Modeling, Detection, and Diagnosis of Faults in Multilevel Memristor Memories

Sachhidh Kannan, *Member, IEEE,* Naghmeh Karimi, *Member, IEEE,* Ramesh Karri, *Senior Member, IEEE,* and Ozgur Sinanoglu, *Member, IEEE*

***Abstract*—Memristors are an attractive option for use in future memory architectures but are prone to high defect densities due to the nondeterministic nature of nanoscale fabrication. Several works discuss memristor fault models and testing. However, none of them considers the memristor as a multilevel cell (MLC). The ability of memristors to function as an MLC allows for extremely dense, low-power memories. Using a memristor as an MLC introduces fault mechanisms that cannot occur in typical two-level memory cells. In this paper, we develop fault models for memristor-based MLC crossbars. The typical approach to testing a memory subsystem entails testing one memory cell at a time. However, this testing strategy is time consuming and does not scale for dense, memristor memories. We propose an efficient testing technique that exploits sneak-paths inherent in crossbar memories to test several memory cells simultaneously. In this paper, we integrate solutions for detecting and locating faults in memristors. We develop a power aware built-in self-test solution to detect these faults. We also propose a hybrid diagnosis scheme that uses a combination of sneak-path and March testing to reduce diagnosis time. The proposed schemes enable and leverage sneak-paths during fault detection and diagnosis modes, while disabling sneak-paths during normal operation. The proposed hybrid scheme reduces fault detection and diagnosis time by 24.69% and 28%, respectively, compared to traditional March tests.**

***Index Terms*—Built-in tests, fault diagnosis, memristors, multilevel memory, sneak-paths.**

I. Introduction

THE INTERNATIONAL Technology Roadmap for Semiconductors highlights that the performance characteristics of emerging resistive random access memory (RRAM) and phase change memory technologies at the advanced technology nodes (<65 nm), may be quite promising and even superior to the current static-CMOS RAM (SRAM) technology [1]. Metal-oxide memristors [2], [3], a type of RRAM, are a promising candidate for next-generation high-performance, high-density storage technology due to their nonvolatility, scalability and low-power consumption [4], [5]. Memristor are also used as multilevel cells (MLCs) allowing $\sim 2-3\times$ denser memories [1], [6].

Notwithstanding these advantages, nanoscale memristor devices are prone to defects. To screen memristor memories, a stuck-at fault model was proposed in [7]. This fault model was extended to include open, short, and bridging faults [8]. A new fault model, where a memristor can hold either an undefined (X) or logics 1 or 0 state has also been proposed [8]. To test these fault models a design-for-test (DfT) circuit based on a variable timing sensitive circuit has been proposed in [9]. A physics-based analysis of the memristor physical structure and fault models that represents all possible defects in memristors have been proposed in [10]. However, all these approaches consider memristor cells capable of storing a single bit. All these approaches (except [10]) use the conventional March memory test [11]. A March test detects faults by applying a fixed pattern of read and write operations to each memory cell. However, cell-by-cell checking is time-consuming for large/dense memories. We developed sneak-path testing in which sneak-paths (unintended and undesirable electrical paths within a circuit) present in the crossbar are exploited to test multiple memristors simultaneously [10], [12], while targeting single-bit (two-level) cells. We extend this technique to test MLCs.

The contributions of this paper are as follows.

1) We develop a set of fault models to model the defects in a multilevel memristor. (Kannan *et al.* [10], [12] performed a similar study on defects in two-level cells, while MLCs introduce unique failure mechanisms.)
2) We also consider faults that model defects in the crossbar structure, providing a comprehensive fault model for multilevel memristor crossbars.
3) We then modify the power-aware sneak-path testing [10], [12] scheme and extend it to detect faults in multilevel memories. This not only involves the derivation of fault detection sequences (March test), but also their adaptation into a parallel sneak-path based framework.
4) We develop a built-in self-test (BIST) architecture to perform sneak-path testing for MLCs.
5) We use the sneak-path diagnosis in [13], which targets two-level cells, and adapt it to perform diagnosis (determine the location of faults and type of defects) in multilevel memories.

Manuscript received May 13, 2014; revised September 1, 2014; accepted December 27, 2014. Date of publication January 21, 2015; date of current version April 17, 2015. This paper was recommended by Associate Editor D. Chen.

