

X-Fault: Impact of Faults on Binary Neural Networks in Memristor-Crossbar Arrays with Logic-in-Memory Computation

Felix Staudigl*, Karl J. X. Sturm*, Maximilian Bartel*, Thorben Fetz*,
Dominik Sisejkovic*, Jan Moritz Joseph*, Leticia Bolzani Pöhls†, and Rainer Leupers*

* Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Germany

† Chair of Integrated Digital Systems and Circuit Design, RWTH Aachen University, Germany

{staudigl, sturm, bartel, fetz, sisejkovic, joseph, leupers}@ice.rwth-aachen.de
poehls@ids.rwth-aachen.de

Abstract—Memristor-based crossbar arrays represent a promising emerging memory technology to replace conventional memories by offering a high density and enabling computing-in-memory (CIM) paradigms. While analog computing provides the best performance, non-idealities and ADC/DAC conversion limit memristor-based CIM. Logic-in-Memory (LIM) presents another flavor of CIM, in which the memristors are used in a binary manner to implement logic gates. Since binary neural networks (BNNs) use binary logic gates as the dominant operation, they can benefit from the massively parallel execution of binary operations and better resilience to variations of the memristors. Although conventional neural networks have been thoroughly investigated, the impact of faults on memristor-based BNNs remains unclear. Therefore, we analyze the impact of faults on logic gates in memristor-based crossbar arrays for BNNs. We propose a simulation framework that simulates different traditional faults to examine the accuracy loss of BNNs on memristive crossbar arrays. In addition, we compare different logic families based on the robustness and feasibility to accelerate AI applications.

Index Terms—ReRAM, memristor, faults, reliability, logic-in-memory

I. INTRODUCTION

Non-volatile memories (NVMs) such as resistive RAM (ReRAM) offer advantages over conventional RAMs in terms of density, power consumption, and computing-in-memory (CIM) capabilities. CIM is most promising to address the von Neumann bottleneck by reducing data exchange, thus addressing the limitations of memory-bound AI accelerators [1], [2].

Several architectures using ReRAM have been proposed for CIM-based AI [3]. These accelerators use analog computing on ReRAM crossbars by utilizing Kirchhoff's law (Fig. 1a). While this approach promises power and latency benefits, in practice, many architectures are limited by (1) inefficient analog-to-digital conversion or vice versa, and (2) the low resilience to faults in analog computing. The former is caused by ADC/DAC drawing large amounts of power [3]. The latter results from stored values being altered by write and read accesses [4], [5].

Logic-in-memory [6] is an orthogonal method to analog CIM that does not suffer from the mentioned limitations. It uses binary vectors stored in crossbar columns. Two columns are combined with given logic operations, and the result is stored

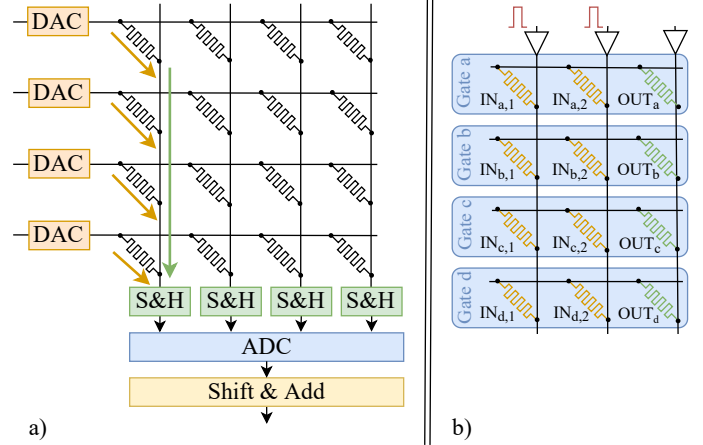


Fig. 1: Comparison of CIM paradigms: (a) Kirchhoff-based analog CIM, and (b) binary logic-in-memory.

