Fault-Tolerant Character Recognition in Neuromorphic Systems Using RRAM Crossbar Arrays

Fatemeh Shirinzadeh* Abhoy Kole* Kamalika Datta*‡
fatemeh.shirinzadeh@dfki.de abhoy.kole@dfki.de kdatta@uni-bremen.de

Saeideh Shirinzadeh*† Rolf Drechsler*‡
saeideh.shirinzadeh@dfki.de drechsler@uni-bremen.de

*German Research Centre for Artificial Intelligence (DFKI), Bremen, Germany
†Fraunhofer Institute for Systems and Innovation Research (ISI), Karlsruhe, Germany
‡Institute of Computer Science, University of Bremen, Germany

***Abstract*—Resistive Random-Access Memory (RRAM) crossbar arrays provide a high-density, low-power platform for neuromorphic computing. In this work, we implement an RRAM-based architecture for alphabet recognition using the EMNIST dataset, where all 26 English letters are represented as 28×28 binary images. Beyond ideal conditions, we study the impact of hardware imperfections, including stuck-at faults, random bit flips, and process variations, on recognition performance. To improve resilience, we evaluate two fault tolerance strategies: Triple Modular Redundancy (TMR) and Algorithm-Based Fault Tolerance (ABFT). TMR delivers strong reliability by masking faults through replication, while ABFT efficiently detects and corrects at a lower storage overhead, but at a higher computational cost. Our results demonstrate that RRAM crossbars combined with lightweight fault tolerance provide accurate, energy-efficient, and resilient neuromorphic computing, highlighting their promise for robust and efficient edge AI deployment.**

***Index Terms*—RRAM crossbar arrays, Neuromorphic computing, Fault tolerance, TMR, ABFT**

I. Introduction

Neuromorphic computing, inspired by the structure and function of biological neural systems, was introduced as a conceptual framework in [1]. It aims to develop *Very Large Scale Integration (VLSI)* architectures that emulate neural behavior. During the past few decades, traditional CMOS technology has been used predominantly to build neuromorphic systems [2], [3]. However, CMOS-based designs now face critical challenges as they approach physical scaling limits [4], [5].

To overcome these limitations, *Resistive Random Access Memory (RRAM)* devices have emerged as a promising alternative. RRAM offers high-density storage and analog computation capabilities, making it particularly suitable for *In-Memory Computing (IMC)* architectures [6], [7]. IMC aims to bypass the memory bottleneck inherent in von Neumann systems by enabling computation directly within memory arrays, thus eliminating the need to transfer data between memory and the CPU. The low power consumption, scalability, and fast switching characteristics of RRAM make it an excellent candidate to enable practical neuromorphic applications such as image and speech recognition.

RRAM crossbars naturally support parallel *Multiply-And-Accumulate (MAC)* operations by summing currents along memory columns, where device conductances are tuned to represent weights. This enables efficient matrix-vector multiplication, a core operation in neuromorphic processing [8]–[10].

Although RRAM-based crossbar architectures have shown significant potential, their adoption in real-world systems is hindered by several reliability concerns. Device-level imperfections, such as stuck-at faults, random bit flips, and process-induced variations, can significantly degrade system performance. Therefore, achieving reliable operation under these non-ideal conditions requires effective fault tolerance mechanisms.

Fault tolerance has been a central theme in the design of digital systems since the advent of computing [11]. Traditional approaches rely on techniques such as redundancy, error detection and correction, and voting mechanisms to ensure correct functionality in the presence of faults. In emerging hardware paradigms like RRAM, however, such techniques must be reimagined to suit analog characteristics and high integration densities [12]–[14].

In this work, we explore the use of RRAM crossbar arrays for a pattern recognition task, specifically, the classification of alphabet characters. For this purpose, we utilize the EMNIST dataset [15], an extended version of the well-known MNIST database, in which each character image is represented as a 28×28 pixel image.

We investigate the impact of several hardware faults and evaluate the system's robustness under degraded conditions. To improve fault tolerance, we study two complementary redundancy-based approaches. The first is *Triple Modular Redundancy (TMR)*, which increases the memory footprint by a factor of three. TMR stores each data bit in triplicate and employs majority voting to mask errors, providing strong resilience at the cost of significant area overhead. The second is an algorithm-based strategy, *Algorithm-Based Fault Tolerance (ABFT)*, which introduces lightweight checksum constraints that enable fault detection and, in some cases, correction without requiring full replication. ABFT leverages the structure of computations to identify inconsistencies, offering reduced redundancy overhead compared to TMR while maintaining competitive accuracy in the presence of faults.

While prior works have demonstrated RRAM-based recognition systems, most of them focus on functionality and performance under ideal conditions, without systematically analyzing the impact of hardware faults or evaluating fault-tolerance mechanisms [16]–[18]. In contrast, our work explicitly models multiple fault types and provides a comparative study of two complementary fault-tolerance approaches, TMR and ABFT. To the best of our knowledge, this is the first work that quantitatively compares hardware-oriented and algorithm-oriented fault tolerance in RRAM

crossbars for character recognition. This contribution addresses an important gap between proof-of-concept crossbar demonstrations and practical, fault-resilient neuromorphic architectures.

