Selective Sensitization of Useless Sneak-Paths for Test Optimization in Memristor-Arrays

Manobendra Nath Mondal, Susmita Sur-Kolay, and Bhargab B. Bhattacharya
Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata-700108, India
E-mail: manobmondal@gmail.com, {ssk, bhargab}@isical.ac.in

***Abstract*—Memristors have shown significant promise in recent times both in logic synthesis and memory-subsystem design. A 2D-crossbar architecture built with memristor arrays provides a convenient platform for storing multi-valued memory states by utilizing the analog variation of current-induced resistance through these cells. The integration of CMOS components and non-CMOS memristor cells further enhances the scope of their applications to various complex system design. However, present-day memristor cells, by virtue of their inherent structural constructs, are prone to various manufacturing defects and sensitive to operational modalities. Therefore, robust and fast testing methods are frequently needed to ensure the correctness of their functionality. Existing techniques for testing memristor arrays are either ad-hoc in nature or suited for application-specific designs, and no attempt has been made to optimize test-time. In this work, we present a graph-based technique to select some sneak-paths and sensitize these to generate test-vectors for a 2D memristor-array. The proposed method relies on Eulerian graph traversal and matching in bipartite graphs to produce tests for detecting stuck-at, slow-to-write, and deep faults in all memristor-cells. Simulation results with LTspice demonstrate the effectiveness and superiority of the proposed method to prior art in terms of test time and fault-coverage.**

***Keywords*—Memristor crossbar, Test optimization, Path sensitization, Sneak-path, Bipartite graph.**

I. Introduction

In the year 1971 Prof. Leon Chua, in a groundbreaking work, hypothesized that apart from resistor, capacitor, and inductor, there exists a fourth fundamental passive circuit element possessing the traits of both memory and resistor, and named it as Memristor [1]. Later in 2008, a group of researchers from HP Labs successfully manufactured the first memristor [2]. Due to its unique property and compatibility with CMOS technology, it has surfaced as a powerful circuit element and thus significantly influenced upcoming design paradigms [3] [4]. Today, memristors are used in memory systems, digital and analog circuits, neuromorphic computers, and for many other applications [5] [6]. Logic synthesis using activation of sneak-paths in memristor crossbar-arrays has also been proposed [7]. New generation of hybrid memory-array integrates both memristor technology and CMOS, where the former is used as storage elements and the latter to build interface circuits. Due to inherent intricacies of memristor devices, such memory systems become susceptible to several manufacturing and field defects, and hence, they need robust testing to ensure the correctness of their functionality [8].

Testing methods based on traditional march algorithms [9] are observed to be suitable for memristor-based memory arrays [10]. Test generation is performed for detecting stuck-at-faults (SAF), read-one-disturb (R1D) and coupling faults (CP). However, test-time may increase significantly for large-sized arrays. A design-for-test (DfT) scheme for memristor arrays was proposed in [11]. Analyzing various physical defects, Kannan et al. proposed a fault model for SAF in memristor-arrays, and a test technique making use of sneak-paths [12]. In [13], the fault model was extended to testing of multi-valued cells, and test is performed by simultaneously reading multiple memristor-cells known as region-of-detection (RoD). The size and shape of RoDs depend on the nature of faults and at least one memristor from each RoD ought to be accessed for test [12–14]. Unfortunately, such a condition may not be fulfilled in all cases. Moreover, RoD is formed sequentially by including a single memory cell satisfying certain constraints, and thus, concurrent writing may not be possible. Additionally, most of the previous approaches do not address multiple stuck-at-0 ($SA0$) or stuck-at-1 ($SA1$) faults. In this paper, we describe a new technique for testing 2D memristor arrays based on the sensitization of otherwise useless sneak-paths. The proposed method uses graph-based path-covering techniques to optimize test time for a general class of faults, and outperforms all previous solutions in test cost and fault-coverage.

The rest of the paper is organized as follows. In Section II, a brief description of the crossbar architecture for a memristor-array is given, and fault models are discussed. Section III presents the path-selection strategy used for testing. Experimental results are reported in Section IV, and finally, Section V concludes the paper.

II. Memristor Model, Crossbar Architecture, Sneak-Path and Fault Model

1) Memristor Model: A memristor is a two-terminal passive element in which the magnetic flux (ϕ) between the terminals is a function of the amount of electric charge (q) that passes through the device [1, 2].

The memristance M, measured in *ohm*, is equal to $\frac{d\phi}{dq}$. The internal state $\alpha(t)$ of a memristor at instant t, is given by $w(t)/D$, where $w(t)$ is the length of the doped region and D, the total length of the memristor Fig. 1(a)). The memristance M can also be expressed as $\alpha M_{on} + (1-\alpha)M_{off}$, where M_{on} and M_{off} are respectively the values of the memristance when $w(t) = D$ and $w(t) = 0$. An equivalent resistive model of a memristor (Fig. 1(b)) has the corresponding resistance as $R_{MEM} = \alpha R_{ON} + (1-\alpha)R_{OFF}$, where R_{ON} and R_{OFF} are

The work of B. B. Bhattacharya was supported partially by INAE Chair Professorship (2016-2018).