S. Kannan was with the Polytechnic Institute of New York University, Brooklyn, NY 11201 USA (e-mail: skanna01@students.poly.edu).

N. Karimi is with Rutgers University, Piscataway, NJ 08854 USA.

R. Karri is with the Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, NY 11201 USA.

O. Sinanoglu is with the Department of Engineering, New York University Abu Dhabi, Abu Dhabi 129188, UAE.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2015.2394434

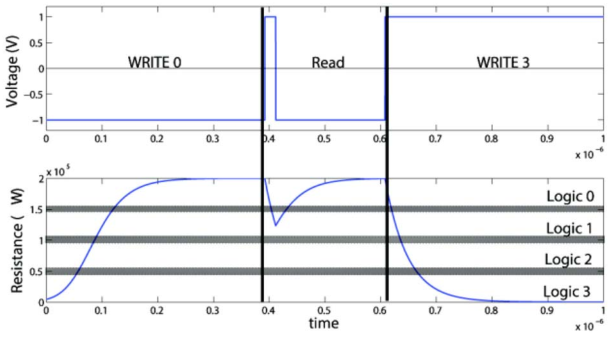


Fig. 1. Memristor read and write operation in a four-level memristor. The resistive range ($R_{\max}$ to $R_{\min}$) is divided into four equal discrete regions for each logic level. The gray region represents undefined region.

6) We provide a brief summary of techniques that repair faults in a multilevel memory. These techniques are entirely new and unsuitable for use with two-level memories.

This paper is structured as follows. First, in Section II, we review the memristor and how we model this device. We describe memristor defect mechanisms and then develop fault models that represent each defect. Section III introduces the crossbar memory architecture and shows how we leverage sneak-paths to test the memristor crossbar in an expedited manner. Section IV describes our fault detection scheme and Section V describes our proposed BIST architecture used for fault detection. Section VI demonstrates the fault diagnosis and repair methodologies, respectively. Section VII provides an analysis of the proposed test and diagnosis techniques. Section VIII summarizes and concludes this paper.

II. Memristor Defects and Faults

A. *Multilevel Memristor Cell*

The metal-oxide memristor is a two-terminal passive element in which the resistance of the device is determined by the voltage or current applied across the device as a function of time. In this paper, we consider the TiO_x metal-oxide memristor device, developed by HP Laboratories [3]. MLC nonvolatile memories allow a single memory cell to store multiple bits of data, exponentially increasing the memory density. A memristor can be programmed to any resistance level between a maximum possible resistance ($R_{\max}$) and minimum resistance ($R_{\min}$). The range of resistance in each cell can be quantized into q discrete levels, which can store $\log_2 q$ bits. In this paper, without loss of generality, we assume that the memristor is quantized with $q = 4$ (as shown in Fig. 1). Each logic level represents two bits of data 00, 01, 10, and 11. We follow a compact notation where 0–3 indices are used to denote logic levels. While a new analysis and notation are required for memristor quantizations with $q > 4$, the analysis in this paper can be easily extended for any quantization value.

The logic level of a memristor can be determined by using a sense amplifier that measures the current flowing through a memristor relative to known reference currents (I_{ref}). In memristor-based memories a current sensing scheme is faster than voltage sensing schemes [14]. However, if the current flowing through the device lies too close to I_{ref}, due to additive noise in the circuit, the output of the sense amplifier may show an erroneous value [9]. Hence, to prevent the effects of noise, a safety margin (or undefined region) is added to ensure the reliability of the memristor. The undefined region for a memristor is shown in Fig. 1.

1) Write Operation: Applying a negatively biased voltage ($v(t) < 0$) across the memristor, gradually increases the overall resistance of the memristor to R_{ma}. Conversely, applying a positively biased voltage ($v(t) > 0$) across the memristor reverses the above process, reducing the overall resistance of the memristor to R_{min}. Intermediate resistances are obtained by either reducing the pulse width or reducing the voltage. In this paper, we use varying pulse widths to program the intermediate resistance levels.

2) Read Operation: To read the logic stored in a memristor, a positive pulse followed immediately with a negative pulse, with the magnitude and duration adjusted to create a zero net change in resistance, is used [4].

