RRAM Crossbar-Based Fault-Tolerant Binary Neural Networks (BNNs)

Gebregiorgis, Anteneh; Zografou, Artemis ; Hamdioui, Said

DOI
10.1109/ETS54262.2022.9810414

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 2022 IEEE European Test Symposium (ETS)

Citation (APA)
Gebregiorgis, A., Zografou, A., & Hamdioui, S. (2022). RRAM Crossbar-Based Fault-Tolerant Binary Neural Networks (BNNs). In *Proceedings of the 2022 IEEE European Test Symposium (ETS)* (pp. 1-2). Article 9810414 IEEE. https://doi.org/10.1109/ETS54262.2022.9810414

RRAM Crossbar-Based Fault-Tolerant Binary Neural Networks (BNNs)

Anteneh Gebregiorgis, Artemis Zografou and Said Hamdioui
Computer Engineering Lab, Delft University of Technology, Delft, The Netherlands
{A.B.Gebregiorgis, S.Hamdioui}@tudelft.nl, A.Zografou@student.tudelft.nl

2022 IEEE European Test Symposium (ETS) | 978-1-6654-6706-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ETS54262.2022.9810414

***Abstract*—Computation-In Memory (CIM) using RRAM crossbar array is a promising solution to realize energy-efficient neuromorphic hardware, such as Binary Neural Networks (BNNs). However, RRAM faults restrict the applicability of CIM for BNN implementation. To address this issue, we propose a fault tolerance framework to mitigate the impact of RRAM faults on the accuracy of CIM-based BNN hardware. Evaluation results using MNIST, Fashion-MNIST and CIFAR-10 datasets demonstrate that the proposed framework outperforms the related works as it restores more than 99% of the RRAM fault induced accuracy reduction with relatively less overhead.**

***Index Terms*—CIM, fault tolerance, RRAM, BNN**

I. Introduction

Computation-In-Memory (CIM) with Resistive Random Access Memories (RRAM), integrating computation and storage in the same physical location, has emerged as a promising solution to deploy deep neural networks (DNNs) on resource constrained platforms [1]. However, RRAM devices may suffer from non-idealities and manufacturing defects such as Stuck-at Fault (SAF) [2], [3]. Thus, addressing these is of paramount importance for reliable BNN operation on CIM hardware.

Several software and hardware-based fault tolerance approaches have been proposed to mitigate the impact of SAF in RRAM-based CIM [4], [5], [6], [7], [8], [9]. Some of them focus on optimal mapping [7], [4], while others rely on retraining [5], [6]. However, these solutions have various limitations, such as mapping complexity. Therefore, efficient techniques are needed for a reliable CIM operation.

We propose a fault tolerance framework consisting of three techniques addressing the impact of SAF on the accuracy of DNNs with binary weights, Binary Neural Networks (BNNs), mapped to RRAM-based CIM hardware. The first technique investigates different activation functions in the presence of SAF to choose a fault-tolerant activation function. The second and third techniques further enhance the fault tolerance by applying redundancy and retraining methods, respectively.

II. Proposed Fault Tolerance Framework

Figure 1 shows the proposed fault tolerance framework. First, a fault-tolerant activation function is determined, by evaluating different activation functions in the presence of SAF. Then, it is used as a baseline redundancy and retraining techniques.

A. Fault-tolerant activation function

Activation functions introduce non-linearity on the neuron's output, making the network learn nonlinear behaviors [11], [12]. There are several activation functions and among them three are widely used, namely $Sigmoid$, hyperbolic tangent $Tanh$ and Rectified Linear Unit $ReLU$. These functions have their own pros and cons with respect to non-linearity, fault tolerance etc. This fact is exploited to choose a fault-tolerant activation function for reliable BNNs. As shown in Figure 1(a) the fault tolerance capability of the three functions is evaluated using RRAM crossbar. In both $Sigmoid$ and $Tanh$ functions the neurons are mostly active, making them vulnerable to the impact of SAF. On the other hand, the neurons under $ReLU$ are largely inactive, masking some of the SAFs. Thus, $ReLU$ is expected to be fault-tolerant than $Sigmoid$ and $Tanh$.

