Structural Testing of a RRAM-based AI Accelerator Core

Serlis, Emmanouil Anastasios; Xun, Hanzhi; Taouil, Mottaqiallah; Hamdioui, Said; Fieback, Moritz

DOI
10.1109/ETS63895.2025.11049610

Publication date
2025

Document Version
Final published version

Published in
Proceedings of the 2025 IEEE European Test Symposium (ETS)

Structural Testing of a RRAM-based AI Accelerator Core

Emmanouil Anastasios Serlis[1], Hanzhi Xun[1], Mottaqiallah Taouil[1,2], Said Hamdioui[1,2], Moritz Fieback[1]
[1]Computer Engineering Laboratory, Delft University of Technology, Mekelweg 4, 2628CD, Delft, The Netherlands
[2]CognitiveIC, Van der Burghweg 1, 2628CS, Delft, The Netherlands
Email: {e.a.serlis, h.xun, m.taouil, s.hamdioui, m.c.r.fieback}@tudelft.nl

2025 IEEE European Test Symposium (ETS) | 979-8-3315-9450-3/25/$31.00 ©2025 IEEE | DOI: 10.1109/ETS63895.2025.11049610

***Abstract*—Edge AI accelerators have revolutionized intelligent information processing, enabling applications, such as self-driving cars and low-power IoT devices. Design efforts prioritize computational power and energy efficiency. Nevertheless, testability is also critical for in-field, reliable operation, especially for novel architectures such as memristive, analog Computation-in-Memory (CIM) cores. These structures combine emerging Resistive Random Access Memory (RRAM) with CMOS peripherals to efficiently implement vector-matrix-multiplication (VMM) operations for inference. Current research on AI Accelerator testing relies on functional test patterns, derived from abstract and unrealistic fault models. This paper presents a novel structural testing methodology for CIM VMM circuits. The methodology utilizes device-level defect models and defines new fault models for CIM VMM. The resulting test patterns are optimized to maximize defect coverage and minimize test time, since they require only a single write operation per victim cell.**

***Index Terms*—VMM, Computation-in-Memory, Structural Testing, RRAM**

I. Introduction

Artificial Intelligence (AI) hardware accelerators are a promising solution for efficient, cost-effective edge computing, with applications ranging from self-driving cars [1] to real-time biomedical diagnostics [2]. Companies like Tesla and Google have developed custom AI-focused ASICs to reduce reliance on power-hungry GPUs. However, these ASICs and GPUs still use the traditional Von Neumann architecture, which separates data storage from computation, causing latency and high energy consumption [3]. Computation-in-memory (CIM) AI Accelerators address this bottleneck by integrating data and computation, offering improved efficiency [4, 5]. These SoCs combine analog and digital circuits and often use memristive RRAM devices [6, 7], which present unique manufacturing defects [8, 9]. Testing CIM AI chips for post-manufacturing defects is challenging, requiring high detection rates with minimal testing cost.

Various testing methods have been proposed for conventional and emerging AI Accelerators, with the majority of them focusing on functional testing. Many use standardized inputs and outputs, such as signals or images to identify faults [10–12]. For example, Ma *et al.* introduced a scheme for analog memristive crossbar arrays that uses a reduced test set of diverse images to predict accuracy via an outlier detector [11]. The proposed method achieved a 10.3x test time reduction compared to using the entire initial test set. Similarly, Li *et al.* developed an online functional testing system for RRAM-based Neural Networks that deploys adversarial images to detect soft faults and incorporates memory testing algorithms or fault-aware retraining, which is an integral part of structural testing [12]. In terms of the proposed structural testing techniques, Chaudhuri *et al.* proposed a checkerboard-style ATPG approach for multicore systolic array accelerators, reducing test time by leveraging identical Processing Elements (PEs) and, thus, avoiding redundant test patterns [13]. Gebregiorgis *et al.* explored structural and functional testing for neuromorphic circuits, finding that functional pipelines offer higher fault coverage and lower runtimes but more false negative cases [14]. Studies have also addressed defects, fault models, and test solutions for memristive memories [15–17], but integrating high-level functional inputs with circuit-level defect analysis and accurate fault modeling remains a challenge [18].

This paper proposes a novel testing methodology for analog CIM cores. It demonstrates that the microarchitecture of a RRAM AI Accelerator is prone to different fault models than conventional RRAM memories, and thus cannot be tested the same way. We demonstrate that CIM cores allow for highly-parallel array testing, targeting multiple defects simultaneously with one-shot inference operations. The main contributions of the paper are to:

- Define defect and fault models for CIM AI Accelerator circuit components.
- Develop a testing methodology based on AI Accelerator inference, parallel reading operations for all device-level defects.
- Demonstrate tradeoffs of the methodology between testing time and defect coverage.

