2 = random()×100% #Random num generation
  if $(R_1 \leq SA0)$
    $Mtx_p(j) = 0$
    else
    $Mtx_p(j) = Mtx_p(j)$
  end if
end for
for j = 0,1,2,…,N-1 in Mtx_n
  R_1 = random()×100% #Random num generation
  if $(R_2 \leq SA1)$
    $Mtx_n(j) = 1$
    else
    $Mtx_n(j) = Mtx_n(j)$
  end if
end for
for j = 0,1,2,…,N-1 in Mtx_n
  R_2 = random()×100% #Random num generation
  if $(R_2 \leq SA0)$
    $Mtx_n(j) = 0$
    else
    $Mtx_n(j) = Mtx_n(j)$
  end if
end for
Mtx_p - Mtx_n //final weight matrix

VI. Comparison With State-of-the-Art

As shown in Table IX, our proposed method offers some advantages over the state-of-the-art. So far, most of the SAF

Algorithm 2 AMM When SA1:SA0 = 1:5

Weight Matrix W, Matrix Element Index i, Total Matrix Elements N, Percentage of Stuck-at-1 Fault $SA1$, Percentage of Stuck-at-0 Fault $SA0$.
for $i = 0, 1, 2, \ldots, N-1$
  if $(W_i \geq 0)$
    $W_i^+ = W_i$
    $W_i^- = 0$
  else
    $W_i^+ = 0$
    $W_i^- = |W_i|$
  end if
end for
Append W_i^+ value to a matrix, Mtx_p;
Append W_i^- value to matrix Mtx_n;
#pseudocode for SAF inside the weight matrix
for j = 0,1,2,…,N-1 in Mtx_p
  R_1 = random()×100% #Random num generation
  if $(R_1 \leq SA1)$
    $Mtx_p(j) = 1$
    else
    $Mtx_p(j) = Mtx_p(j)$
  end if
end for
for j = 0,1,2,…,N-1 in Mtx_p
  R_2 = random()×100% #Random num generation
  if $(R_1 \leq SA0)$
    $Mtx_p(j) = 0$
    else
    $Mtx_p(j) = Mtx_p(j)$
  end if
end for
for j = 0,1,2,…,N-1 in Mtx_n
  R_1 = random()×100% #Random num generation
  if $(R_2 \leq SA1)$
    $Mtx_n(j) = 1$
    else
    $Mtx_n(j) = Mtx_n(j)$
  end if
end for
for j = 0,1,2,…,N-1 in Mtx_n
  R_2 = random()×100% #Random num generation
  if $(R_2 \leq SA0)$
    $Mtx_n(j) = 0$
    else
    $Mtx_n(j) = Mtx_n(j)$
  end if
end for
Mtx_p - Mtx_n //final weight matrix

handling approaches develop an intricate algorithm to determine the significant weights first. Then a complex read circuit identifies SAFs free regions for mapping those significant weights. Although using intricate algorithms and the complex read circuit can recover accuracy to a certain extent, but they cause a large hardware and software overhead, and generates additional energy consumption and latency. However, the

Algorithm 3 AMM When SA1:SA0 = 1:1

Weight Matrix W, Matrix Element Index i, Total Matrix Elements N, Percentage of Stuck-at-1 Fault $SA1$, Percentage of Stuck-at-0 Fault $SA0$.
for $i = 0, 1, 2, \ldots, N-1$
  if $(W_i \geq 0)$
    $W_i^+ = W_i$
    $W_i^- = 0$
  end if
  if $(W_i \leq 0)$
    $W_i^+ = 0$
    $W_i^- = |W_i|$
  end if
  if $(W_i == 0)$
    $W_i^+ = 1$
    $W_i^- = 1$
  end if
end for
Append W_i^+ value to a matrix, Mtx_p;
Append W_i^- value to matrix Mtx_n;
#pseudocode for SAF inside the weight matrix
for j = 0,1,2,…,N-1 in Mtx_p
  R_1 = random()×100% #Random num generation
  if $(R_1 \leq SA1)$
    $Mtx_p(j) = 1$
    else
    $Mtx_p(j) = Mtx_p(j)$
  end if
end for
for j = 0,1,2,…,N-1 in Mtx_p
  R_2 = random()×100% #Random num generation
  if $(R_1 \leq SA0)$
    $Mtx_p(j) = 0$
    else
    $Mtx_p(j) = Mtx_p(j)$
  end if
end for
for j = 0,1,2,…,N-1 in Mtx_n
  R_1 = random()×100% #Random num generation
  if $(R_2 \leq SA1)$
    $Mtx_n(j) = 1$
    else
    $Mtx_n(j) = Mtx_n(j)$
  end if
end for
for j = 0,1,2,…,N-1 in Mtx_n
  R_2 = random()×100% #Random num generation
  if $(R_2 \leq SA0)$
    $Mtx_n(j) = 0$
    else
    $Mtx_n(j) = Mtx_n(j)$
  end if
end for
Mtx_p - Mtx_n //final weight matrix