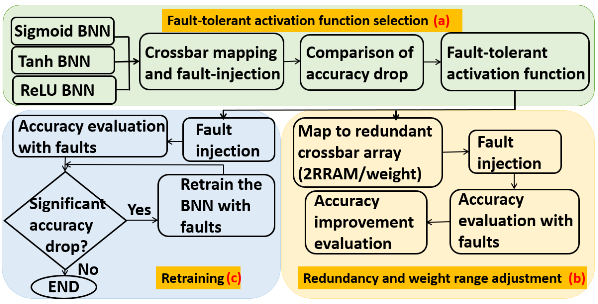

Fault-tolerant activation function selection (a)
Sigmoid BNN
Tanh BNN
ReLU BNN
Crossbar mapping and fault-injection
Comparison of accuracy drop
Fault-tolerant activation function
Accuracy evaluation with faults
Fault injection
Significant accuracy drop?
Yes
No
Retrain the BNN with faults
END
Retraining (c)
Map to redundant crossbar array (2RRAM/weight)
Fault injection
Accuracy improvement evaluation
Accuracy evaluation with faults
Redundancy and weight range adjustment (b)

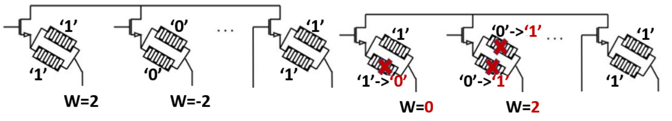
Fig. 1: Proposed fault tolerance framework.

B. Redundancy and weight range adjustment

Redundancy is utilized for fault tolerance not by remapping the faulty weights to secondary devices, but by increasing the range of the stored weights, thus minimizing the impact of SAF (Figure 1(b)). This is realized by altering the crossbar architecture from 1T1R into 1T2R (2 RRAMs in parallel), where the weights are mapped into two RRAMs and the weight range is changed to {-2, +2} as shown in Figure 2(a). When the two RRAM devices have different resistance states (HRS/LRS or LRS/HRS) due to SAF, the value of the mapped weight becomes 0 instead of flipping from ± 2 to ∓2 (Figure 2(b)). Thus, the weight may obtain three different values {-2, 0, 2} and the extra state '0' enhances the fault tolerance. However, when both RRAMs are faulty, the weight could flip from ± 2 to ∓2, which is equivalent to 1T1R.

C. Retraining for fault tolerance

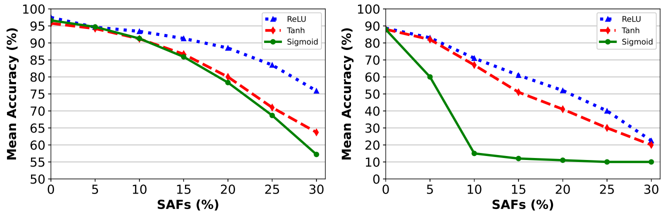
Retraining is another technique which can improve the accuracy significantly, when combined with other orthogonal



(a) Fault free case.

(b) Faulty case.

Fig. 2: Proposed redundancy approach with weight range adjustment [10], from {-1, +1} to {-2, +2}.



(a) MNIST dataset (b) Fashion-MNIST dataset

Fig. 3: Fault tolerance capability of different activation functions.

techniques [13]. The proposed framework exploits the potential of retraining by combining it with the fault-tolerant activation function and redundancy techniques. Retraining requires the location of SAF in the RRAM crossbar in order to exclusively retrain the fault free weights. This is realized by first extracting the SAF distribution, and then a gradient mask is applied to prevent the faulty weights from retraining.

III. Results

A. Fault-tolerant activation function

The effectiveness of fault-tolerant activation function is evaluated using MNIST and Fashion-MNIST datasets (see Figure 3). Figure 3(a) shows that BNNs employing $Tanh$ or $Sigmoid$ functions exhibit similar accuracy degradation for MNIST dataset, whereas the BNN employing $ReLU$ function has relatively less accuracy drop, making it more fault-tolerant. Figure 3(b) shows similar trend for Fashion-MNIST dataset.