B. *Memristor Model*

The memristor is modeled using the TEAM model [15], which is based on the Simmons tunneling effect. The model provides us with an asymmetric switching behavior and nonlinear I–V characteristics. The I–V relationship in this model is given by

$$i(t) = v(t) \cdot R_{\text{on}} e^{-(\lambda / x_{\text{off}} - x_{\text{on}})(x - x_{\text{on}})}$$
$$\text{Window: } f(w) = 1 - (w/D - stp(-i))^{2p} \tag{1}$$

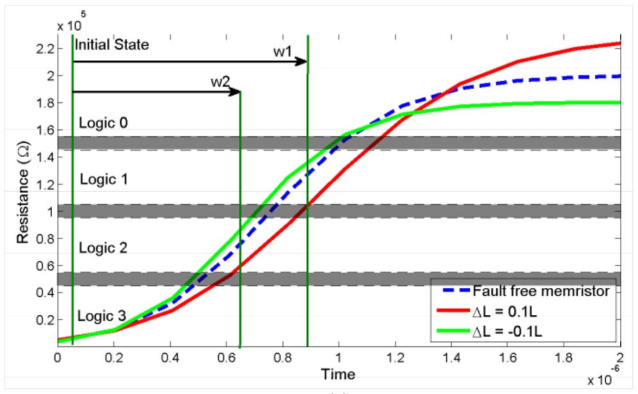
where w, D, p, λ, x_{on}, and x_{off} are experimental fitting parameters. The window function is based on the function used in [15] which introduces the bounds of the device and adds nonlinear behavior close to these bounds.

C. *Memristor Defects and Fault Models*

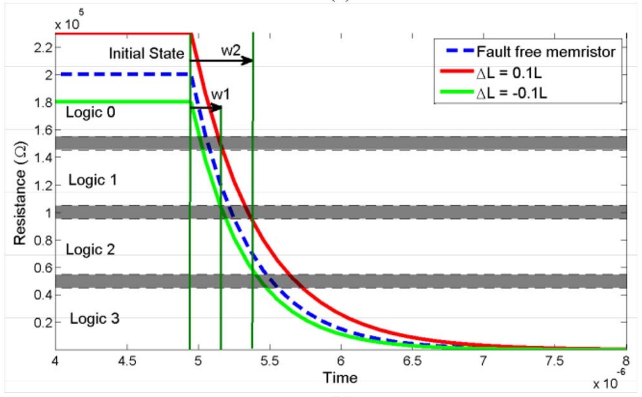
We model a memristor based on the equations described in (1). We then introduce different types of physical defects, such as variation in length, width, and doping concentration and study their effect on the functionality of a memristor. We also study the effect of introducing defects into the crossbar structure. We model these defects as faults and determine the appropriate test sequence that can be used to sensitize and detect these faults.

1) Variation in Length and Cross-Sectional Area: Let us assume that we have a memristor with length L and cross-sectional area A. Any variation in the length (ΔL) causes a change in resistance $\Delta R = \rho(\Delta L/A)$, where ρ is the resistivity of the memristor. Any variation in the cross-sectional area (ΔA) of the memristor causes the resistance to shift by $\Delta R = \rho[(L\ \Delta\ A)/(A(A - \Delta A))]$.

The change in the minimum resistance (ΔR_{on}) and maximum resistance of the device (ΔR_{off}) is evaluated using the resistivity of the doped and the undoped region of the memristor. The maximum and minimum resistance increase to $R_{\text{off}} + \Delta R_{\text{off}}$ and $R_{\text{on}} + \Delta R_{\text{on}}$. This results in three unique



(a)



(b)

Fig. 2. (a) $3w1$ and $3w2$ operation on a memristor with a defective ±10% change in length. A slow-write fault occurs when the write pulse is not long enough to switch the faulty device into the logic 1 state. (b) $0w1$ and $0w2$ operation on a memristor with a defective ±10% change in length. This time, a slow-write fault occurs when the write pulse is not long enough to switch the faulty device into the logics 1 or 2. The gray region represents the undefined region.

faults: 1) the slow-write fault; 2) fast-write fault; and 3) the deep fault.

a) Slow-write fault: In Fig. 2(a), the dotted line represents a fault-free memristor and the red line represents a defective memristor with a 10% increase in length. The device is initially at logic 3. When a w1 (write logic 1) pulse is applied, the fault-free memristor switches to logic 1. A faulty memristor transitions slower between states, resulting in the memristor to remain in logic 2 or in the undefined region. This type of fault is a slow-write-3-1 (SW3-1) fault. A SW3-1 is represented by $\langle\{3w1\}/X_{21}\rangle$; i.e., a write 1 operation is performed while the device is currently at logic 3, sensitizing the fault and resulting in a faulty response or X_{21} which represents the undefined region between logics 2 and 1. A similar study of the memristor with different initial logic levels also shows a possible SW2-0 fault, represented by $\langle\{2w0\}/X_{10}\rangle$.