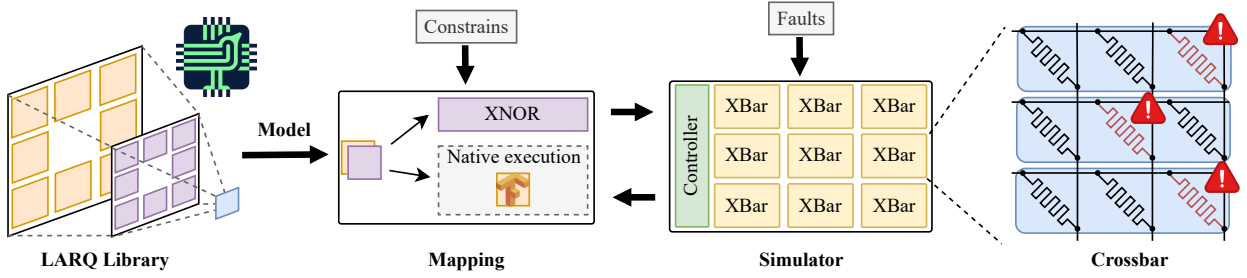
in a third column (Fig. 1b) [7]. This method is less error-prone, as only two states (high and low resistive) are stored, and expensive ADCs/DACs are not required [6]. Despite worse latency and density compared to analog CIM, LIM effectively accelerates AI applications in memory [8]. Nevertheless, LIM still suffers from faults caused by immature technology [4], including both traditional and unique faults. Traditional faults are represented by conventional fault models prevalent in CMOS-based memories. Unique faults are emerging faults associated with memristive devices [9]. Despite their impact on individual memristors, faults might be acceptable for AI workloads if accuracy is not significantly reduced. This effect is well understood for analog computing [10] but not for LIM.

Contributions: We present the first thorough investigation of the impact of *traditional faults* on binary logic families. Thereby, we measure the resilience of binary logic families based on two introduced metrics. Furthermore, we propose *X-Fault*, an end-to-end mapping and simulation framework for binary neural networks (BNNs) using LIM in ReRAM.

II. BACKGROUND

A. Related Work

Previous work investigated non-idealities in memristive crossbars and their impact on CIM [11], [12]. Various simula-

Fig. 2: *X-Fault*'s end-to-end simulation workflow.

tion frameworks have been proposed to investigate the impact of faults on machine learning applications. Chakraborty et al. [13] presented a generalized approach to simulate neural networks on faulty memristive crossbars. The framework can simulate linear and non-linear non-idealities at architectural level. He et al. [14] proposed an end-to-end neural network tool that builds upon PyTorch. The tool takes into account the non-ideal effects of crossbars and adjusts mapping and training to drastically limit the impact of errors. The existing research has focused on analog-based CIM without exploiting binary logic families mapped to the memristive crossbar.

B. Binary Neural Networks (BNNs)

BNNs emerged as a promising low-power, low-cost, and reduced accuracy approach using aggressive quantization [15]. This method is particularly promising to deploy deep neural networks to resource-constrained devices, such as on the edge. The accuracy of BNNs is not on par with its less-quantized counterparts. This limitation is an unsolved challenge for today's complex datasets. However, simple classification tasks can achieve competitive performance. Table 2 in [15] reports up to 98.4% accuracy for the MNIST dataset using an MLP, but accuracy drops to 40% for Imagenet using VGG. Due to the low area and power cost of BNNs, an ecosystem of commercial tools recently emerged. For example, the Larq library [16] provides reference implementations and functions for training and deploying BNNs. In a BNN, the basic arithmetic scheme of convolutions in neural networks—the matrix-matrix-multiplication—is equivalent to an XNOR operation between two single-bit precision vectors. This arithmetic relation maps directly to LIM for memristive crossbars. Hence, a BNN is currently a preferred operation mode for neuromorphic devices using ReRAM in edge applications. While convolutional and dense layers can be represented as XNOR operations, and therefore executed on ReRAMs, this is not the case for other, non-binary operations. We decided to take a conservative approach, in which other operations (e.g., integer bit-count operations after each layer) are executed in CMOS in the hardware model.

C. LIM families on Memristive Crossbars

As stated in the introduction, memristive crossbars can be used in an analog or LIM fashion. We investigate LIM as it trades higher error resilience and reduced ADC/DAC overhead with lower power density and higher latency. In LIM, the logical state (0 or 1) is represented as a high or low resistive value programmed in a memristor. A set of memristors are operating

together for any logical operation to build a certain logic gate. An operation voltage is applied for a logical operation between two inputs. It is modulated by the state of the input memristors and applied to the output memristor. This voltage alters the state of the output memristor. Logic families have been classified into three categories: statefulness, proximity of computation, and flexibility [17]. Within the scope of this work, we focus on MAGIC [7] and IMPLY [18], which define basic logic operations. A full set of operations can be defined by daisy-chaining the basic ones. Fig. 3a illustrates the basic operations (OPs) supported by the logic families. To compute the inference of a BNN, we extend the basic OPs towards more sophisticated OPs, including the dominant XNOR operation.