These complementary approaches demonstrate a trade-off between fault coverage, hardware cost, and energy efficiency: TMR offers strong resilience at high hardware and power overhead, while ABFT provides lightweight error detection with lower cost but limited correction ability and additional time overhead [14], [19], [20].

The rest of the paper is organized as follows: Section II provides the preliminary concepts necessary for understanding the proposed approach. Section III details the methodology and implementation of the RRAM-based recognition system. Section IV presents the experimental results and comparisons. Finally, Section V concludes the paper.

II. Background

A. RRAM Crossbar Array

RRAM is an emerging non-volatile memory technology built on a simple metal–oxide–metal structure [21], [22]. By applying a voltage of appropriate magnitude and polarity, the resistance of the device can be modulated between a Low Resistance State *(LRS)*, representing logic 1, and a High Resistance State *(HRS)*, representing logic 0. RRAM devices are commonly arranged in a crossbar architecture, where each memory cell is located at the intersection of perpendicular nanowires, known as *bitlines* (horizontal) and *wordlines* (vertical).

Programming a device into the HRS involves applying $V/2$ to the bitline and $-V/2$ to the wordline, while switching to the LRS requires $-V/2$ on the bitline and $V/2$ on the wordline. The binary interpretation of the resistance states and the voltage biasing scheme enables RRAM crossbars to serve not only as dense memory arrays but also as computational elements. These structures have demonstrated success in both digital and analog in-memory computing applications [10], [23]–[25].

B. MAC Operation

RRAM's unique analog computation capabilities distinguish it from other emerging memory technologies, particularly in neural network implementations. These capabilities enable efficient MAC operations, crucial for matrix-vector multiplication.

The RRAM devices within the crossbar structure (Fig. 1) are initialized in a resistive state of $a_{j,k}^{-1}$. Here, $a_{j,k}$ represents the conductance of the device at the intersection of row j and column k. By applying voltages $x_1, \ldots, x_n$ to the rows, up to m MAC operations can be performed simultaneously across m crossbar columns.

The outputs are the currents flowing in the corresponding crossbar columns, which is the sum of currents in each RRAM device, i.e. $i_j = \sum_{k=1}^{n} x_k \cdot a_{j,k}$. This can be denoted as $I = XA$, where $I = (i_1, \ldots, i_m)$, $X = (x_1, \ldots, x_n)$ and the matrix A is defined as follows:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \ldots & a_{1,m} \\ a_{2,1} & a_{2,2} & \ldots & \ldots & a_{2,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \ldots & a_{n,m} \end{bmatrix} \quad (1)$$

Thus, m MAC operations, each involving n multiplications, can be computed in parallel within a single cycle.

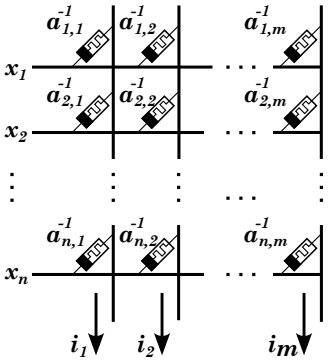

Fig. 1: MAC computation in RRAM crossbar

C. Related Work

Several works have investigated memristor crossbar architectures for neuromorphic pattern recognition, primarily focusing on image and character recognition. Le et al. [16] proposed a memristive crossbar-based character recognition system using a single crossbar array with bipolar inputs to perform XNOR operations between the input vector and stored 8×8 character patterns. Their approach showed high accuracy and tolerance to memristance variation but did not investigate the impact of faults or process variations.

In [17], a single memristor crossbar circuit with bipolar inputs was proposed for neuromorphic image recognition. This design implements a simplified XNOR function to measure similarity between input and stored patterns while reducing the number of memristors by 50% compared to complementary and twin architectures. It also achieves lower power consumption and improved fault tolerance, maintaining higher recognition accuracy even with 10% defective devices. However, the effects of random faults or process variation beyond fixed defect rates were not explored.

Another work [18] introduced a twin crossbar architecture using two identical $M+$ memristor arrays instead of the traditional complementary $M + /M$ setup. By applying Discrete Cosine Transform, this design reduces the number of low-resistance state cells, significantly lowering power consumption. Depending on the degree of coefficient reduction, power savings of up to 77.4% were achieved compared to previous complementary designs. This study focused on power optimization and did not address the impact of faults on recognition accuracy.

Finally, Truong et al. [26] proposed a binary memristor crossbar for neuromorphic speech recognition, demonstrating strong robustness to memristance variation in filamentary-switching devices. Although their focus was on speech signals and fabrication advantages, their results highlight the potential of binary memristor arrays for reliable pattern recognition under variability.

Our work builds on these approaches by explicitly modeling faults, noise, and process variations in RRAM-based character recognition, evaluating system reliability under realistic computing conditions.

III. RRAM-Based Character Recognition

In this section, we discuss our proposed RRAM-based character recognition methodology, hardware fault modeling, and the fault-tolerance strategies based on hardware and algorithmic redundancy.