The defects considered in this paper have a maximum defect margin of 10%. For example, if a defect-free memristor has length L, a defective memristor has a maximum possible length of 1.1 L. Defects with a larger variation are extremely unlikely. Hence, the memristor model we are considering suggests a possibility of only a SW3-1 and a SW2-0 fault.

Writing from a high resistance state to a low resistance state is more susceptible to a slow-write fault. As we can see in Fig. 2(b), our memristor is susceptible to a SW0-1 and a SW0-2 fault. A SW0-1 is represented by $\langle\{0w1\}/X_{01}\rangle$ and a SW0-2 is modeled as $\langle\{0w2\}/X_{12}\rangle$. The memristor also demonstrates a SW1-2 (not shown in figure) fault that is represented by $\langle\{1w2\}/X_{12}\rangle$.

b) Fast-write fault: As we can see in Fig. 2(b), the dotted line represents a fault-free memristor and the green line represents a defective memristor with a 10% decrease in length. The device is initially at logic 0. When a w1 pulse is applied to the fault-free memristor, it switches to logic 1, but the faulty memristor transitions faster between states and results in the memristor to switch to logic 2. We refer to this type of fault as a fast-write-0-1 (FW0-1) fault. A FW0-1 can be represented by $\langle\{0w1\}/X_{21}\rangle$.

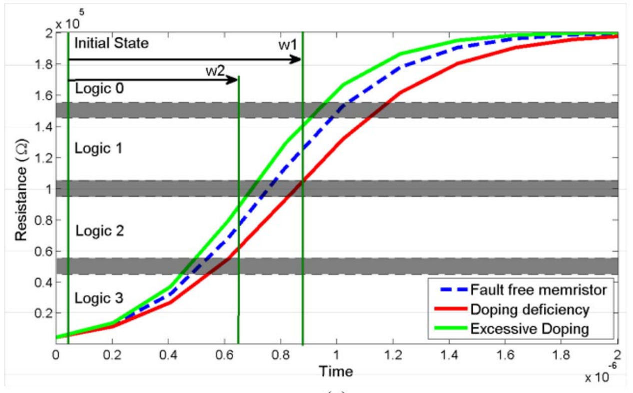
c) Deep fault: An increase in the length (ΔL) or a decrease in the cross-sectional area (ΔA) results in an upward shift in the upper and lower resistance limits of the memristor by $R_{\text{off}} + \Delta R_{\text{off}}$ and $R_{\text{on}} + \Delta R_{\text{on}}$, respectively. The memristor enters a "deep-0" state when its resistance $R > R_{\text{off}}$ [12]. When we attempt to write a higher logic level, say logic 2, to a memristor that is in the deep-0 state, the duration of the write pulse is too short to switch from deep-0 to logic 2. This type of fault is called a deep-0 fault. A deep-0 can be sensitized using a sequence of operations and is represented as $\langle\{w0,\ w0,\ w1\}/X_{01}\rangle$; a series of operations {w0, w0, w1} are required to sensitize the fault.

d) Deep-(q-1)-fault: This fault occurs in a memristor with q quantization levels, due to a decrease in the length (ΔL) or an increase in the cross-sectional area (ΔA) of the memristor. This causes the range of resistance to decrease to $R_{\text{on}} - \Delta R_{\text{on}}$ and $R_{\text{off}} - \Delta R_{\text{off}}$. When a memristor with $q = 4$ enters the "deep-3" state (i.e., $R < R_{\text{on}}$) the write pulse is not long enough to switch the state from deep-1 to logic 0. A deep-3 fault can be sensitized by a sequence of operations denoted by $\{w3,\ w3,\ w2\}$. The fault is represented by $\langle\{\text{w3, w3, w2}\}/X_{32}\rangle$.