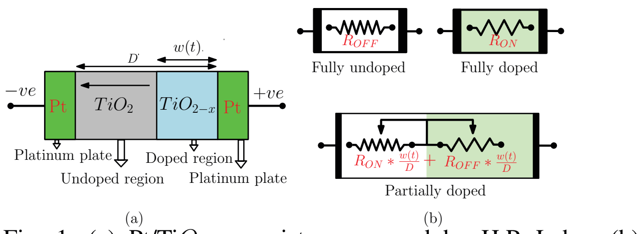

Fig. 1: (a) Pt/TiO_2 memristor proposed by H.P. Labs., (b) equivalent resistive model.

the equivalent resistances when the memristor is fully doped and un-doped, respectively.

2) Crossbar Architecture: An $n \times n$ crossbar architecture of memristors consists of n horizontal wires or Word-Lines (WL), n vertical wires or Bit-Lines (BL), and a two dimensional $n \times n$ array of memristors $M_{ij}, 1 \leq i, j, \leq n$, where the memristor M_{ij} has one terminal connected to the wire WL_i and the other to the wire BL_j (Fig. 2(a)). A memristor M_{ij} acts as a memory cell in the crossbar architecture, and is accessed by applying appropriate voltage between wires WL_i and BL_j.

Apart from $1M$ architecture (as in Fig. 2(a)), there are other types of crossbar architecture, namely $1D1M$, $1T1M$, where a diode or a transistor is connected in series with each memristor through RRAM/CMOS heterogeneous fabrication technology. A $1T1M$ crossbar architecture is shown in Fig. 2(b).

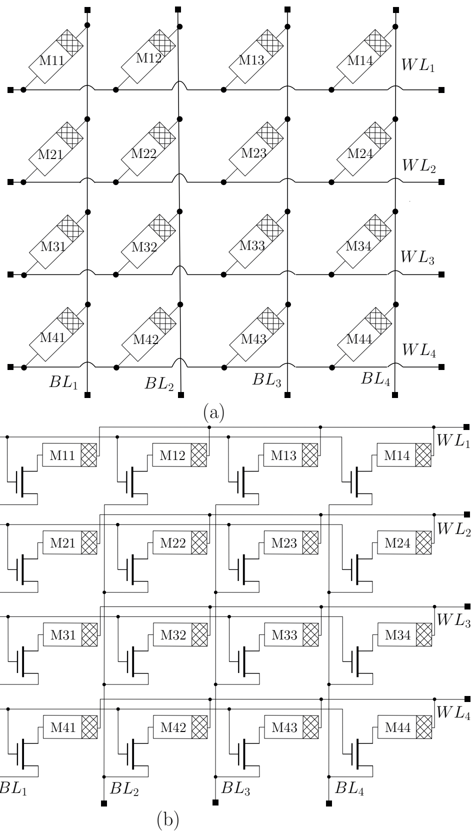

Fig. 2: Memristor crossbar architecture using (a) $1M$ (b) $1T1M$ elements.

3) Sneak-Path: Since memristors are bi-directional devices, while accessing a particular memristor in a crossbar architecture, a small amount of current may also flow through some un-selected memristor cells. This current is known as sneak-current, and the path through which it flows is called a sneak-path (see Fig. 3.) $1D1M$, $1T1M$ crossbar architectures are mainly used to mitigate undesired sneak currents in functional mode.

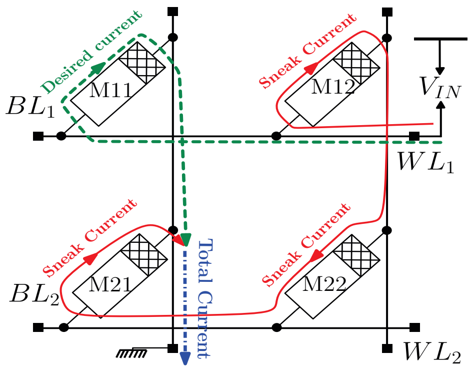

Fig. 3: Desired current (green line), sneak-current (red line) and total current (blue line) in a 4×4 crossbar architecture when input voltage is applied across memristor M_{11}.