A. Proposed Recognition Method

As the initial step of our methodology, we use 26 binary black-and-white images from the EMNIST dataset [15], each

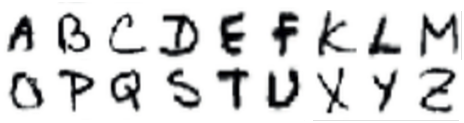

Fig. 2: Representative samples of binary images from the EMNIST dataset [15]

representing a capital English letter ('A' to 'Z') with a resolution of 28×28 pixels. These images serve as the input patterns for character recognition. Each image is then flattened into a 1×784 vector, forming the corresponding input pattern vector. Representative examples of these input images are illustrated in Fig. 2.

These vectors are then stored column-wise in the RRAM crossbar array. Consequently, the crossbar is configured with 784 rows and 26 columns, resulting in a matrix of size 784×26. We refer to this matrix as the weighted memory array $\mathbf{A}$. In this representation, each column of $\mathbf{A}$ corresponds to one character: the first column to "A," the second to "B," and so on, with the 26th column representing "Z". In the kth column, the RRAM cell at the ith row is programmed to a resistive value of either LRS or HRS, depending on whether the corresponding pattern bit is 1 or 0, respectively.

For character recognition applications, RRAM crossbar arrays perform the bitwise Exclusive-NOR *(XNOR)* operation between an input pattern and the pre-stored reference patterns within the array. This operation effectively measures the similarity between the input and the stored patterns, as expressed in Eq. (2) [17]:

$$\begin{aligned} \overline{X \oplus A} &= XA + X'(1-A) \\ &= XA - X'A + X' \\ &= (X - X')A + X' \end{aligned} \tag{2}$$

Here, X denotes the input vector of size 1×784, X' represents its bitwise complement, and $\mathbf{A}$ refers to the memory array matrix containing the stored patterns. Since X' does not directly interact with $\mathbf{A}$ during the core computation, its contribution can be separated, allowing for simplification of the matrix operation between X and $\mathbf{A}$. By defining $\mathbf{I} = (X - X')$, Eq. (2) can be reformulated as:

$$Y = \overline{X \oplus A} = \mathbf{IA} \tag{3}$$

Assuming the input vector $X = [x_1, x_2, \ldots, x_n]$ has a dimension of $1 \times n$, the resulting vector $\mathbf{I}$ also has the dimension $1 \times n$, as shown in Eq. (4):

$$\mathbf{I} = (X - X') = [i_1 \quad i_2 \quad \ldots \quad i_n] \tag{4}$$

Each element i_k in $\mathbf{I}$ is a bipolar value, which can be $+1$ or -1 depending on the corresponding bit in X. For example, if $X = [0, 0, 1]$, then $\mathbf{I} = [-1, -1, 1]$. Substituting Eq. (1) and Eq. (4) into Eq. (3), the output vector Y is obtained as follows:

$$\begin{aligned} Y = [i_1 \quad i_2 \quad \ldots \quad i_n] \cdot & \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,1} & \cdots & a_{n,m} \end{bmatrix} \\ &= [y_1 \quad y_2 \quad \ldots \quad y_m] \end{aligned} \tag{5}$$

Each output element y_j is obtained through a MAC operation between the vector $\mathbf{I}$ and the jth column of the memory matrix $\mathbf{A}$ as shown in Eq. (6). Thus, the entire output vector $\mathbf{Y}$ can be computed in a single MAC cycle across the crossbar array.

$$y_j = \sum_{k=1}^{n} i_k \cdot a_{k,j} \tag{6}$$

The resulting vector Y, derived from the XNOR-based similarity comparison between the input X and the stored patterns in $\mathbf{A}$, quantifies the degree of match between the input and each stored pattern. A *Winner-Take-All (WTA)* mechanism as described in [16], [26] is then applied to identify the index j at which y_j is maximized, thereby indicating the stored pattern that most closely matches the input.

B. Modeling Hardware Faults

To evaluate the robustness of our RRAM-based character recognition system under non-ideal conditions, we incorporate three common sources of hardware-level imperfections: *stuck-at faults*, *random bit flips*, and *memristance variation*.

Stuck-at faults are a well-known defect in resistive memory technologies, in which certain memory cells become permanently fixed at either a logic high (LRS) or logic low (HRS), regardless of the intended value. To evaluate their impact, we inject randomly distributed stuck-at faults into the memory cells of the crossbar array, represented by the stored matrix $\mathbf{A}$. The *fault rate* is defined as the ratio of faulty cells to the total number of memory cells. For each defective cell, a stuck-at value of 0 or 1 is assigned uniformly at random.

Random bit flips simulate soft errors and transient noise that can occur during inference. In this fault model, individual bits in the input vector $\mathbf{X}$ or in the memory cells represented by the stored matrix $\mathbf{A}$ can randomly flip from 0 to 1 or from 1 to 0, according to a defined *bit flip probability*. These bit flips represent dynamic and non-permanent disturbances that may result from thermal noise, radiation, or circuit instability.

Memristance variation reflects the analog variability in RRAM devices caused by manufacturing imperfections and environmental conditions [27]–[31]. These variations affect the resistance levels of the LRS and HRS states, known as memristance values, and can degrade the accuracy of analog computations. To capture this effect, the memristance values are modeled using a Gaussian (normal) distribution centered around the nominal LRS/HRS values, with a specified standard deviation representing the level of variation.

All three types of non-idealities are introduced exclusively during the *inference phase*, allowing us to evaluate the post-deployment resilience of the system without modifying the training process.