III. SIMULATION METHODOLOGY

X-Fault's end-to-end workflow is presented in Fig. 2. The input to the flow is a user-defined BNN model in the Larq library [16] in Tensorflow. *X-Fault* iterates the BNN, maps XNOR-compatible operations to the memristive crossbar simulator, and executes the remainder of the model in native Tensorflow (representing CMOS logic next to a crossbar).

Mapping: *X-Fault*'s mapping tool brings convolutional and dense layers of a BNN to a crossbar of a given size. The mapper takes the kernel values from a BNN Tensorflow model and generates crossbar write, read, and logic instructions. It tracks if partial kernels fit on the remaining places to minimize write operations. We apply a linear mapping that iterates the BNN layerwise. The mapper writes all instructions as a binary file containing memory addresses of kernels and values, plus required logic operations.

Simulator: The simulator implements a memory controller that parses the given binary file from the mapping tool and provides the parameterized crossbars with the kernels and input values. The controller executes the logic operations of the logic families. The simulator tracks the resulting pulses applied to the respective bit and word lines for each crossbar access (read, write, and compute operation). The tracking information is used to trigger the currently active fault models.

Crossbar: The crossbar module implements a memristive crossbar array including a set of fault models. We focused on the conventional fault models, which are detailed in [9], [19]. The crossbar consists of memristors with binary states that connect the respective bit with word lines. Each memristor can be assigned with one of the following fault models:

Logic family	MAGIC		IMPLY	
Logic gates	NOR	NIMP	IMPLY	FALSE
		OR		
Extended gates	XOR	AND	XNOR	XOR
		NAND	NAND	OR
		XNOR	NOT	NOR

(a)

	IMPLY		MAGIC	
	# mem	# cycle	# mem	# cycle
AND	3	4	5	9
IMP	2	1	/	/
NAND	3	3	5	12
NOR	3	5	3	1
NOT	2	2	/	/
OR	3	3	3	1
XNOR	4	6	4	6
NIMP	/	/	3	1
XOR	4	5	3	3

(b)

Fig. 3: Simulation methodology: (a) overview of basic operations and the extended logic gates, and (b) number of memristive devices and required clock cycles of the implemented logic families.

- **Stuck-at-Fault (SAF)** is modeled as a constant resistive value of the memristor, which is either the high resistive state (HRS) or the low resistive state (LRS).
- **Read-Destructive-Fault (RDF)** flips the current state of the memristive cell and returns a correct value.
- **Deceptive Read Destructive Fault (DRDF)** alters the current state of the cell but returns the incorrect value.
- **Incorrect Read Fault (IRF)** does not change the cell state but returns an incorrect value.
- **Slow Write Fault (SWF)** does not successfully write the cell, and hence returns the unmodified value.

The fault models are randomly assigned to a certain percentage of all instantiated memristors, defined as the *injection rate*. To better understand the resilience of faults on the functional behavior of logic families, we introduce two new metrics. The *Quality of Logic (QoL)* is defined for a single fault model as

$$QoL = \sum_{i=0}^{G-1} \frac{\Lambda}{\Omega} \cdot 100\%, \quad (1)$$

where G is the number of gate types, Λ the total number of faulty outputs, and Ω the number of all outputs. QoL implies how well the entire set of supported logic gates performs when affected by a certain fault. The *Impact of Fault (IoF)* is defined for a single gate type as

$$IoF = \sum_{i=0}^{F-1} \frac{\Lambda}{\Omega} \cdot 100\%, \quad (2)$$

where F is the number of fault types. IoF indicates the impact of all faults on a single logic gate. These two metrics are calculated on the basis of the information provided in Table 3b, which shows the required number of memristors and the resulting cycle count of each logic gate.

IV. RESULTS AND DISCUSSION

In this section, we validate the simulation framework and investigate the robustness of two available logic families with respect to a sub-set of traditional faults. Furthermore, we investigate the fault impact on the accuracy of a BNN.

Table 4a and Table 4b illustrate the simulation results of logic gates versus the injected faulty behavior. The analysis

is based on the execution of single operations of all possible input values and initial values of all memristive devices. As an example, the NOT gate of the IMPLY logic family uses two distinct memristors (Table 3b). The memristors are initialized with the patterns $[(0,0), (1,0), (0,1), (1,1)]$, and the inputs 0 and 1 are supplied. The shown percentages represent the number of wrong outputs and are visualized with a heat map, which indicates that a reduced number of faults was propagated at logic level, affecting the functional behavior of the logic family. Note that coupling faults have not been considered in this first set of experiments because they require two or more consecutive accesses. Both QoL and IoF are calculated and shown in the last row and column of both tables, respectively. The OR gate performs the best in terms of resilience for both logic families, with an IoF of 31% (IMPLY) and 30% (MAGIC). The calculated QoL indicates that RDF has the least impact on logic gates, while the DRDF has a high impact. In addition, the worst logic gate in terms of fault resilience is the NOT (IMPLY) and the XOR gate (MAGIC). In general, *the experiment shows that both logic families perform equally well considering the similar QoL and IoF values*. However, architectural design decisions can be optimized based on the resilience towards certain fault models. For instance, the designer should favor the IMPLY implementation of the NAND gate with an IoF of 34% over the MAGIC implementation with an IoF of 45%.