B. Redundancy and weight range adjustment

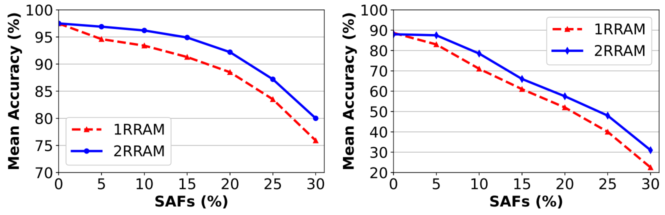
The potential of redundancy and weight range adjustment technique is evaluated using fault-tolerant activation function, $ReLU$ in this case, for MNIST and Fashion-MNIST datasets as shown in Figure 4. For both datasets, redundancy and weight range adjustment technique improves the accuracy by 5-10%.

C. Evaluation of retraining for fault tolerance

Table I presents the accuracy improvement of retraining method combined with fault-tolerant activation function, $ReLU$, for different datasets. It is observed that retraining is able to almost fully recover the accuracy in all cases.

D. Comparison with state-of-the-art techniques

The proposed approach is compared with three state-of-the-art fault tolerance techniques [5], [4], [14], as shown in Table II. The accuracy improvement comparison is conducted using a 2-Layer BNN running MNIST dataset in the presence of 20% SAFs. In the table, the term recovered accuracy is the ratio obtained when the restored accuracy is divided by the baseline accuracy, and the redundancy column (R) indicates the redundant RRAM devices used for fault tolerance.



(a) MNIST dataset (b) Fashion-MNIST dataset

Fig. 4: Accuracy improvement of redundancy technique

TABLE I: Retraining with SAFs

SAF distribution (%)	MNIST		Fashion-MNIST		CIFAR	
	Accuracy (%)		Accuracy (%)		Accuracy (%)	
	Baseline	Retrained	Baseline	Retrained	Baseline	Retrained
0	97.3	97.3	88.22	88.22	90.09	90.09
5	96	97.3	84	87.1	65	89.9
10	95.4	97	71	88	10	89.7
15	93	97.2	62	87.9	10	89.6
20	89	97.19	53	88	9	89

TABLE II: Comparison with related fault tolerance techniques, using a 2-Layer NN for the MNIST dataset, with 20% SAFs.

Related Works	Recovered Accuracy (%)	Retraining	Redundancy (R)
[5]	95.1	Yes	1
[4]	43	No	1
	96		2
[14]	30.1	No	1
	97.6		2
Our proposal	99.8	Yes	1

Authors in [5], use a modified retraining method to achieve 95.1% accuracy, while authors in [4] adopt a redundancy and mapping technique to obtain 96% accuracy. Similarly, the work in [14] uses redundancy and matrix transformations to achieve an accuracy of 97.6%. Both techniques in [14] and [4] add software and hardware overheads (R=2) to improve the inference accuracy. Overall, the proposed fault tolerance framework outperforms the related works as it achieves better accuracy improvement with a comparatively less overhead.

IV. Conclusion

The paper investigated the impact of RRAM defects on CIM-based BNNs and proposed mitigation techniques to reduce their impact. Results showed that these techniques achieve significant inference accuracy improvement with relatively less overhead.

References

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

[2] H. Amrouch *et al.*, “Towards reliable in-memory computing: From emerging devices to post-von-neumann architectures,” in *VLSI-SoC*, 2021.

[3] M. Fieback *et al.*, “Intermittent undefined state fault in rrams,” in *ETS*, 2021.

[4] W. Huangfu *et al.*, “Computation-oriented fault-tolerance schemes for rram computing systems,” in *ASP-DAC*, 2017.

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

[6] L. Xia *et al.*, “Fault-tolerant training with on-line fault detection for rram-based neural computing systems,” in *DAC*, 2017.

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

[8] L. Xia *et al.*, “Stuck-at fault tolerance in rram computing systems,” *Journal on Emerging and Selected Topics in Circuits and Systems*, 2017.

[9] A. Chaudhuri *et al.*, “Hardware fault tolerance for binary rram crossbars,” in *ITC*, 2019.

[10] A. Shafiee *et al.*, “Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” *Computer Architecture*, 2016.

[11] D. Misra, “Mish: A self regularized non-monotonic neural activation function,” *arXiv*, 2019.

[12] S. Sharma, “Activation functions in neural networks,” *towards data science*, 2017.

[13] C. Torres-Huitzil and B. Girau, “Fault and error tolerance in neural networks: A review,” *IEEE Access*, 2017.

[14] B. Zhang *et al.*, “Handling stuck-at-faults in memristor crossbar arrays using matrix transformations,” in *ASP-DAC*, 2019.