C. Redundancy Technique for Fault Tolerance Design

In fault-tolerant design, redundancy provides the additional information required to mitigate the impact of failures. *Hardware redundancy* is the most widely adopted approach and can be implemented in several forms. One of the most prominent is *static redundancy*, which provides fault tolerance without the explicit detection of errors. Within this category, two fundamental techniques are *Duplication with Comparison* and *Triple Modular Redundancy (TMR)* [32].

Hardware redundancy, however, comes with substantial trade-offs, most notably increased area, power consumption, weight, and cost. For example, the duplication-with-comparison method requires duplicating every functional unit, resulting in a 100%

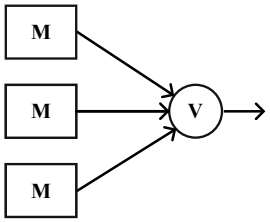

Fig. 3: Triple modular redundancy diagram [33]

hardware overhead, which is often prohibitive in resource-constrained environments. TMR imposes an even greater overhead of 200% [33], [34], as each module must be triplicated. This very high hardware overhead is a significant drawback of redundancy-based techniques. Nonetheless, TMR remains attractive in safety-critical systems due to its robustness: it can tolerate a single module failure without disrupting normal operation. This makes it particularly valuable in applications where *system availability* and *high reliability* are of paramount importance.

In this work, we employ TMR as one of our fault-tolerance strategies. As shown in Fig. 3, the architecture consists of three identical modules (denoted **M**), each executing the same computation independently. Their outputs are forwarded to a *majority voter* (denoted **V**), which selects the majority value as the final output. Since the outputs are binary and the number of modules is odd, the majority vote guarantees an unambiguous and fault-tolerant result, even in the presence of a faulty module [35], [36]. This mechanism enables effective masking of both *transient* and *permanent faults*, thereby enhancing system reliability despite the overhead.

D. Algorithm-Based Fault Tolerance (ABFT)

In addition to hardware redundancy techniques such as TMR, we also investigate an ABFT scheme as a lightweight alternative [37]. ABFT introduces checksum-based redundancy directly at the data level of the RRAM crossbar computation, enabling detection and correction of faulty memory cells with significantly reduced overhead compared to TMR.

For an $m \times n$ memory array, ABFT augments the matrix with one checksum row and one checksum column, producing an $(m+1) \times (n+1)$ encoded array. Each element of the checksum row stores the sum of the corresponding column entries, while each element of the checksum column stores the sum of the respective row entries. The additional corner cell at position $(m+1, n+1)$ holds the checksum of all row or column checksums, ensuring global consistency.

During inference, the row and column checksums are recomputed from the received data and compared against the stored checksums. A mismatch in row i indicates at least one error in that row, while a mismatch in column j indicates a fault in that column. By intersecting these conditions, a faulty element can often be precisely localized at position (i, j). Once localized, the correct value of the cell can be reconstructed using either the row checksum or column checksum equation:

$$a_{i,j} = \text{RowChecksum}(i) - \sum_{k \neq j} a_{i,k} \tag{7}$$

or symmetrically using the column relation. This enables single-cell fault correction.

In comparison to hardware redundancy, ABFT has a much lower storage overhead. By expanding from m and n to $(m+1)$ and similar $n+1$ elements, the overhead ratio is given by:

$$\frac{(m+1)(n+1) - mn}{mn} = \frac{m+n+1}{mn} \tag{8}$$

For the 784×26 EMNIST-based memory array, the overhead is approximately 4%, in contrast to the 200% overhead of TMR. This makes ABFT attractive in scenarios where area and power efficiency are critical.

IV. Experimental Results

This section presents simulation results evaluating the performance and fault tolerance of the proposed RRAM-based character recognition system under various hardware-level imperfections, including permanent faults and memristance variations. All experiments were carried out on a machine equipped with an Intel® Core™ i7 2.10 GHz processor and 8GB of main memory.

A. Performance in Ideal Environment

The RRAM-based crossbar array used in our character recognition system was modeled and simulated in Python, based on a simplified behavioral abstraction of the Verilog-A model developed by the Stanford Nanoelectronics Research Group [38]. Each RRAM device in the crossbar operates in two distinct resistance states: an HRS of $1\ \text{M}\Omega$ representing binary '0', and an LRS of $10\ \text{k}\Omega$ representing binary '1'.

The character inputs, which are binary images of size 28×28, are flattened into 784-element vectors according to Eq. (4). This transformation yields bipolar vectors, where logical '1' and '0' are mapped to voltage levels of $+1\,\text{V}$ and $-1\,\text{V}$, respectively. These vectors are then applied to the word lines of the crossbar array.

By applying these bipolar inputs, the crossbar performs analog current summation at the bit lines, effectively carrying out a MAC operation. The resulting output currents form distinctive profiles that are used for character recognition. This computation method not only exploits the analog nature of RRAM but also leverages its inherent parallelism for efficient processing.

Fig. 4(a) illustrates the distribution of output currents across the first ten columns of the crossbar array when the input character "A" is applied, as conceptually depicted in Fig. 1. Among the ten columns of the reference matrix $\mathbf{A}$, the highest output current, i_1, is observed in the first column. This indicates the strongest correlation between the input pattern and the reference pattern stored in column one, confirming correct recognition of character "A".