Next, we perform a preliminary experiment with the full simulation framework to investigate the impact on a BNN during inference. We use a BNN model from the LARQ library examples [16] and train it with the MNIST dataset [20]. Within the scope of this work, only a preliminary simulation was performed, using a subset of the test data set for the inference. Each configuration has been executed twenty times to eliminate the effects resulting from the randomly placed errors based on the given injection rate. Fig. 4c and Fig. 4d illustrate the impact of the injected faults on the accuracy of the BNN. Independent of the chosen logic family, it can be observed that already *low injection rates significantly reduce the accuracy of the BNN*. Furthermore, the accuracy reaches a plateau at around 15%, independent of the logic family. We observe that the SWFset and SWFreset reduce the accuracy to roughly 20% for the MAGIC version, independent of the injection rate.

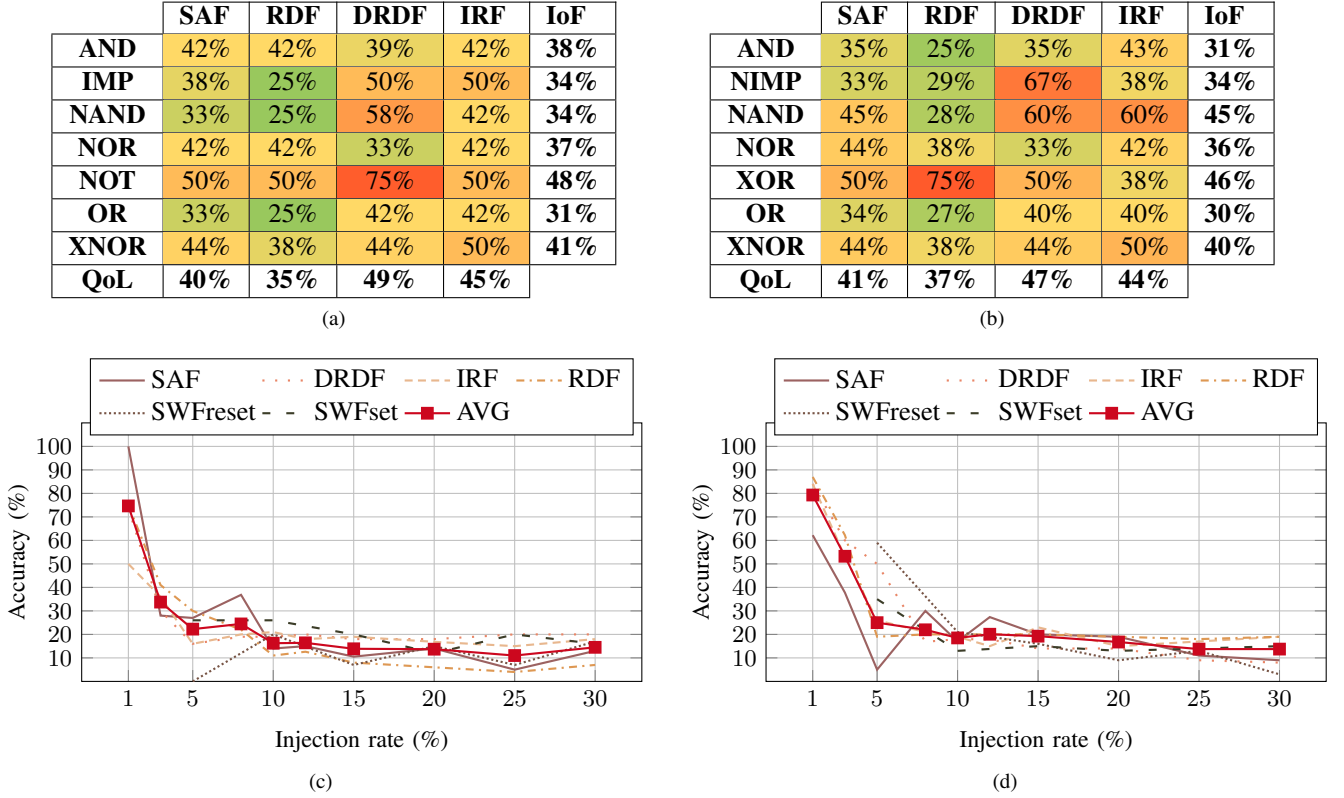


Fig. 4: Simulation results: Impact of faults on (a) the IMPLY logic family and (b) the MAGIC logic family. Effect of injection rate on the BNN inference accuracy for (c) IMPLY and (d) MAGIC.