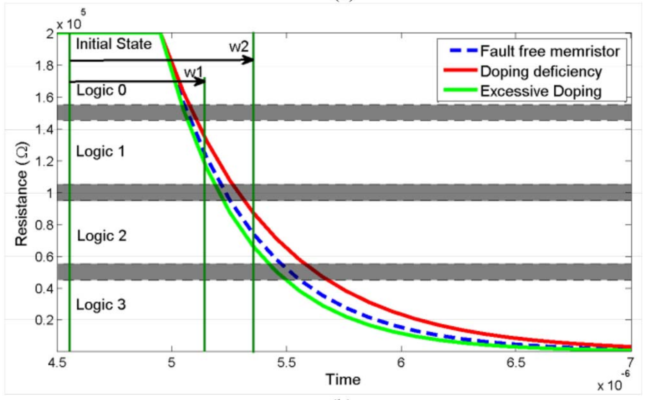
2) Variation in Doping: The variation in doping density affects the mobility of the charge carries in the memristor. As we can see from Fig. 3(a), an excess in doping results in an increased rate at which the memristor transitions from one state to another. Similarly, a deficiency in doping [Fig. 3(b)] results in a decreased rate at which the memristor transitions from one state to another. A completely undoped memristor is permanently at resistance R_{max} and is modeled as a stuck-at 0 fault.

3) Open/Short Defects: Open and short defects in the memory circuits such as broken or missing metal wires, extra metal lines, etc. can be modeled at the electrical level using a resistor [9]. An open is an unintended series resistance R_{open} where, $0 < R_{\text{open}} < \infty\Omega$. A short is an unintended resistive path R_{short} between any part of the circuit and a supplied voltage V_{dd} or ground. The short resistance can be in the range of $0 < R_{\text{short}} < \infty\Omega$. An open or a short defect can be modeled as a stuck-at fault.

a) Stuck-at 0: A short to ground at either terminal of the memristor will result in the memristor being stuck-at logic 0



(a)



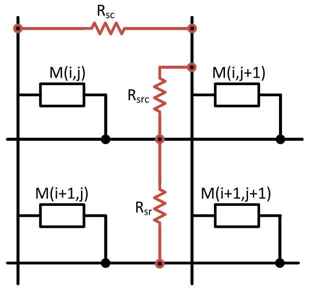
(b)

Fig. 3. (a) $3w1$ and $3w2$ operation for a memristor for $\pm 10\%$ change in doping concentration. A slow-write-3-1 (SW3-1) fault occurs when the transition from a low resistance state to a high resistance state is slower than normal. (b) $0w1$ and $0w2$ operation for a memristor with a $\pm 10\%$ change in doping concentration. A slow-write-0-1 (SW0-1) fault occurs when the transition from a high resistance state to a low resistance state is slower than normal.

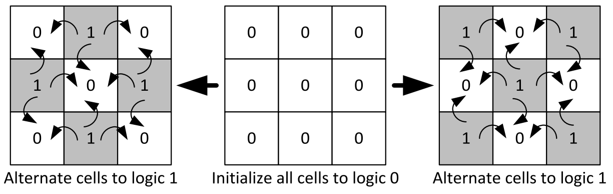
irrespective of the voltage applied across it. This fault is called a stuck-at 0 (SA0) fault and represented by $\langle wL/0\rangle$; a logic L is the expected output of a fault-free memristor (where $L = 2, 3$). A $\langle w1/0\rangle$ falls short to unambiguously detect a SA0 since a logic 0 output may also occur if the device resistance is stuck in X_{01} resulting in a false positive fault detection. To detect a SA0, we should use a write operation to a logic level that is at least two logic levels apart.

b) Stuck-at L: The defect represented by a stuck-at L (SAL) fault, where L is any logic level in 1–3, is slightly different from a stuck-at 0 fault. An open defect in series with the memristor results in the memristor to remain stuck-at any logic level independent of any voltage applied across the device. The fault can be sensitized by any write operation and detected by reading for the stuck-at logic level L. However, there is a chance that the stuck-at value may be of a resistance that lies in the undefined region. Hence, to detect a *SAL* fault we use $\langle w(L \pm 2)/L\rangle$. For instance a stuck-at-2 fault can be detected by a $\langle w0/2\rangle$.