4) Fault Model: The high memristance state of a memristor is defined as OFF state and denoted as $Logic$-0. Similarly, the low memristance state of a memristor is defined as ON state and denoted as $Logic$-1. Different types of defect may occur due to hetero-structure fabrication process and parametric variations [14] [15], which are modelled as following faults:

- *Stuck-at-0 ($SA0$)*: A memristor is always turned off irrespective of the voltage applied across it. This may be caused due to deficiency of doping.
- *Stuck-at-1 ($SA1$)*: A memristor is turned on permanently regardless of the voltage applied. It may happen due to excessive doping.
- *Slow-write-1 ($SW1$)*: A single voltage-pulse for performing write is inadequate to change a memristor from $Logic$-0 to $Logic$-1 state.
- *Slow-write-0 ($SW0$)*: Similarly, a single write voltage pulse is not capable of changing the state of a memristor from $Logic$-1 to $Logic$-0.
- *Deep-0 ($D0$) and Deep-1 ($D1$)*: An increase in the length or a decrease in the cross-section results in $D0$ fault, whereas the opposite phenomena causes $D1$.

III. Path Sensitization Of Selective Sneak Path

The underlying principle of sneak-path based crossbar memory testing is to sense the current through either a few selective or all possible sneak-paths in the absence and the presence of a fault, i.e., I_{ideal} and $I_{measured}$ respectively, and to compare these in order to detect faults. In the proposed scheme, for detecting single/multiple $SA0$ faults, we selectively activate only one sneak-path at a time and test all the memristors lying on it; while testing single/multiple $SA1$ faults, we activate n parallel sneak-paths for an $n \times n$ crossbar and test them simultaneously.

Testing of parallel paths is not suitable for $SA0$ faults as memristors are sensitized with $Logic$-1, and hence, the total current flowing through all the parallel sneak-paths may become very high in the fault-free case.

The length l_p of a selected sneak-path is the number of memristors present in that sneak-path. During testing, the maximum path-length that can be chosen depends on memristor parameters, fault types, applied voltage, and current sense-amplifier. Note that a choice of longer and edge-disjoint paths reduces test time for detecting $SA0$ faults.

A. *Proposed Method for Testing*

We consider $1T1M$ architecture where selected sneak-paths can be conveniently sensitized. When we access a specific memristor in an $n \times n$ crossbar, we get a corresponding memristor network comprising all possible sneak-paths (hereafter referred to as *sneak_path_network*). In Fig. 4 and 5(a), we have shown (only the memristors) a 4×4 $1T1M$ crossbar and its corresponding $sneak\_path\_network$ while accessing memristor M_{11}.

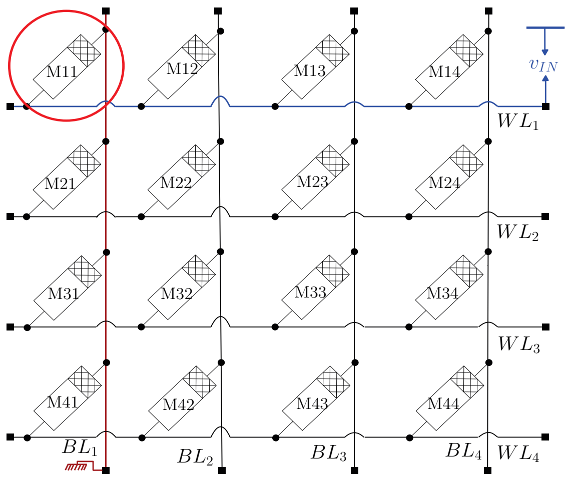

Fig. 4: Accessing memristor $M11$ in test mode in a 4×4 $1T1M$ crossbar

1) Graph Construction: From the $sneak\_path\_network$ for a memristor $M_{i,j}$, we construct a graph $G_{spn} = (V, E)$ where there is a vertex in V for each of the BLs and the WLs, and an edge (v_i, v_j) in E for each memristor. By utilizing well-known graph algorithms, we identify the desired sneak-paths in the network which are to be sensitized. The graph corresponding to Fig. 5(a) is shown in Fig. 5(b).

The graph G_{spn} has a source vertex u and a destination vertex v representing the WL connected to v_{IN}, and the BL connected to the ground. The vertices adjacent to u and v are in the subsets L and R, respectively such that $V = L \cup R \cup \{u, v\}$. Test-time reduction for a given crossbar memristor-array can be formulated as TEST OPTIMIZATION BY SELECTIVE SNEAK PATH (TOSSP): *Given a G_{spn} and path-length l_p, determine the minimum number ν of paths from vertex u to vertex v such that the length of each path is less than or equal to l_p, and collectively these cover all the edges in E.*