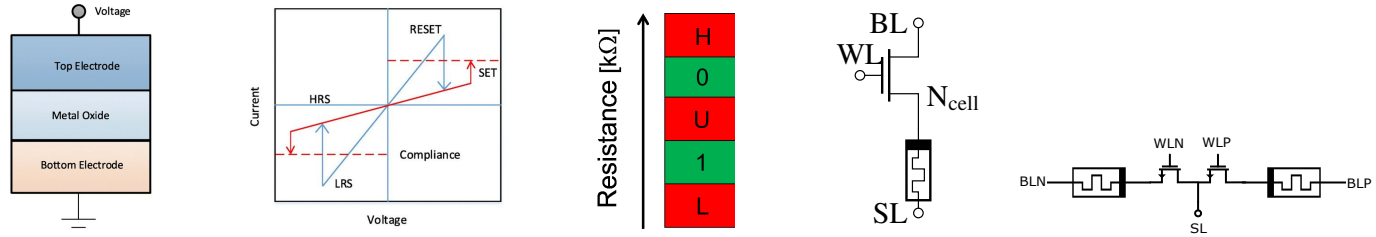
The remainder of the paper is structured as follows. Section II provides the required background. Section III compares conventional memory testing to AI Accelerator testing needs. Section IV analyzes the proposed testing methodology and the extracted results. Section V concludes the paper.

II. Background

A. RRAM

In CIM architectures, weight storage is implemented using Resistive Random Access Memory (RRAM), a two-terminal device built in a metal–insulator–metal (MIM) structure, as shown in Fig. 1a. To store a logic ‘1’, a voltage above

979-8-3315-9450-3/25/$31.00 ©2025 IEEE



(a) RRAM Structure [19] (b) RRAM bipolar I-V curve [19] (c) RRAM Resistance Range (d) 1T1R cell (e) 2T2R cell

Fig. 1: RRAM Device

TABLE I: Mapping of RRAM resistive states and BL voltage inputs to numerical values

	Positive Side	Negative Side	Value
Weights (W)	HRS	LRS	-1
	LRS	LRS	0
	LRS	HRS	1
Inputs (IN)	250mV	850mV	-1
	550mV	550mV	0
	850mV	250mV	1

V_{SET} is applied to the top electrode, forming a conductive filament (CF) between the electrodes. Conversely, applying a voltage below V_{RESET} at the bottom electrode breaks the CF, yielding a logic '0' with reduced conductivity. The I–V curve from these operations is presented in Fig. 1b and shows various resistive states, including nominal low and high states (LRS/HRS) and potential variations to very-low(L), very-high(H), or undefined states(U), as depicted in Fig. 1c.

The two main cell designs in a RRAM crossbar array are the One-Transistor-One-Resistor (1T1R) of Fig. 1d and the Two-Transistor-Two-Resistor (2T2R) of Fig. 1e. 1T1R includes the Word Line (WL) for cell activation, with the BL and SL for the read/write operations. Meanwhile, the 2T2R cell includes differential WLs (WLP & WLN) and BLs (BLP & BLN) for cell access and operation execution. That structure offers multi-state encoding of both inputs and weights, thus making it more suitable in the context of an AI Accelerator that requires signed operations. Inputs (IN) and weights (W) can have values [-1,0,+1] denoting negative, neutral, or positive numerical states respectively. The device-to-arithmetic matching for both RRAM weights and voltage BL inputs is provided on Table I. SL provides the output current as the difference between the cell currents, thus enabling multi-state dot product outputs from a single 2T2R cell. Overall, a negative output can be produced by both combinations of {W=+1, IN=-1} and {W=-1, IN=+1}, a positive output is generated by {W=+1, IN=+1} and {W=-1, IN=-1}, whereas for neutral output either W or IN needs to be set to 0.