AMM can be used as a ubiquitous solution to avoid all these complexities. Besides, the AMM has approved not to add

energy and latency to the system [8]. Furthermore, the AMM has strong immunity to the nonlinearity and conductance drift.

VII. Conclusion

High integrated density and simple crossbar architecture makes memristor suitable for the implementation of large and complex DNN model in edge AI systems. But unavoidable SAF defects impede its commercial success, because the inference accuracy drop is inevitable. In this paper, the AMM is proposed to deal with such accuracy degradation. The experiment results show that the AMM can restore the interference accuracy to 90% when the SAF is less than or equal to 7.5%/10%/2.5% at SA1:SA0 = 5:1/1:5/1:1. Even in some extreme cases, for example SAF = 50%, the AMM also effective and achieves the accuracy up to 80%/82%/66% at SA1:SA0 = 5:1/1:5/1:1. The AMM remaps the original weights to HRS or (and) LRS and makes it immune against the SAFs no matter which SAF is dominator. Besides, this unique mapping causes memristor-based edge AI system invulnerable towards nonlinearity and conductance drift. As a result, with significant nonlinearity (LTP = 4, LTD = -4), and 2.5% SAF, the DNN with the AMM achieves 70%/69%/65% accuracy at SA1:SA0 = 5:1/1:5/1:1. Finally, as compared with state-of-the-art, our proposed method implies the superiority.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[3] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.

[4] M. M. Waldrop, "The chips are down for Moore's law," *Nature*, vol. 530, no. 7589, pp. 144–147, Feb. 2016.

[5] J. Fu, Z. Liao, and J. Wang, "Memristor-based neuromorphic hardware improvement for privacy-preserving ANN," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 12, pp. 2745–2754, Dec. 2019.

[6] J. J. Yang, M. D. Pickett, X. M. Li, D. A. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nat. Nanotechnol.*, vol. 3, no. 7, pp. 429–433, Jul. 2008.

[7] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: Threshold adaptive memristor model," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 211–221, Jan. 2013.

[8] M. Oli-Uz-Zaman *et al.*, "Mapping transformation enabled high-performance and low-energy memristor-based DNNs," *J. Low Power Electron. Appl.*, vol. 12, no. 1, pp. 10–24, 2022.

[9] A. C. Torrezan, J. P. Strachan, G. Medeiros-Ribeiro, and R. S. Williams, "Sub-nanosecond switching of a tantalum oxide memristor," *Nanotechnology*, vol. 22, no. 48, Dec. 2011, Art. no. 485203.

[10] B. J. Choi *et al.*, "High-speed and low-energy nitride memristors," *Adv. Funct. Mater.*, vol. 26, no. 29, pp. 5290–5296, 2016.

[11] B. Govoreanu *et al.*, "10×10nm^2 Hf/HfO$_x$ crossbar resistive ram with excellent performance, reliability and low-energy operation," in *IEDM Tech. Dig.*, Dec. 2011, pp. 31.6.1–31.6.4.

[12] S. C. Bartling, S. Khanna, M. P. Clinton, S. R. Summerfelt, J. A. Rodriguez, and H. P. McAdams, "An 8mhz 75μa/mhz zero-leakage non-volatile logic-based cortex-m0 MCU SoC exhibiting 100% digital state retention at V($_{DD}$)=0$_V$ with <400ns wakeup and sleep transitions," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2013, pp. 432–433.

[13] Z. Liao, J. Fu, and J. Wang, "Ameliorate performance of memristor-based ANNs in edge computing," *IEEE Trans. Comput.*, vol. 70, no. 8, pp. 1299–1310, Aug. 2021.