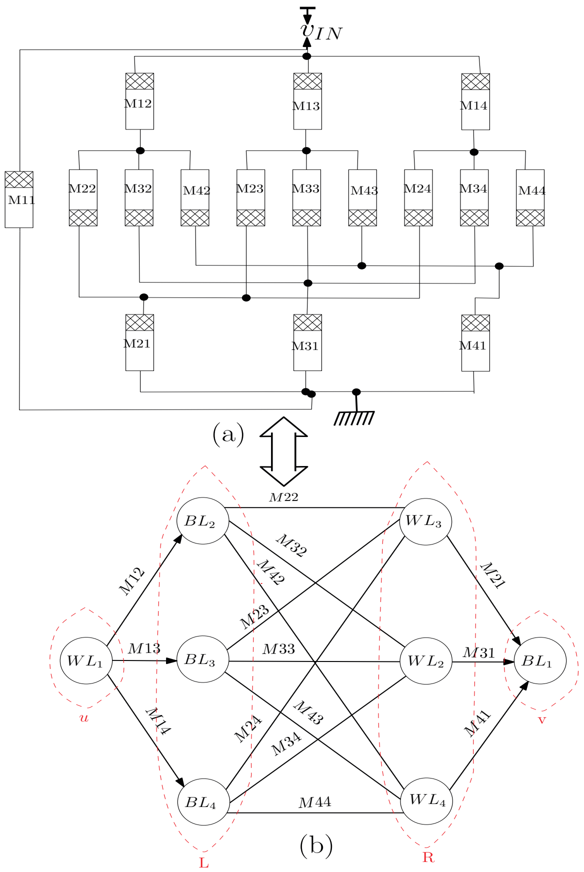

Fig. 5: (a) $Sneak\_path\_network$ while accessing $M11$ in a 4×4 crossbar, (b) its corresponding graph G_{spn}

Note that a G_{spn} may correspond to access of a specific memristor but the ν sneak paths sensitize all the memristors in the crossbar array.

2) Graph Decomposition: In order to solve TOSSP, we decompose G_{spn} into three sub-graphs (Figs. 6(a), (b) and (c) resp.) as follows.

Boundary Directed Source Graph $G_{BDS} = (V_s, E_s)$ where $V_s = \{u\} \cup L$ and $E_s = \{< u, l_i > | l_i \in L\}$.

Boundary Directed Destination Graph $G_{BDD} = (V_d, E_d)$ where $V_d = R \cup \{v\}$ and $E_d = \{< r_j, v > | r_j \in R\}$.

Intermediate Complete Undirected Bipartite Graph $G_{ICB} = (V_i, E_i)$ where $V_i = L \cup R$ and $E_i = \{(l_i, r_j) | l_i \in L, r_j \in R\}$.

Since $|E_i| >> |E_s| + |E_d|$, our main aim is to find edge-disjoint paths in G_{ICB}, and then addition of two boundary edges from G_{BDS} and G_{BDD} to get the complete sneak path.

3) Single and Multiple $SA0$ Detection: Depending on l_p in G_{ICB} for an $n \times n$ crossbar, two cases arise.

Case I: $l_p \geq n - 1$

For each selected path in G_{ICB}, no vertex is repeated, hence the maximum path-length is $n - 1$ instead of l_p. Thus the TOSSP problem reduces to *finding the minimum number of paths with length $l_p = n - 1$ in G_{ICB} which covers all the edges E_i at least once.*

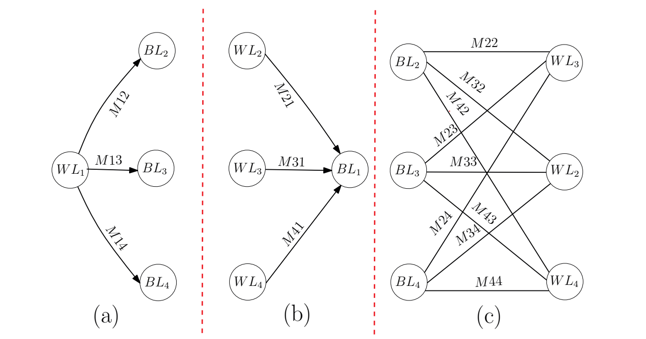

Fig. 6: Decomposition of G_{spr} in Fig. 5: (a) $G_{BDS}(V_s, E_s)$ (b) $G_{BDD}(V_d, E_d)$ (c) $G_{ICB}(V_i, E_i)$.

Since G_{ICB} is the complete bipartite graph $K_{n-1,n-1}$, we have $n-1$ perfect matching, each of size $n-1$. We add those $n-1$ perfectly matched edges one by one to get $n-1$ edge disjoint paths.

For even n, the two endpoints of each path in G_{ICB} lie on the vertex sets L and R, respectively. But for odd n, both the endpoints of each path lie either in the vertex set L or in R. We repeat an edge for each path to go to the other vertex set increasing the path length by one; however, the number of paths remain the same.

In Fig. 7, we show three perfectly-matched edge sets for a G_{ICB} $K_{3,3}$ and Fig. 8 shows the complete set of three paths, each of length five obtained by adding two boundary edges from G_{BDS} and G_{BDD} for a 4×4 crossbar.