V. CONCLUSION

This work investigated the impact of a sub-set of traditional faults on the logic-in-memory paradigm in general and on the accuracy of BNNs in particular. We developed an end-to-end simulation framework that maps the XNOR operations of an arbitrary Tensorflow model onto memristive crossbar arrays. In addition, the crossbar includes different faults randomly injected into the memristive devices. The results show that faults on memristive crossbar arrays significantly impact their functionality. Furthermore, a comparison of two logic families based on their fault resilience is facilitated by introducing two novel metrics. In future work, we plan to run extensive simulations with different models and data sets. In addition, we intend to expand the framework by including memristor-specific faults as well as the implementation of other logic families.

REFERENCES

- [1] F. Staudigl *et al.*, “A survey of neuromorphic computing-in-memory: Architectures, simulators and security,” *IEEE Design Test*, pp. 1–1, 2021.
- [2] J. M. Joseph *et al.*, *NEWROMAP: Mapping CNNs to NoC-Interconnected Self-Contained Data-Flow Accelerators for Edge-AI*. New York, NY, USA: Association for Computing Machinery, 2021, p. 15–20. [Online]. Available: <https://doi.org/10.1145/3479876.3481591>
- [3] A. Ankit *et al.*, “PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference,” in *ASPLOS*. IEEE, 2019.
- [4] P. Liu *et al.*, “Fault modeling and efficient testing of memristor-based memory,” *TCAS I*, vol. 68, no. 11, pp. 4444–4455, 2021.
- [5] F. Staudigl *et al.*, “NeuroHammer: Inducing bit-flips in memristive crossbar memories,” *arXiv preprint arXiv:2112.01087*, 2021.
- [6] P.-E. Gaillardon *et al.*, “The programmable logic-in-memory (PLiM) computer,” in *DATE*. IEEE, 2016.
- [7] S. Kvatinisky *et al.*, “MAGIC—Memristor-aided logic,” *TCAS II*, vol. 61, no. 11, pp. 895–899, 2014.
- [8] G. Papandroulidakis *et al.*, “Crossbar-based memristive logic-in-memory architecture,” *TNANO*, vol. 16, no. 3, pp. 491–501, 2017.
- [9] M. Fieback *et al.*, “Testing resistive memories: Where are we and what is missing?” in *2018 IEEE International Test Conference (ITC)*, 2018.
- [10] M. J. Rasch *et al.*, “A flexible and fast PyTorch toolkit for simulating training and inference on analog crossbar arrays,” in *AICAS*, 2021.
- [11] S. Kannan *et al.*, “Modeling, detection, and diagnosis of faults in multilevel memristor memories,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015.
- [12] Y.-X. Chen *et al.*, “Fault modeling and testing of 1T1r memristor memories,” in *2015 IEEE 33rd VLSI Test Symposium (VTS)*, 2015, pp. 1–6.
- [13] I. Chakraborty *et al.*, “Geniex: A generalized approach to emulating non-ideality in memristive xbars using neural networks,” in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.
- [14] Z. He *et al.*, “Noise injection adaption: End-to-end reram crossbar non-ideal effect adaption for neural network mapping,” in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.
- [15] H. Qin *et al.*, “Binary neural networks: A survey,” *Pattern Recognition*, vol. 105, p. 107281, 2020.
- [16] L. Geiger *et al.*, “Larq: An open-source library for training binarized neural networks,” *Journal of Open Source Software*, vol. 5, no. 45, p. 1746, Jan. 2020. [Online]. Available: <https://doi.org/10.21105/joss.01746>
- [17] J. Reuben *et al.*, “Memristive logic: A framework for evaluation and comparison,” in *2017 27th PATMOS*, 2017.
- [18] S. Kvatinisky *et al.*, “Memristor-based material implication (IMPLY) logic: Design principles and methodologies,” *TVLSI*, 2013.
- [19] P. Liu *et al.*, “Fault modeling and efficient testing of memristor-based memory,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 11, pp. 4444–4455, 2021.
- [20] Y. LeCun *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.