B. Compute-in-memory

CIM architectures perform computations directly within the memory tile to reduce data transfer. In memristive AI accelerators, this mainly involves vector-matrix multiplication (VMM) between digital inputs and stored weights as shown in Fig. 2. Digital-to-Analog converters (DACs) convert input values into analog voltages applied across a memristor crossbar, where RRAM devices store weights as different conductance levels and generate proportional output currents. These currents are summed per column and converted back to digital form

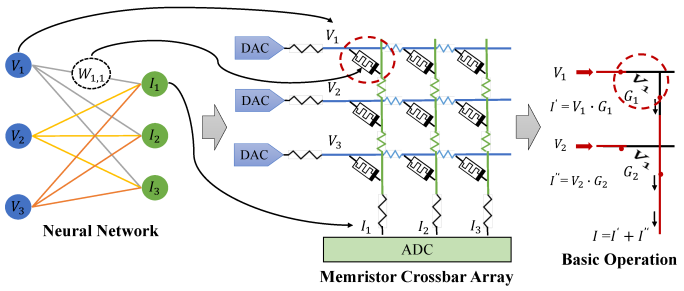


Fig. 2: VMM mapping on RRAM Crossbar [20]

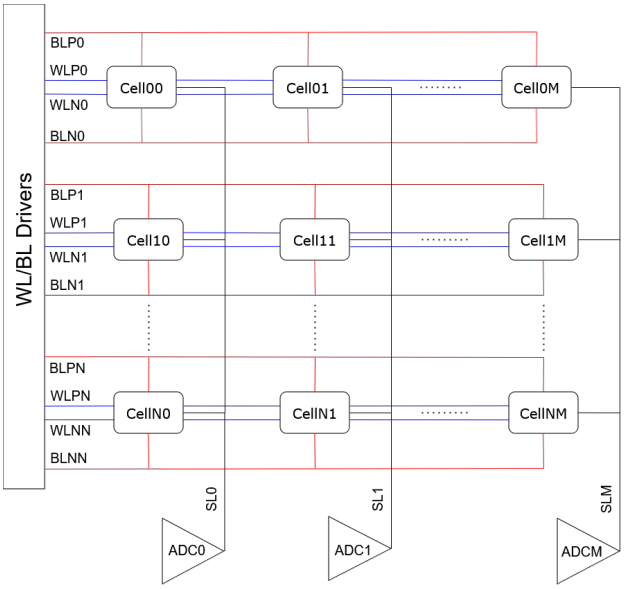


Fig. 3: Crossbar setup with N rows and M columns

by Analog-to-Digital converters (ADCs). This RRAM CIM method not only avoids the Von Neumann bottleneck but also offers non-volatile storage and $O(1)$ VMM complexity [21].

For conventional RRAM memory modules as the one in [22], the main circuit blocks are the address decoders (AD) for column selection, the Bit Line (BL) driver for data read/write, the Select Line (SL) driver for the RESET operation, as well as the Sense Amplifier (SA) for readout. The SA acts as a current comparator between the sensitized cell current and a reference value and results in a binary output. Each read or write operation can only be performed in a single cell at once, which makes reading the whole memory array an operation of $O(n^2)$ complexity.

For the purpose of converting a RRAM memory module to a CIM AI Accelerator, as the one of Fig. 3, memory address

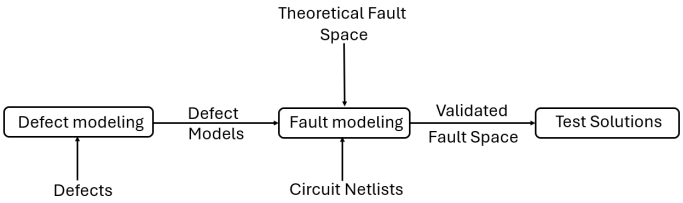


Fig. 4: Test Development Methodology

TABLE II: Common peripheral faults

	Static Faults	Dynamic Faults
Sense Amplifier	Stuck-at	Slow transition
Address Decoder	No access, Multiple access, Wrong cell access	Activation/Deactivation Delay
ADCs/DACs	Stuck-at, Gain, Offset, Non-linearity	Conversion Delay

decoders are replaced by DACs, which apply inputs to all rows in parallel, i.e., the circuit applies a parallel read operation on the entire row. The state of all cells on the same column is sensed as the accumulative current on the input of the ADC, which is done in $O(1)$ complexity during inference. This inherent difference in parallel cell access and the N-bit readout resolution of the ADC are the distinguishing factors between regular memory and AI Accelerator configuration.

III. Testing Methodology

Testing RRAM CIM cores includes three distinct steps, as in any structural test development, shown in Fig. 4. At first, the different types of defects need to be determined for the memory array and the peripheral circuits, in order to develop equivalent defect models. Next, the defect models are inserted in the initial netlist and they are mapped to the theoretical fault space, creating fault models. The fault space is defined for the peripheral circuitry and the RRAM cells. Finally, via fault analysis of the validated fault space, adequate test solutions are explored such as Design-for-test (DfT) circuits or pattern generation algorithms, that guarantee the highest possible fault detection.