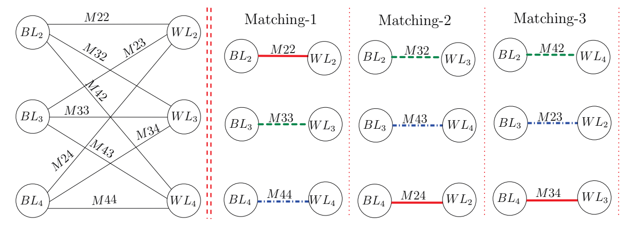

Fig. 7: $K_{3,3}$ G_{ICB} and a cover with three edge-disjoint perfect-matching.

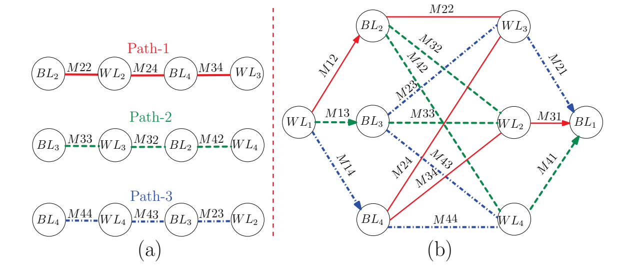

Fig. 8: Illustration of Case I ($l_p = n = 3$): (a) the three paths in G_{ICB}, and (b) three complete sneak-paths in G indicated by different colours taking one edge from G_{BDS} and G_{BDD}.

Case II: $l_p < n-1$

In this case, the TOSSP problem reduces to *finding minimum number of paths of length* l_p *in* G_{ICB} *covering all the edges in* E_i. The following two sub-cases arise depending on the value of n.

Sub-case IIa: n is odd

The degree $(n-1)$ of each vertex in G_{ICB} is even. We construct an Eulerian cycle [16] covering all the edges. Then we cut the cycle into path-segments of length l_p each sequentially. So there are $\lceil \frac{(n-1)\times(n-1)}{l_p} \rceil$ path-segments. If $(n-1)$ is not divisible by l_p then the last path is of length less than l_p.

Fig. 9 shows an Eulerian path of length 16 and its segmentation into sneak-paths, each of length three, in a $K_{4,4}$ G_{ICB}.

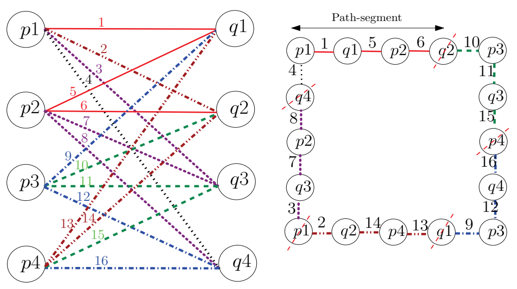

Fig. 9: Illustration of Case IIa with $n = 5, l_p = 3$: G_{ICB} is $K_{4,4}$, and its corresponding Eulerian cycle of length 16 with 5 path segments.

Sub-case IIb: n is even

The degree of each vertex in G_{ICB} is odd, and there is no Eulerian cycle. Deletion of edges belonging to one perfect matching results in even degree of each vertex, and then the problem reduces to the previous case. For testing the $(n-1)$ matched edges which are deleted, we include an edge between two consecutive matched edges. Hence, the total number of sneak paths required is $\lceil \frac{(n-2)(n-1)}{l_p} \rceil + \lceil \frac{2(n-1)}{l_p} \rceil$. Fig. 10 shows the Eulerian cycle for a G_{ICB} which is $K_{5,5}$, and the additional sneak paths for testing the 5 deleted matched edges.

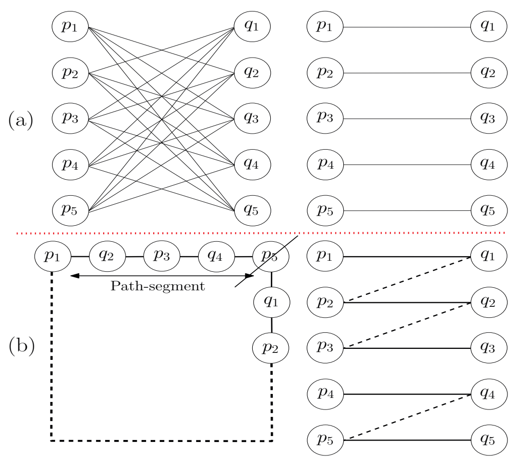

Fig. 10: Illustration of Case IIb for $n = 6, l_p = 4$, $K_{5,5}$ as G_{ICB}: (a) Eulerian cycle after deleting a perfect matching, (b) sneak paths from the Eulerian cycle and the additional ones for the 5 deletedd matched edges .