[14] N. Sakimura *et al.*, "10.5 A 90nm 20MHz fully nonvolatile microcontroller for standby-power-critical applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2014, pp. 184–185.

[15] P.-F. Chiu *et al.*, "Low store energy, low VDDmin, 8T2R nonvolatile latch and SRAM with vertical-stacked resistive memory (memristor) devices for low power mobile applications," *IEEE J. Solid-State Circuits*, vol. 47, no. 6, pp. 1483–1496, Jun. 2012.

[16] K.-H. Kim, S. Hyun Jo, S. Gaba, and W. Lu, "Nanoscale resistive memory with intrinsic diode characteristics and long endurance," *Appl. Phys. Lett.*, vol. 96, no. 5, 2010, Art. no. 05310.

[17] Q. Xia *et al.*, "Memristor–CMOS hybrid integrated circuits for reconfigurable logic," *Nano Lett.*, vol. 9, no. 10, pp. 3640–3645, Oct. 2009.

[18] J. Fu, Z. Liao, J. Liu, S. C. Smith, and J. Wang, "Memristor-based variation-enabled differentially private learning systems for edge computing in IoT," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9672–9682, Jun. 2021.

[19] C.-Y. Chen *et al.*, "RRAM defect modeling and failure analysis based on March test and a novel squeeze-search scheme," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 180–190, Jan. 2015.

[20] L. Chen *et al.*, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 19–24.

[21] C. Liu, M. Hu, J. P. Strachan, and H. Li, "Rescuing memristor-based neuromorphic design with high defects," in *Proc. 54th Annu. Design Autom. Conf.*, Jun. 2017, pp. 1–6.

[22] L. Xia, M. Liu, X. Ning, K. Chakrabarty, and Y. Wang, "Fault-tolerant training with on-line fault detection for RRAM-based neural computing systems," in *Proc. 54th Annu. Design Autom. Conf.*, Jun. 2017, pp. 1–6.

[23] L. Xia *et al.*, "Stuck-at fault tolerance in RRAM computing systems," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 102–115, Mar. 2018.

[24] B. Zhang, N. Uysal, D. Fan, and R. Ewetz, "Handling stuck-at-fault defects using matrix transformation for robust inference of DNNs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2448–2460, Oct. 2020.

[25] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proc. 53rd Annu. Design Autom. Conf.*, Jun. 2016, pp. 1–6.

[26] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A low energy oxide-based electronic synaptic device for neuromorphic visual systems with tolerance to device variation," *Adv. Mater.*, vol. 25, pp. 1774–1779, Mar. 2013.

[27] E. R. Kandel, "From nerve cells to cognition: The internal cellular representation required for perception and action," *Princ. Neural Sci.*, vol. 1, pp. 381–403, Jan. 2000.

[28] J. Fu, Z. Liao, N. Gong, and J. Wang, "Mitigating nonlinear effect of memristive synaptic device for neuromorphic computing," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 377–387, Jun. 2019.

[29] 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 Design (ICCAD)*, Nov. 2015, pp. 194–199.

[30] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, 2010.

[31] Y.-X. Chen and J.-F. Li, "Fault modeling and testing of 1T1R memristor memories," in *Proc. IEEE 33rd VLSI Test Symp. (VTS)*, Apr. 2015, pp. 1–6.

[32] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, "Modeling, detection, and diagnosis of faults in multilevel memristor memories," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 34, no. 5, pp. 822–834, May 2015.

[33] N. Z. Haron and S. Hamdioui, "On defect oriented testing for hybrid CMOS/memristor memory," in *Proc. Asian Test Symp.*, Nov. 2011, pp. 353–358.

[34] X. Peng, S. Huang, H. Jiang, A. Lu, and S. Yu, "DNN+NeuroSim V2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 11, pp. 2306–2319, Nov. 2021.

[35] O. Krestinskaya, A. P. James, and L. O. Chua, "Neuromemristive circuits for edge computing: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 1, pp. 4–23, Jan. 2020.

[36] G. Yuan *et al.*, "Improving DNN fault tolerance using weight pruning and differential crossbar mapping for ReRAM-based edge AI," in *Proc. 22nd Int. Symp. Quality Electron. Design (ISQED)*, Apr. 2021, pp. 135–141.