A. Defect Modeling

Defects in the memory array may occur within a single RRAM cell or between neighboring cells [23]. For each RRAM device, defects could be modeled via simplified resistive defects. However, it has been proven that such an approach is not sufficient for accurate RRAM defect modeling. As a result, methods such as device-aware defect modeling provide a systematic approach by incorporating defects directly into device model parameters [24]. In contrast, conventional CMOS defects, such as interconnects and pinhole defects, can be modeled with linear resistors.

B. Fault Modeling

Similar to defect modeling, fault modeling is different between the memory array and the peripherals. Memory array fault modeling focuses on estimating the state of the victim RRAM cell in the presence of defects for all potential test patterns. For peripherals, fault modeling is based on the behavior of the defective circuit. Table II lists the main fault categories for peripherals used in memory and computation mode. The fault space for data converters used in AI Accelerators includes more static faults than the ones for decoders and SAs, namely non-linearity, gain and offset errors, which are traditionally characterized via functional testing sweeps. This makes the validation of the ADC fault space based on CMOS defects a challenging endeavor, especially when it comes to unifying peripheral and memory testing in a limited number of testing patterns.

C. Test Development

When it comes to test development, conventional March algorithms that deploy serial read and write operations have been extensively researched for memory testing. However, in RRAM CIM architectures, it has already been shown that computation configuration testing can lead to improved fault coverage, while reducing test time [24]. In contrast to regular memory operation, when performing digital logic or VMM operations, either two or all column cells are accessed each time, thus increasing the number of potential parallel sensitizing patterns for each CMOS or RRAM defect.

Another differentiating factor between testing memory and analog CIM refers to the small number of transistors and the two-state operation of the SA, which downscales the amount of potential defects and sensitization levels. In contrast, ADC circuits in RRAM CIM AI Accelerators include both analog and digital blocks with more components each, while offering multi-bit resolution on the digital output. This increases the different defects and output states that need to be considered for efficient test development.

IV. Application of the Testing Methodology

A. Defect modeling setup

The main types of defects targeted for this work are open, short, and bridge resistive defects within the 2T2R cell, as shown in Fig. 5. Note that defects between two nodes of neighboring cells are not included and will be part of future work. For every defect, the strength is swept between $10\,\Omega$ and $5\,\mathrm{M}\Omega$ in 10 logarithmically-spaced steps. Specialized RRAM defect modeling strategies such as Device-Aware Test [15], and CMOS defects for ADCs and DACs are not included in this work, thus focusing solely on 2T2R defect injection.

B. Fault modeling

1) Fault modeling setup: The fault modeling and analysis is based on parallel inference operations on each column, where the accumulative column current of multiple RRAM Devices is read by a single ADC. For the deployed input sensitizing sequence, a ($\mathrm{R}_P\mathrm{R}_N$:$\mathrm{IN}_P\mathrm{IN}_N$:$\mathrm{N}_i$) notation is utilized - with $i \in [1, 4]$ - for this work. R_P and R_N are the resistive states of the positive and negative sub-cells of the defective 2T2R cell, IN_P and IN_N are the positive and negative differential inputs provided from the BL driver and N_i are the equivalent IN·W states of the neighboring cells.

A graphical representation of the input patterns is given in Fig. 6. Neighbors N_1 to N_3 are unique RRAM cells each with their discrete IN·W value of [-1,0,1]. The choice of three