4) Single and Multiple $SA1$ Detection: We consider each sneak-path of length one in G_{ICB}; note that very small sneak-current passing through each path allows us to test these faults in parallel, and thus, enables us to reduce test-time. Therefore, TOSSP problem reduces to that of *finding maximum number of vertex-disjoint parallel paths of unit length in G_{ICB}*. Equivalently, this can be solved by constructing a perfect matching in G_{ICB} and testing all n edges belonging to each matching-set, in parallel.

Note that the presence of a single or multiple $SA1$ faults may cause the output current to become very high. A very large resistor is used in series to limit the output current. Fig. 7 illustrates three matchings that cover all edges corresponding to $K_{3,3}$ G_{ICB}. All edges in each matched solution can be tested in parallel.

IV. Experimental Results

A. Theoretical Analysis

We deploy four standard memory-access operations as mentioned below [9]:

- $\{w0, w1\} \Rightarrow$ Write a memory cell with logic value zero and one, respectively.
- $\{r0, r1\} \Rightarrow$ Read a memory cell with expected logic value zero and one, respectively.

We apply appropriate combinations of four access-modes sequentially in order to test all faults considered in the fault model. Sensitization of a particular sneak-path allows concurrent write of $Logic$-0 or $Logic$-1 in all the memristors lying on the selected path. Consequently, this technique leads to reduction of test-time as demonstrated below.

***Stuck-at-0** ($SA0$)*: A $SA0$ fault can be detected by applying the sequence $\{w1, r1\}$. When $l_p \geq n-1$ in an $n \times n$ crossbar, n tests are sufficient for writing $\{w1\}$ and for reading $\{r1\}$ whereas for $l_p \leq n$, test time will be $\lceil \frac{(n-1)(n-1)}{l_p} \rceil$ and $\lceil \frac{(n-1)(n-2)}{l_p} \rceil + \lceil \frac{2(n-1)}{l_p} \rceil$ for even and odd n, respectively.

***Stuck-at-1** ($SA1$)*: The sequence $\{w0, r0\}$ detects a $SA1$ fault. A total of $(n-1)$ tests cover all such faults in the entire $n \times n$ crossbar.

***Deep-0** ($D0$)*: The sequence $\{w0, w0, w1, r1\}$ can be applied to test deep-zero faults. As three sequential write pulses are needed, write-time becomes three times more than read-time.

***Deep-1** ($D1$)*: To test $D1$, the following sequence $\{w1, w1, w0, r0\}$ is applied. Write-time is three time more than read-time, which is same as that needed for testing a $SA1$ fault.

***Slow-write-0** ($SW0$)*: The following sequence can be applied to test a $SW0$ fault: $\{w1, w0, r0\}$. In this case, write-time is two times larger than read-time.

***Slow-write-1** ($SW1$)*: Applied sequence is $\{w0, w1, r1\}$ with write-time two times larger than read-time.

For performing concurrent write operations, test-time entirely depends on the number of memristors that are read simultaneously. Table-I illustrates the number of memristors being read concurrently for the faults mentioned in fault model in an $n \times n$ crossbar.

TABLE-I : Read-time for different faults depending on l_p.
[l_p depends on memristor parameters, fault types, applied voltage and sense amplifier.]

Fault	**Proposed Method**		
	$l_p \geq n-1$	$l_p < n-1$	
		n is odd	n is even
$SA1$	$(n-1)$	$(n-1)$	$(n-1)$
$SA0$	$(n-1)$	$\lceil \frac{(n-1)(n-1)}{l_p} \rceil$	$\lceil \frac{(n-2)(n-1)}{l_p} \rceil + \lceil \frac{2(n-1)}{l_p} \rceil$
$D1$	$(n-1)$	$(n-1)$	$(n-1)$
$D0$	$(n-1)$	$\lceil \frac{(n-1)(n-1)}{l_p} \rceil$	$\lceil \frac{(n-2)(n-1)}{l_p} \rceil + \lceil \frac{2(n-1)}{l_p} \rceil$
$SW0$	$(n-1)$	$(n-1)$	$(n-1)$
$SW1$	$(n-1)$	$\lceil \frac{(n-1)(n-1)}{l_p} \rceil$	$\lceil \frac{(n-2)(n-1)}{l_p} \rceil + \lceil \frac{2(n-1)}{l_p} \rceil$

In all earlier work [9], [8], [14], test time varies quadratically with n. While read-time is constant, write-time becomes quadratic in [17]. On the other hand, in the proposed scheme, the total test-time (including both read- and write-time) grows linearly with n as shown in Table-II.

TABLE-II : Test-time comparison between TOSSP and other existing methods; improvement over March-MoM [14].
[NT = faults not covered by the test technique.]