Recognition results for other characters, including "B", "C", and "D", are also shown in Fig. 4(b)–(d). In each case, the highest current is observed in the expected column, demonstrating the effectiveness and accuracy of our RRAM-based system in distinguishing between different input patterns.

These results validate the suitability of RRAM crossbar arrays for low-power, parallel, and analog computation tasks, particularly suitable in edge AI applications where memory and logic integration are critical [39].

B. Impact of Hardware Faults

To assess fault tolerance, we randomly inject defects into the RRAM crossbar matrix by assuming a fault rate e, which defines the percentage of affected cells. Each faulty cell is randomly assigned one of three fault types, stuck-at-0, stuck-at-1, or random

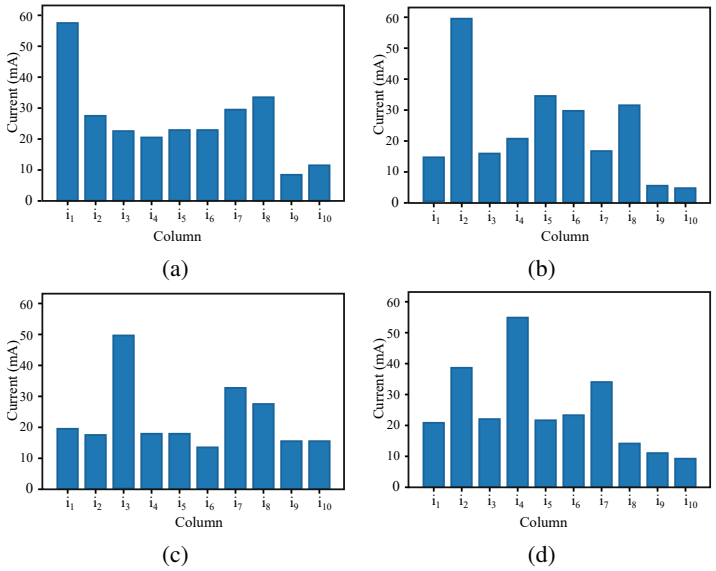

Fig. 4: Output currents of the first 10 columns for: (a) Character ”A” (b) Character ”B” (c) Character ”C” (d) Character ”D”

bit flip, with equal probability. For each fault level, we run 100 independent trials and report average recognition accuracy. For visualization, Fig. 5 shows per-letter accuracy for a representative subset (A–J); the remaining letters exhibit the same monotonic trend with varying sensitivity.

As expected, increasing the fault rate leads to a gradual degradation in classification performance. Within the displayed subset (A–J), letters such as “F”, “I”, and “J” show stronger sensitivity. Averaged over the entire 26-character dataset, the non-redundant baseline system achieves 89.8% accuracy at a 50% fault rate, highlighting both its resilience and the need for additional fault-tolerance mechanisms.

To improve robustness against hardware faults, we implement a TMR scheme within the RRAM crossbar architecture. In TMR each character is stored in three separate columns of the crossbar array. During inference, the MAC operation is performed on all three redundant copies, and the final recognition result is obtained via majority voting. This ensures that even if one copy is corrupted by faults, the remaining two can recover the correct output, thereby enhancing overall reliability. To evaluate fault

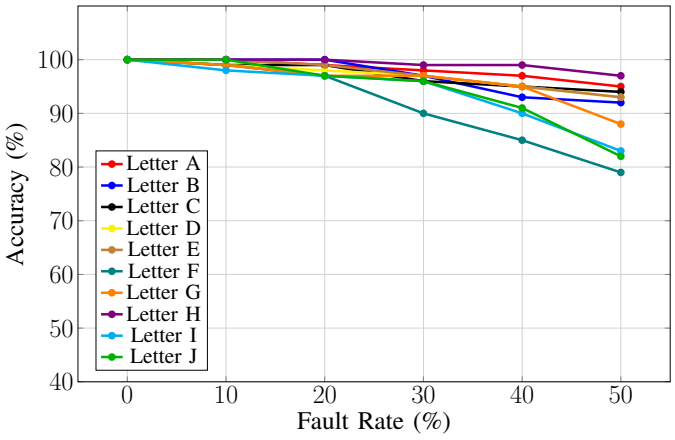

Fig. 5: Per-letter recognition accuracy with the baseline system for the representative subset (A–J)

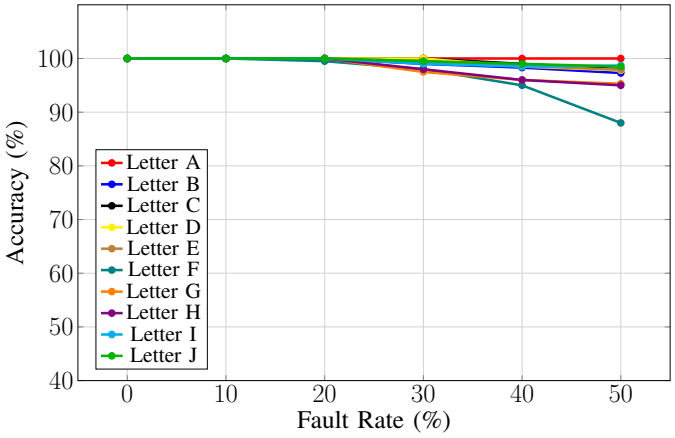

Fig. 6: Per-letter recognition accuracy with TMR for the representative subset (A–J)