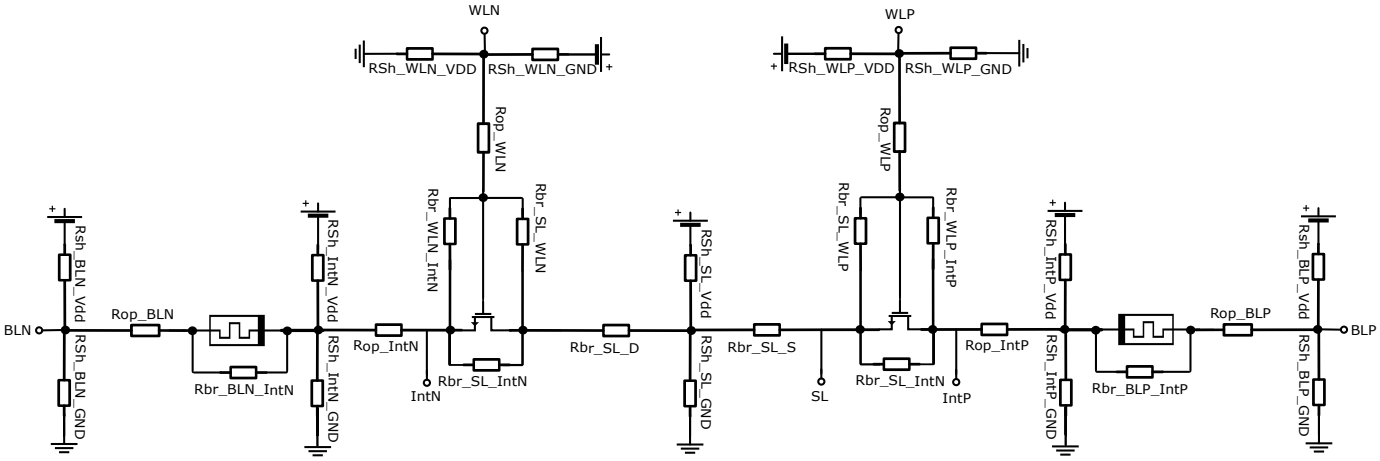


Fig. 5: Defect injection

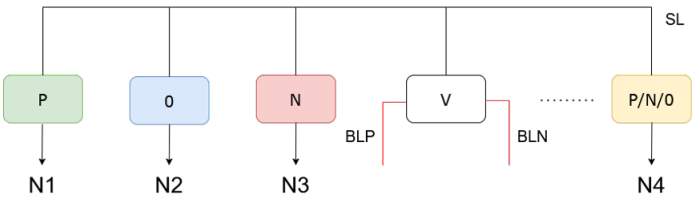


Fig. 6: Input patterns for victim cell and its neighbors on a single column

neighbors comes due to the three different arithmetic states provided by each 2T2R cell. Choosing less than 3 discrete neighbors (e.g. 2 or 1) would lead to limited output current discrete levels on the input of the ADC. Neighbor N_4 includes a bulk of all the rest RRAM cells, apart from the victim cell and the neighbors N_1-N_3. All 4 cells included in the N_4 group have the identical IN·W value, which leads to fewer patterns for test generation and reduced simulation overhead. For example, the sequence (**+1-1:00:-1-1-1-1**) requires writing LRS to R_P and HRS to R_N, applying neutral voltage ($550mV$) to both BL_P and BL_N and setting all neighboring cells to a negative arithmetic product IN·W.

Simulations are performed on the analog CIM Crossbar architecture of Fig. 3 for M=1 column and N=8 rows. The analysis assumes that only a single 2T2R cell - as the one of Fig. 1e - can be defective each time, with the remaining 7 2T2R cells considered non-defective. All different combinations of inputs and weights are taken into consideration for the defective cell. For the neighboring column devices, only the equivalent IN·W product is considered as it affects the accumulated current on the output SL, a scheme that reduces combinations for each neighboring cell from 9 to 3. Finally, the 7 neighboring cells are split into 3 unique cells and a bulk of 7-3=4 cells that produce identical output current (positive, neutral or negative).

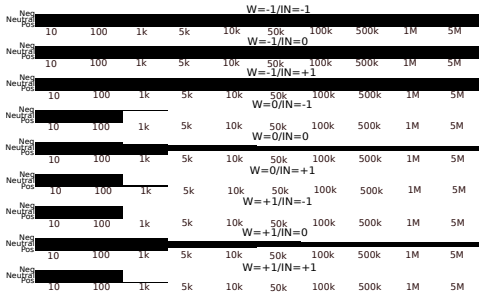
The circuit architecture of Fig.3 is implemented in Cadence Virtuoso and simulated via Cadence Spectre. In terms of the simulation settings, the TSMC 40nm technology is employed for all CMOS devices along with the RRAM model from [25]. Capacitive loads of $150fF$ are added to the inputs and the outputs of the WL, BL, and SL drivers. The nominal voltage for the WL drivers is set to $1.6V$, whereas the BL drivers and the ADC operate at $1.1V$. A mixed-signal, voltage-controlled oscillator (VCO) 8-bit ADC is utilized from [26]. The design has been validated in an analog CIM setup, with reduced area overhead, allowing one ADC per column. Each distinct simulated netlist includes a single defect and column input.