Fault	**March Algo.** [9]	[8]	**March-MoM** [14]	**TOSSP** (Proposed)	**Test Time Improvement in %** $\frac{\lvert(March-MoM)-(TOSSP)\rvert}{(March-MoM)} \times 100$	
					8×8 crossbar	64×64 crossbar
SA0	$2n^2$	$2n^2$	$n^2 + n$	$(n-1) + (n-1)$	80.55	96.97
SA1	$2n^2$	$2n^2$	$n^2 + \frac{n^2}{15}$	$(n-1) + (n-1)$	80.55	97.11
D0	NT	NT	$3n^2 + \frac{n^2}{5}$	$3(n-1) + (n-1)$	86.32	98.07
D1	NT	NT	$3n^2 + \frac{n^2}{5}$	$3(n-1) + (n-1)$	86.32	98.07
SW0	NT	$4n^2$	$2n^2 + \frac{n^2}{5}$	$2(n-1) + (n-1)$	85.08	97.90
SW1	NT	$4n^2$	$2n^2 + \frac{n^2}{5}$	$2(n-1) + (n-1)$	85.08	97.90

B. Simulation

Simulation is performed in LTspice using the memristor model as given in [18] with the following memristor parameters: R_{off} = 200kΩ, R_{on} = 100Ω, R_{init} = 120kΩ, D = 10N, p = 2, I_{th} = 0.120μA (minimum change in output current to detect a fault). The parameters used in the plot denote the following currents through sensitized sneak-path.

- $I_{Sp} \rightarrow$ Read current of a fault-free sneak-path of length p.
- $I_{Pkp} \rightarrow$ Read current of k parallel fault-free sneak-paths each of length p.
- $I_{FqSp} \rightarrow$ Read current of a sneak-path of length p in the presence of q faults.
- $I_{FqPk} \rightarrow$ Read current of k parallel sneak-paths in the presence of q faults.
- $\Delta_{FSq} \rightarrow$ Difference in magnitude of read current in the presence of q faults.
- $\Delta_{FPq} \rightarrow$ Difference in magnitude of read current in the presence of q faults in the parallel paths.

SA1 Detection: Our results are shown in Fig. 11. Note that for single $SA1$ faults, Δ_{FP1} = 0.83 μA ($\geq$0.12μA),

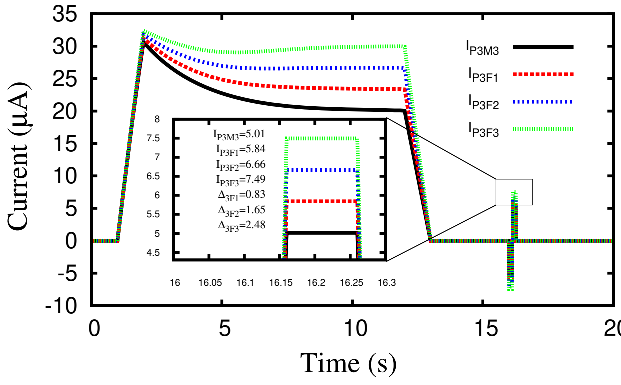

Fig. 11: $SA1$ detection for single and multiple faults present in any of the 3 parallel paths each of length 3.

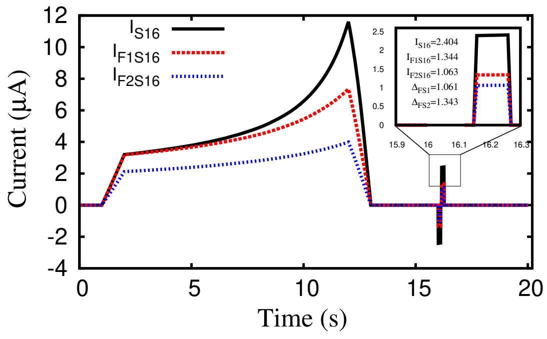

Fig. 12: Testing multiple $SA0$ faults in a 16×16 crossbar.

and for double and triple $SA1$ faults Δ_{FP2} = 1.65 μA and Δ_{FP3} = 2.48 μA, respectively. Now $I_{th} \leq \Delta_{FP1} \leq \Delta_{FP2} \leq \Delta_{FP3}$ leads to the conclusion that as the number of faults increases, the difference in read-current increases accordingly; this guarantees the detection of multiple $SA1$ faults present in parallel sneak-paths.

SA0 Detection: Fig. 12, Fig. 13, and Fig. 14, depict simulation results for testing $SA0$ faults in 16×16, 32×32 and 64×64 crossbar architectures, respectively. In all three cases, $I_{th} \leq \Delta_{FS1} \leq \Delta_{FS2}$, and this guarantees detection of all multiple $SA0$ faults in the array.