[37] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," *Nature Electron.*, vol. 1, no. 1, pp. 22–29, Jan. 2018.

[38] J. Fu, Z. Liao, and J. Wang, "Level scaling and pulse regulating to mitigate the impact of the cycle-to-cycle variation in memristor-based edge AI system," *IEEE Trans. Electron Devices*, vol. 69, no. 4, pp. 1752–1762, Apr. 2022.
[39] P.-Y. Chen and S. Yu, "Reliability perspective of resistive synaptic devices on the neuromorphic system performance," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Mar. 2018, p. 5.
[40] A. Dendouga, S. Oussalah, D. Thienpont, and A. Lounis, "Program for the optimization of an OTA for front end electronics based on multi objective genetic algorithms," in *Proc. 29th Int. Conf. Microelectron.*, 2014, pp. 443–446.
[41] G. Jung, M. Fouda, S. Lee, J. Lee, A. Eltawil, and F. Kurdahi, "Cost- and dataset-free stuck-at fault mitigation for ReRAM-based deep learning accelerators," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 1733–1738.
[42] J. Zhang, D. Kline, L. Fang, R. Melhem, and A. K. Jones, "Dynamic partitioning to mitigate stuck-at faults in emerging memories," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 651–658.
[43] G. Charan, A. Mohanty, X. Du, G. Krishnan, R. V. Joshi, and Y. Cao, "Accurate inference with inaccurate RRAM devices: A joint algorithm-design solution," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 6, pp. 27–35, 2020.
[44] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.
[45] I. Yeo, M. Chu, S.-G. Gi, H. Hwang, and B.-G. Lee, "Stuck-at-fault tolerant schemes for memristor crossbar array-based neural networks," *IEEE Trans. Electron Devices*, vol. 66, no. 7, pp. 2937–2945, Jul. 2019.

Md. Oli-Uz-Zaman (Student Member, IEEE) received the B.S. degree in electronics and telecommunication engineering from the Rajshahi University of Engineering and Technology (RUET), Bangladesh, in 2016. He is currently pursuing the Ph.D. degree with the University of South Alabama, Mobile, AL, USA. His research focuses on AI hardware design and neuromorphic computing.

Saleh Ahmad Khan (Student Member, IEEE) received the B.S. degree in electrical and electronics engineering from American International University-Bangladesh, Bangladesh, in 2019. He is currently pursuing the M.S. degree with the University of South Alabama, Mobile, AL, USA. He received the Dean's Award from American International University-Bangladesh for the best undergraduate capstone project. His research focuses on AI hardware design and neuromorphic computing.

William Oswald (Student Member, IEEE) received the B.S. degree in computer engineering and the M.S. degree in electrical engineering from the University of South Alabama in 2020 and 2021, respectively. He is currently pursuing the Ph.D. degree. He has worked in industry as a Systems Analyst at Packaging Corporation of America from 2019 to 2020. His research interests include computer architecture design, machine learning, and neural networks.

Zhiheng Liao received the B.E. and M.S. degree in electrical engineering from the Beijing University of Technology, China, in 2014 and 2017, respectively, and the Ph.D. degree from North Dakota State University, Fargo, ND, USA, in 2021. His research focuses on the emerging device and algorithm optimization for neuromorphic computing and artificial intelligence (AI) technology.

Jinhui Wang (Senior Member, IEEE) received the B.E. degree in electrical engineering from Hebei University, Hebei, China, and the Ph.D. degree in electrical engineering from the Beijing University of Technology, Beijing, China. He was a Post-Doctoral Fellow with the University of Rochester, Rochester, NY, USA; a Visiting Professor with the State University of New York at Buffalo, Buffalo, NY; and a Visiting Scholar with IMEC, Leuven, Belgium. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of South Alabama, Mobile, AL, USA. He has published over 150 refereed journal/conference papers and holds 20 patents in the area of emerging semiconductor technologies. His research interests include neuromorphic computing, AI technology, low-power, high-performance, and variation-tolerant IC design, 3-D IC, and thermal solution in VLSI. His previous work received the Best Paper Award/Nomination at DATE 2021, ISVLSI 2019, ISLPED 2016, ISQED 2016, and EIT 2016.