2) Fault modeling results: Fig. 7 presents the different types of fault maps generated from the defect injection setup. On each subfigure, column indices correspond to the defect strengths on a logarithmic $10-5M\Omega$ scale. Each subfigure is also split vertically into 9 tiles, based on the different input and weight states of the simulated victim cell. The 81 rows of each tile represent the 3^4 potential states of all neighbor cells sorted from the most negative to the most positive combination, as notated on the left-hand side of each tile.

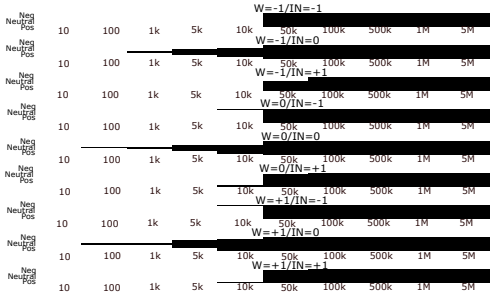
The color map on each figure represents whether a single input-defect combination sensitizes a fault (white) or not (black). Non-sensitized faults produce the same output code on the ADC as the non-defective simulation, while the sensitized ones generate at least a Least Significant Bit (LSB) of difference on the digital output. Gray color is applied only on tiles that are fully non-sensitized for all defect strengths and input patterns.

Figs. 7a and 7b showcase the results on an open and a bridge resistive defect. As expected, opens up to $100\ \Omega$ and bridges higher than $50k\ \Omega$ do not sensitize faults for all input patterns. Furthermore, both defects demonstrate lower defect coverage from neutral input patterns (IN_P=IN_N=0), which are located in the middle rows of each subfigure tile. For the open defect, similar lack of detectability is observed for patterns that include negative victim cell weights (R_P=-1 and R_N=1), which correspond to the upper 3 tiles of Fig. 7a.

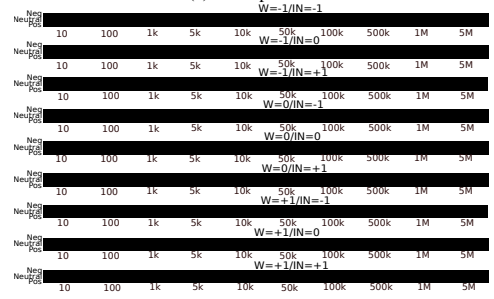
Figs. 7d and 7c are the fault maps for BL and WL shorts. In contrast to open and bridge defects, the detectability of such defects is non-existent for the WL and limited only to a few, low resistive values for the BL. Especially in the WL case, the WL driver is constantly activated during inference, offering a steady input voltage of $1.6V$ to activate each access transistor



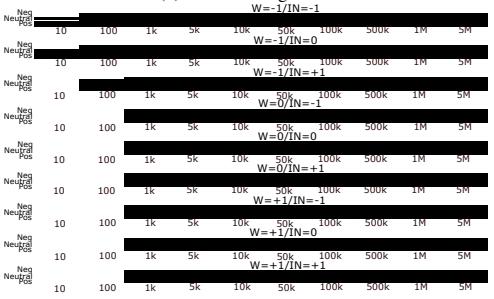
(a) BL Open defect



(b) RRAM bridge defect



(c) WL-VSS Short Defect



(d) BL-VDD Short Defect

Fig. 7: Fault maps for four different types of defects

of the 2T2R cell. Moreover, due to the large width of the employed WL driver transistors (W=25um), the WL driver has a lower resistance path to the resistive defect connected to ground. Thus, only minor fluctuations are expected on the WL voltage of the defective cell, whose operation is not affected. Lastly, for the BL-VDD defect, the input patterns with victim cell weights set at neutral or positive are preferable for maximum defect coverage of more strengths values.

C. Test development

1) Testing Setup: The generated fault maps for all defects are the basis for developing adequate test solutions, which can be delivered in the form of DfT schemes or Automatic Test Pattern Generation (ATPG) algorithms. As a part of this work, a custom ATPG mechanism is developed. It is in the form of an Integer Linear Programming (ILP) optimization problem, that generates the test vector with the highest fault coverage and the minimum length by combining input signals and weights. The aforementioned target can be mathematically described via the following equation:

$$\min \sum_{k=1}^{S} (w_k \cdot S_{sel,k}), \tag{1}$$

which is subject to:

$$\min \sum_{j=1}^{p} (a_{k,j} \cdot S_{sel,j}) \geq 1, \tag{2}$$

for all rows k. In more detail, $S_{sel,k}$ indicates whether the kth sensitization input vector has been selected or not and takes a binary value. The selection has a unique weight value w_k, which is equal to 1 for all sequence patterns deployed. Equation (1) is responsible for selecting the minimum number of sensitizing sequences, while Equation (2) guarantees that all defects are detected by at least one input in vector S_{sel}. As a result, the binary matrix a, which includes elements $a_{k,j}$ of Equation (2) and encodes the generated fault maps per defect, should include at least one non-zero value. To solve the set ILP problem, we utilize Python's SciPy library for all the simulated fault maps [27].