4) Bridges Between Rows/Columns: This defect arises from the structure of the crossbar rather than the memristor. When adjacent rows or columns are bridged together [R_{sc} and R_{sr} in Fig. 4(a)] a transition in one cell may result in a similar transition in the adjacent cell. This behavior is modeled as a coupling fault.



(a)


Alternate cells to logic 1
Initialize all cells to logic 0
Alternate cells to logic 1

(b)

Fig. 4. (a) Coupling faults occurring due to resistive bridge defects (R_{sc}, R_{sr}, R_{src}). (b) Two tests required to detect all coupling faults. Gray cells are aggressor cells and white ones are victim cells. Arrows indicate aggressor/victim pairs that may be detected. With an initial state as logic 0, alternate cells are programmed to a logic 1. Any change in the victim cells indicate a fault.

a) Coupling fault: The memory element that is addressed is the aggressor cell and the victim cell is a vertically or horizontally adjacent cell. The fault can either be written as $\langle Xw1; 0/1\rangle$ (with any initial state, a w1 triggers a change from logic 0 to logic 1 in the adjacent element) or $\langle Xw0; 1/0\rangle$.

As shown in Fig. 4(b), in order to sensitize and detect coupling faults we perform $\langle 0w1\rangle$ on alternate memory cells. Any change in the other set of memory cells indicates a coupling fault. As shown in [16], the resistive bridge R_{src} does not result in a fault.

Table I summarizes the defects in a memristor caused during fabrication and the associated fault models. Defects that cause an undefined output, such as a deep or slow-write, may possibly be undetected by a standard March test since the faults cause a random logic value to be read from the defective memristive cell. To detect these faults, we propose a testing scheme (described in Section III) that provides two capabilities.

1) The ability to distinguish between a logic level and an undefined state.
2) Improved test time and minimized cost by using sneak-paths to test multiple memristors simultaneously.

III. Leveraging Sneak-Paths for Testing

A. Crossbar Memory

Nanoscale memories are constructed using dense nanoscale crossbar [17]. The crossbar array consists of two sets of nanowires running perpendicular to one another. There is a memristor cell at the intersection of each pair of perpendicular nanowires. One set of parallel wires are used as bit-lines and the second set of wires, orthogonal to the first, are used as word lines.

Due to the bidirectionality of metal-oxide memristors, crossbar architectures suffer from sneak-paths. Sneak-paths are

TABLE I
CORRESPONDENCE BETWEEN DEFECTS AND FAULT TYPES

Fault Type	Defect type/source			
	Doping	Length	Area	Others
Ideal	Normal	L	A	-
SA0	-	-	-	Short [9]
	Undoped	L	A	-
SAL	-	-	-	Open [9]
SW	Under doped	L	A	-
	Normal	L + ΔL	A	-
		L	A − ΔA	-
		L + ΔL	A − ΔA	-
FW	Excessive doping	L	A	-
	Normal	L − ΔL	A	-
		L	A + ΔA	-
		L − ΔL	A + ΔA	-
Deep-0	Normal	L + ΔL	A	-
		L	A − ΔA	
		L + ΔL	A − ΔA	
Deep-(q-1)	Normal	L − ΔL	A	-
		L	A + ΔA	
		L − ΔL	A + ΔA	
Coupling	Normal	L	A	Bridged rows/ columns [9]

unintended and undesirable electrical paths within a circuit that may corrupt the output current, resulting in incorrect read and write operations [18]. Several techniques to eliminate sneak-paths have been proposed [19], [20].

B. Testing Methodology

Most memory testing techniques, including the methods discussed in [7]–[9] use the March testing [11]. March exhaustively tests a memory for a number of faults, one memory cell at a time. However, when applied to large memristor memories, this technique leads to prolonged test times. Other techniques such as [21] use a divide-and-conquer approach that works extremely well for crossbars with a small number of faults ($<5 \sim 6$ faults/crossbar), but test time increases exponentially with increased number of faults.

We propose a "sneak-path testing" technique for multi-level memories. In every memristor cell, crossbar current and voltage are the only two measurable quantities. We leverage sneak-paths to measure the accumulated sneak currents, capturing information about multiple memristors in a single measurement.