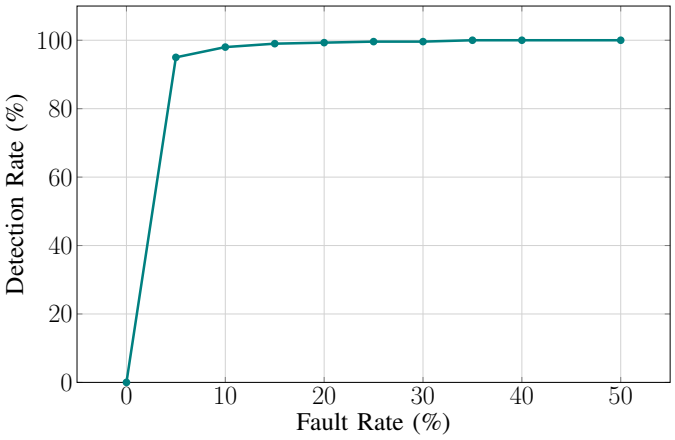

Fig. 7: Average ABFT detection rate as a function of increasing fault rate

tolerance, we repeated the fault-injection experiments described earlier, using the same fault rates and types across the replicated memory blocks. This guarantees that all blocks are uniformly impacted by faults, simulating a realistic scenario where no region is completely fault-free. As shown in Fig. 6, the TMR design consistently outperforms the non-redundant baseline.

To evaluate the effectiveness of ABFT, we repeated the same fault-injection experiments described in Section III-B, this time using the checksum-based redundancy scheme. For each fault rate, 100 Monte Carlo trials were conducted to obtain average recognition accuracy, fault detection, and correction rates. The results indicate that ABFT achieves consistently high fault-detection coverage. Beyond a 5% fault rate, more than 96% of injected faults are detected, and detection exceeds 99% once the fault rate reaches 15%, as shown in Fig. 7.

The overall comparison across the three systems is summarized in Fig. 8. For TMR, averaged over the full 26-character dataset, recognition accuracy remains at 96.8% even under a 50% fault rate, compared to 89.8% without redundancy. ABFT also maintains robust performance, achieving 90.2% accuracy at the same fault rate. At fault rates below 20%, all systems achieve nearly perfect accuracy, confirming that TMR and ABFT effectively mitigate the impact of hardware faults in RRAM-based computing systems. In contrast to TMR’s straightforward replication, ABFT not only detects errors but can also *correct* certain faults: when a single

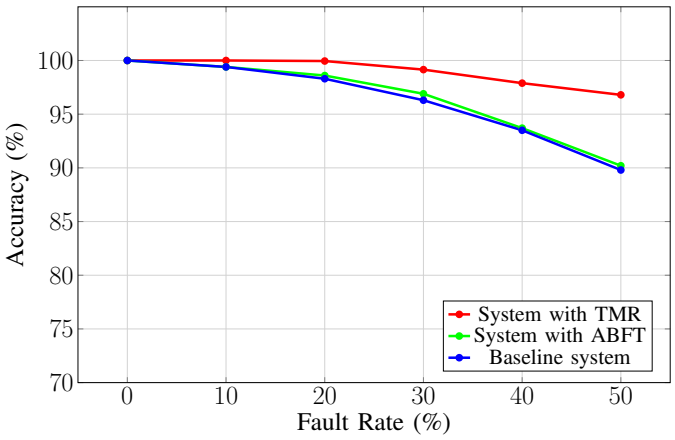

Fig. 8: Average recognition accuracy vs. fault rate for Baseline, TMR, and ABFT systems

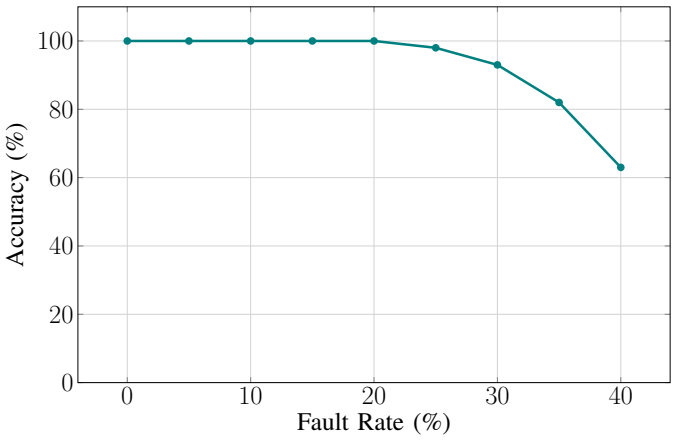

Fig. 9: Impact of memristance variation on recognition accuracy

memory cell within a row or column is faulty, checksum constraints enable accurate localization and recovery of the correct value. Consequently, ABFT maintains recognition accuracy close to the baseline, while providing fault detection and selective correction with only modest storage overhead. The trade-off, however, lies in computation time, as checksum calculations introduce additional latency compared to TMR's simpler replication mechanism.