V. Conclusions

We address the problem of fault detection in a 2D memristor-array under various fault models such as multiple stuck-at-0/1, deep-0/1, and slow-write-0/1 faults. A formulation based on graph-theoretic representation of sneak-paths is presented, and test generation is then abstracted as an optimal path-covering problem subject to certain constraints on differentiable sensing current. We theoretically show that the proposed method reduces test time by an order of magnitude in terms of crossbar size compared to previous approaches. Simulation results are presented to demonstrate the efficacy of the technique. As a future problem, one may study how this method can be used for the diagnosis of memristor faults and for testing irregular-pattern memristor ensembles.

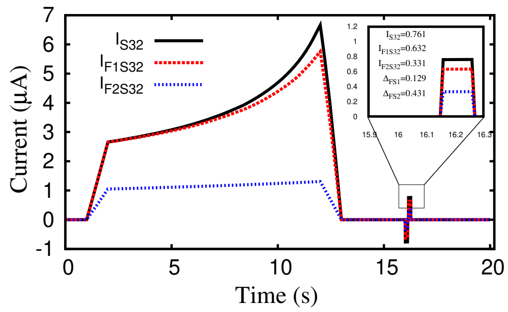

Fig. 13: Testing multiple $SA0$ faults in a 32×32 crossbar.

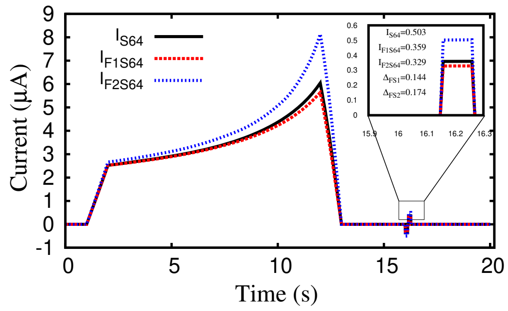

Fig. 14: Testing multiple $SA0$ faults in a 64×64 crossbar.

References

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

[2] D. R. S. Dmitri B. Strukov, Gregory S. Snider and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[3] D. B. Strukov and K. K. Likharev, "Prospects for terabit-scale nanoelectronic memories," *Nanotechnology*, vol. 16, no. 1, p. 137, 2005.

[4] G. S. Snider and R. S. Williams, "Nano/cmos architectures using a field-programmable nanowire interconnect," *Nanotechnology*, vol. 18, no. 3, p. 035204, 2007.

[5] A. Dehon, "Nanowire-based programmable architectures," *J. Emerg. Technol. Comput. Syst.*, vol. 1, no. 2, pp. 109–162, 2005.

[6] N. Zheng and P. Mazumder, "Hardware-friendly actor-critic reinforcement learning through modulation of spike-timing-dependent plasticity," *IEEE Trans. on Computers*, vol. 66, no. 2, pp. 299–311, 2017.

[7] D. Chakraborty and S. K. Jha, "Automated synthesis of compact crossbars for sneak-path based in-memory computing," in *Proc. DATE*, 2017, pp. 770–775.

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

[9] M. L. Bushnell and V. D. Agarwal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits.* Springer, 2000.

[10] C. Y. Chen, H. C. Shih, C. W. Wu, C. H. Lin, P. F. Chiu, S. S. Sheu, and F. T. Chen, "RRAM defect modeling and failure analysis based on march test and a novel squeeze-search scheme," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 180–190, 2015.

[11] S. Hamdioui, M. Taouil, and N. Z. Haron, "Testing open defects in memristor-based memories," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 247–259, 2015.

[12] S. Kannan, J. Rajendran, R. Karri, and O. Sinanoglu, "Sneak-path testing of memristor-based memories," in *Proc. 26th International Conference on VLSI Design and Embedded Systems*, 2013, pp. 386–391.

[13] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, "Modeling, detection, and diagnosis of faults in multilevel memristor memories," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 822–834, 2015.

[14] S. Kannan, J. Rajendran, R. Karri, and O. Sinanoglu, "Sneak-path testing of crossbar-based nonvolatile random access memories," *IEEE Transactions on Nanotechnology*, pp. 413–426, 2013.

[15] O. Ginez, J. M. Portal, and C. Muller, "Design and test challenges in resistive switching RAM (ReRAM): An electrical model for defect injections," in *Proc. 14th IEEE European Test Symposium*, 2009, pp. 61–66.

[16] J. Bondy and U. Murty, *Graph Theory*. Springer Publishing Company, 2008.

[17] L. Sun, N. Zheng, T. Zhang, and P. Mazumder, "Fault modeling and parallel testing for 1T1M memory array," *IEEE Transactions on Nanotechnology*, vol. 17, no. 3, pp. 437–451, May 2018.

[18] Z. Biolek, D. Biolek, and V. Biolkov, "Spice model of memristor with nonlinear dopant drift," *Radioengineering*, pp. 210–214, 2009.