2) Testing Results: Based on all unique fault maps, an optimized input sequence can be generated for all 32 partially or fully detectable defects. As observed in Section IV-B for the BL open defect, multiple different inputs can be used to detect the same defect strengths, which can be scaled up to the entire set of fault maps. For that purpose, the ILP optimization problem is set in order to select the minimum amount of inputs that detect all sensitizable defects, which is set as a hard constraint. Defect strengths that are fully non-sensitizable are excluded from the problem formulation, as no existing pattern can reliably detect them and it would lead to an infeasible problem formulation.

Based on the aforementioned, the solver produces a set of test patterns that include a minimum amount of column inputs, while still sensitizing all potential defects. In order to select the final sequence, all different sensitizing patterns of minimum length were scored upon the number of write operations necessary. That is due to the higher cost of write operations for RRAMs reliability and write delay, in contrast to read or inference operations. The most optimal sequence found for that purpose is: **{(-1+1:-1+1:-1-1-1-1),(-1+1:+1-1:+1+1-1-1),(+1+1:+1-1:-1-1-1-1)}**.

The generated test vector includes only 3 inference operations and requires a single write operation per victim cell. The neighbor cells, since they are all either negative or positive, must only be initialized once, at opposite resistive states within

TABLE III: Test time comparison for structural and functional RRAM tests for a 128x128 array.

Algorithm	Type	Read/Write	Images	Time(s)	Fault coverage(%)
RRAMedy [12]	Functional	-	1100	0.11	93
Low-cost test [11]	Functional	-	300	0.03	84
March C* [17]	Structural	6N/4N	-	1.19	99.3
March MOM [28]	Structural	4N/5N	-	1.44	99.8
This work	**Structural**	**3N/1N**	**-**	**0.32**	**85**

the same cell. In the simulation setup, all positive and negative 2T2R cells were set at R_P=LRS and R_N=HRS. The distinction between a positive and a negative neighboring cell can be provided by adequate differential signals on the BL based on the arithmetic demonstration of Table I instead of rewriting them for each pattern on the input sequence. The generated test vector also validates the intuitive conclusion drawn from Fig. 7 that positive and negative input patterns are preferred over neutral ones for better defect coverage.

Table III offers a comparison of 3 structural and 2 functional test approaches for RRAM CIM microarchitectures. For calculation of the effective test time for structural approaches, we assume a read time of 0.5 μs and write time of 10μs, which were utilized in the deployed simulation setup. For the functional test approaches, each image is assumed to have 100 μs inference time [6]. Based on the above, we notice that our approach offers the highest speed-up time in contrast to all other structural test approaches. At the same time, its coverage is based on circuit-level defects and not on high-level fault models, such as stuck-at and delay faults, that are being deployed by the functional test approaches. However, due to a lack of detectability of the WL defects as the one of Fig. 7b, it leads to an inferior defect coverage percentage, with its improvement being part of future work.

V. Discussion and Conclusion

This paper presented a new column-based structural testing strategy for analog CIM AI cores. It was shown that with a minimum-length, single-write test vector, all cell-level resistive defects can be sensitized. We observe the following:

Applicability: Data converters do not complicate the testing mechanism, if considered non-defective. That assumes that they have been structurally tested beforehand. Otherwise, data converter test patterns need to be added, as a part of the whole RRAM CIM testing, which is left as future work. Furthermore, testing of activation functions, such as Relu and tanh functions, is different to RRAM-CIM testing with ADC, as such circuits are implemented as either fully digital or analog. Such approach also applies for other types of memory defects such as interconnects and device-aware ones.

Structural Testing: This paper has demonstrated that structural testing approaches for RRAM-based AI accelerators are more time efficient than functional test proposals, such as adversarial pattern generation or statistical/regression fault modeling. Furthermore, because these test approaches are structural, they can also guarantee the detection of all defects, whereas functional testing relies on abstract fault models for that purpose.