Overall, the complementary strengths of TMR and ABFT indicate their suitability for different deployment scenarios: TMR is preferred when efficiency and low latency are critical, while ABFT is beneficial when fault diagnosis and correction are vital design requirements.

C. Impact of Memristance Variation

In addition to analyzing permanent faults, we investigate the impact of device-to-device variability in memristance, a common and significant source of uncertainty in RRAM-based systems arising from process variations and fabrication imperfections. This variability is modeled by introducing random multiplicative noise to the conductance values of memristive elements within the crossbar array.

Specifically, the conductance G of each memristor is perturbed by a Gaussian-distributed random variable with mean 1 and standard deviation σ, yielding:

$$G_{\text{noisy}} = G \times \mathcal{N}(1, \sigma) \tag{9}$$

where $\mathcal{N}(1, \sigma)$ represents a normal distribution with mean 1 and standard deviation σ [16]. This multiplicative noise model effectively captures relative variations in conductance, thereby reflecting realistic device-level process-induced fluctuations. The variability parameter σ is varied from 0 to 0.4, corresponding to up to 40% standard deviation from the nominal conductance values.

As σ increases, both HRS ($1\,\text{M}\Omega$) and LRS ($10\,\text{k}\Omega$) exhibit proportional fluctuations, introducing uncertainty in readout operations. For each variation level, we perform 100 independent Monte Carlo trials to mitigate stochastic effects and compute the average recognition accuracy. The impact of memristance variation on system performance is illustrated in Fig. 9.

The results show that the proposed single-crossbar architecture maintains high recognition accuracy under moderate variation. For instance, recognition rates remain at 100% and 99.1% for variation levels of 20% and 25%, respectively. This robustness is primarily attributed to the large HRS/LRS ratio of 100, which facilitates clear separation between logic states even in the presence of variability.

However, as variation increases further, performance deteriorates significantly. At $\sigma = 0.4$ (i.e., 40% standard deviation), the recognition accuracy drops to 66%. This degradation results from a broader spread in resistance values, particularly when considering the $\pm 3\sigma$ range that covers 99.7% of the Gaussian distribution. Under such high variability, the overlap between the resistance distributions of HRS and LRS increases, thereby reducing the system's ability to reliably distinguish between logic states.

V. Conclusion

We presented an RRAM-based architecture for alphabet recognition using the EMNIST dataset, representing all 26 English letters as 28×28 binary images. The proposed crossbar system achieved high accuracy under ideal conditions, confirming RRAM's suitability for large-scale, parallel in-memory computation.

To evaluate robustness, we introduced hardware fault models including stuck-at faults, bit flips, and memristance variations. Our experiments showed that these non-idealities degrade recognition accuracy, and we investigated two complementary fault-tolerance strategies: Triple Modular Redundancy (TMR) and Algorithm-Based Fault Tolerance (ABFT). Both approaches demonstrated strong resilience, highlighting trade-offs between hardware cost, energy efficiency, and fault tolerance.

To the best of our knowledge, this work provides the first comparative evaluation of hardware-oriented (TMR) and algorithm-oriented (ABFT) fault tolerance in RRAM crossbars for character recognition under realistic fault models. The findings demonstrate that RRAM crossbars equipped with redundancy schemes enable accurate, fault-tolerant, and energy-efficient neuromorphic computing, offering a scalable path toward robust edge AI systems. Future work will extend this framework to device-level circuit validation and integration with more advanced neuromorphic architectures.

Acknowledgement

This work is partly supported by the German Research Foundation (DFG) within the Project PLiM (DR 287/35-1, DR 287/35-2 and SH 1917/1-2) and by the German Ministry for Research, Technology and Space (BMFTR) with project ExaVerse (grant number 01IW25003).

REFERENCES

[1] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[2] G. Indiveri, E. Chicca, and R. Douglas, "A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 211–221, 2006.

[3] C. Mayr, J. Partzsch, M. Noack, S. Hänzsche, S. Scholze, S. Höppner, G. Ellguth, and R. Schüffny, "A biological-realtime neuromorphic system in 28 nm cmos using low-leakage switched capacitor circuits," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 1, pp. 243–254, 2016.

[4] Y. Taur, "Cmos design near the limit of scaling," *IBM Journal of Research and Development*, vol. 46, no. 2.3, pp. 213–222, 2002.

[5] T. Pešić-Branin and B. L. Dokić, "Strained silicon layer in cmos technology," *Elektronika ETF*, vol. 18, no. 2, pp. 63–69, 2014.

[6] J. Meng, I. Yeo, W. Shim, L. Yang, D. Fan, S. Yu, and J.-S. Seo, "Sparse and robust rram-based efficient in-memory computing for dnn inference," in *2022 IEEE International Reliability Physics Symposium (IRPS)*, 2022, pp. 3C.1–1–3C.1–6.

[7] S. Yu, W. Shim, X. Peng, and Y. Luo, "Rram for compute-in-memory: From inference to training," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 7, pp. 2753–2765, 2021.

[8] M. Prezioso *et al.*, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[9] O. Krestinskaya, K. N. Salama, and A. P. James, "Learning in memristive neural network architectures using analog backpropagation circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 719–732, 2018.

[10] L. Xia *et al.*, "Technological exploration of RRAM crossbar array for matrix-vector multiplication," *Journal of Computer Science and Technology*, vol. 31, no. 1, pp. 3–19, Jan 2016.

[11] A. Avizienis, "Fault-tolerance: The survival attribute of digital systems," *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1109–1125, 1978.

[12] A. Gebregiorgis, A. Zografou, and S. Hamdioui, "Rram crossbar-based fault-tolerant binary neural networks (bnns)," in *2022 IEEE European Test Symposium (ETS)*, 2022, pp. 1–2.

[13] R. Schroedter, A. Haidar, and R. Tetzlaff, "Fault impact map for memristive crossbar neural networks," in *2024 13th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, 2024, pp. 01–04.

[14] M. Liu, L. Xia, Y. Wang, and K. Chakrabarty, "Fault tolerance for rram-based matrix operations," in *2018 IEEE International Test Conference (ITC)*, 2018, pp. 1–10.

[15] G. Cohen, S. Afshar, J. Tapson, and A. V. Schaik, "Emnist: Extending mnist to handwritten letters," *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017.

[16] T. A. Le and D. Truong, "Neuromorphic character recognition using the single memristor crossbar array." IEEE, 2021, pp. 1–4.

[17] S. N. Truong, "Single crossbar array of memristors with bipolar inputs for neuromorphic image recognition," *IEEE Access*, vol. 8, pp. 69 327–69 332, 2020.

[18] S. N. Truong, S. Shin, S.-D. Byeon, J. Song, and K.-S. Min, "New twin crossbar architecture of binary memristors for low-power image recognition with discrete cosine transform," *IEEE Transactions on Nanotechnology*, vol. 14, no. 6, pp. 1104–1111, 2015.

[19] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962.

[20] A. Cavelan and F. M. Ciorba, "Algorithm-based fault tolerance for parallel stencil computations," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, 2019, pp. 1–11.

[21] L. Chua, "Memristor – the missing circuit element," *IEEE Trans. on Circuit Theory*, vol. CT-18, no. 5, pp. 507–519, 1971.

[22] D. Strukov, G.S.Snider, D. Stewart, and R. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, 2008.

[23] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (imply) logic: Design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, 2014.

[24] N. Talati, S. D. Gupta, P. S. Mane, and S. Kvatinsky, "Logic design within memristive memories using memristor-aided logic (MAGIC)," *IEEE Trans. on Nanotechnology*, vol. 15, pp. 635–650, 2016.

[25] G. Papandroulidakis, I. Vourkas, A. Abusleme, G. C. Sirakoulis, and A. Rubio, "Crossbar-based memristive logic-in-memory architecture," *IEEE Transactions on Nanotechnology*, vol. 16, no. 3, pp. 491–501, 2017.

[26] S. N. Truong, S.-J. Ham, and K.-S. Min, "Neuromorphic crossbar circuit with nanoscale filamentary-switching binary memristors for speech recognition," *Nanoscale research letters*, vol. 9, pp. 1–9, 2014.

[27] D. Niu, Y. Chen, C. Xu, and Y. Xie, "Impact of process variations on emerging memristor," in *Design Automation Conference*, 2010, pp. 877–882.

[28] M. Hu, H. Li, Y. Chen, X. Wang, and R. E. Pino, "Geometry variations analysis of tio2 thin-film and spintronic memristors," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, 2011, pp. 25–30.

[29] J. Rajendran, R. Karri, and G. S. Rose, "Improving tolerance to variations in memristor-based applications using parallel memristors," *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 733–746, 2015.

[30] J. Rajendran, H. Maenm, R. Karri, and G. S. Rose, "An approach to tolerate process related variations in memristor-based applications," in *2011 24th Internatioal Conference on VLSI Design*, 2011, pp. 18–23.

[31] J. Reuben, M. Biglari, and D. Fey, "Incorporating variability of resistive ram in circuit simulations using the stanford–pku model," *IEEE Transactions on Nanotechnology*, vol. 19, pp. 508–518, 2020.

[32] W. Toy, "Fault-tolerant design of local ess processors," *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1126–1145, 1978.

[33] V. S. Veeravalli, "Fault tolerance for arithmetic and logic unit," in *IEEE Southeastcon 2009*, 2009, pp. 329–334.

[34] M. Kubo and S. Chou, "Fault tolerant techniques for memory components," in *1985 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, vol. XXVIII, 1985, pp. 230–231.

[35] J. von Neumann, "Probabilistic logics," in *Automata Studies*, C. Shannon and J. McCarthy, Eds. Princeton, NJ: Princeton University Press, 1956, pp. 43–98.

[36] D. P. Siewiorek, C. G. Bell, and A. Newell, *Computer Structures: Principles and Examples*. New York: McGraw-Hill, 1982.

[37] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Transactions on Computers*, vol. C-33, no. 6, pp. 518–528, 1984.

[38] Z. Jiang and H.-S. P. Wong, "Stanford university resistive-switching random access memory (rram) verilog-a model," Oct 2014. [Online]. Available: https://nanohub.org/publications/19/1

[39] X. Si, Y. Zhou, J. Yang, and M.-F. Chang, "Challenge and trend of sram based computation-in-memory circuits for ai edge devices," in *2021 IEEE 14th International Conference on ASIC (ASICON)*, 2021, pp. 1–4.