Defect Coverage: The generated test patterns are able to sensitize all studied defects, apart from the WL-VDD and WL-VSS short defects. These defects can also be detected if the ADC is read out while the WL is not activated.

Extension to Other AI Accelerator Circuits: The presented test development methodology is universal and can also be applied to other analog AI accelerator technologies, e.g. phase change memory (PCM), or even SRAM-based ones.

References

[1] S. Liu *et al.*, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, 2019.

[2] J. N. Tripathi *et al.*, "Hardware accelerator design for healthcare applications: Review and perspectives," in *ISCAS*, 2022, pp. 1367–1371.

[3] N. Talati *et al.*, "MMPU—a real processing-in-memory architecture to combat the von neumann bottleneck," *AEMT: BS*, 2020.

[4] S. Hamdioui *et al.*, "Applications of computation-in-memory architectures based on memristive devices," in *DATE*, 2019.

[5] G. Krishnan *et al.*, "In-memory computing for ai accelerators: Challenges and solutions," in *EML-CP-IoT-EC:HA*, 2023.

[6] Q. Liu *et al.*, "33.2 a fully integrated analog reram based 78.4tops/w compute-in-memory chip with fully parallel mac computing," in *ISSCC*, 2020, pp. 500–502.

[7] C.-X. Xue *et al.*, "Embedded 1-mb reram-based computing-in- memory macro with multibit input and weight for cnn-based ai edge processors," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 1, 2020.

[8] L. B. Poehls *et al.*, "Review of manufacturing process defects and their effects on memristive devices," *ACM JET*, 2021.

[9] M. Fieback *et al.*, "Defects, fault modeling, and test development framework for rrams," *J. Emerg. Technol. Comput. Syst.*, Apr. 2022.

[10] S. A. El-Sayed *et al.*, "Compact functional testing for neuromorphic computing circuits," *IEEE TCAD-ICS*, 2023.

[11] K. Ma *et al.*, "Efficient low cost alternative testing of analog crossbar arrays for deep neural networks," in *ITC*, 2022, pp. 499–503.

[12] W. Li *et al.*, "Rramedy: Protecting reram-based neural network from permanent and soft faults during its lifetime," in *ICCD*, 2019.

[13] A. Chaudhuri *et al.*, "C-testing and efficient fault localization for ai accelerators," *IEEE TCAD of Integrated Circuits and Systems*, 2022.

[14] A. Gebregiorgis *et al.*, "Testing of neuromorphic circuits: Structural vs functional," in *ITC*, 2019, pp. 1–10.

[15] M. Fieback *et al.*, "Device-aware test: A new test approach towards dppb level," in *ITC*, 2019, pp. 1–10.

[16] P. Liu *et al.*, "Fault modeling and efficient testing of memristor-based memory," *IEEE TCAS I: Regular Papers*, 2021.

[17] C.-Y. Chen *et al.*, "Rram defect modeling and failure analysis based on march test and a novel squeeze-search scheme," *IEEE TC*, 2015.

[18] F. Su *et al.*, "Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives," *IEEE Design & Test*, 2023.

[19] N. Jain *et al.*, "Resistive random access memory: Materials, filament mechanism, performance parameters and application," in Oct. 2022.

[20] D. Gao *et al.*, "A design methodology for fault-tolerant neuromorphic computing using bayesian neural network," *Micromachines*, 2023.

[21] A. Gebregiorgis *et al.*, "Tutorial on memristor-based computing for smart edge applications," *M-MDCS*, 2023.

[22] S. Hamdioui *et al.*, "Testing computation-in-memory architectures based on emerging memories," in *ITC*, 2019, pp. 1–10.

[23] H. Xun *et al.*, "Data background-based test development for all interconnect and contact defects in rrams," in *ETS*, 2023, pp. 1–6.

[24] M. Fieback *et al.*, "Testing scouting logic-based computation-in-memory architectures," in *ETS*, 2020, pp. 1–6.

[25] H. Li *et al.*, "A spice model of resistive random access memory for large-scale memory array simulation," *IEEE EDL*, 2014.

[26] M. Mayahinia *et al.*, "A voltage-controlled, oscillation-based adc design for computation-in-memory architectures using emerging rerams," *J. Emerg. Technol. Comput. Syst.*, Mar. 2022.

[27] P. Virtanen *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, 2020.

[28] S. Kannan *et al.*, "Sneak-path testing of crossbar-based nonvolatile random access memories," *IEEE Transactions on Nanotechnology*, 2013.