Fig. 6 illustrates the currents flowing through a 2×2 crossbar when the memory cell M_{12} (memristor in row 1, column 2) is accessed. The output current (I_{output}) at the second column is the sum of two parallel currents, the primary current (I_{primary}) and the sneak-path current (I_{sneak}). The output current of a simple 2×2 crossbar can be given as

$$I_{\text{output}} = \frac{V_{pu}}{M_{12} || (M_{11}+M_{21}+M_{22})}. \tag{2}$$

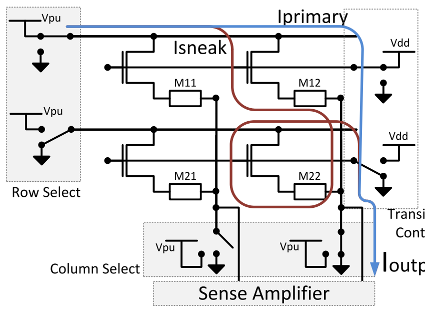


Fig. 5. Sneak-path testing in a 2×2 crossbar. M_{12} is the addressed memory cell. A sneak-path current I_{sneak} flows through M_{11}, M_{21}, and M_{22}. Any variations due to defects in the memristors are reflected as a measurable variation in I_{output}.

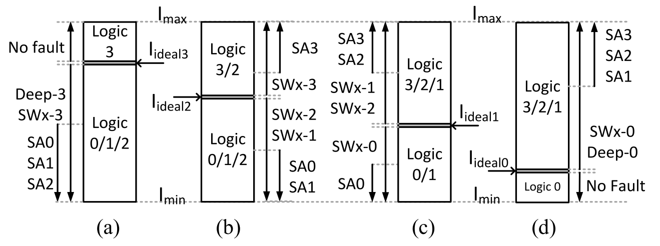


Fig. 6. Output logic of sense amplifier with reference current (a) $I_{\text{ideal}3}$, (b) $I_{\text{ideal}2}$, (c) $I_{\text{ideal}1}$, and (d) $I_{\text{ideal}0}$.

Any variation in the resistance (due to defects) in M_{12}, M_{11}, M_{21}, or M_{22} results in a variation of I_{output}. Hence, by simply measuring the output current while reading a single memory cell, sneak-paths help us capture information about multiple memristors in a single step.

In order to enable the proposed sneak-path testing, we need to modify the crossbar. During typical read/write operation only transistors on the addressed row are turned on elimination sneak-paths. During test phase, we introduce sneak-paths by turning on transistors on all rows. To implement this we modify the transistor control circuitry as shown in [12].

Comparing the actual output current I_{actual} with the ideal current I_{ideal} (output current of a defect free crossbar) we can detect defective memristor(s). The variation in I_{output} diminishes for faults that are further away from the addressed cell. A group of memristors that are tested simultaneously is referred to as the "region of detection" (RoD).

Each memristor with q quantization level has q values of ideal current given by $I_{\text{ideal}L}$; where $0 \leq L \leq q-1$. Based on [22], we determine the minimum noise margin of the sense amplifier to be around 0.12 μA. Hence, to detect a fault, the variation in output current needs to be greater than 0.12 μA. The values of $I_{\text{ideal}L}$ are the reference currents to the sense amplifier during the test phase. As illustrated in Fig. 5, this ensures that faults are detected for any faulty resistance.

IV. Fault Detection

A memory test consists of a sequence of operations that are all applied to a given memory cell before proceeding to the next cell (increasing "$\Uparrow$," or decreasing "$\Downarrow$" address sequence; "$\Updownarrow$" represents either $\Uparrow$, or $\Downarrow$). Operations on memory cells are

TABLE II
FAULT SENSITIZATION AND DETECTION SEQUENCES IN MARCH TEST SEQUENCE (3)

Fault	Sensitized by	Detected by
SA0	M2, M4	M5
SAL	M7, M9	M10
SW0-2	{*w0*, *w2*} of M6	{*r2*} of M6
SW0-1	{*w0*,... *w1*} of M6	{*r1*} of M6
SW3-1	M2, M4, {*w1*} of M6	{*r2*} of M6
SW2-0	{*w2*,... *w0*} of M6	{*r0*} of M6
FW0-1	{*w0*,... *w1*} of M6	{*r1*} of M6
Deep-3	{*w3*, *w3*} of M6, M7, M9	M10
Deep-0	M1, M2, M4	M5

"*wL*" (write logic *L*) and "*rL*" (read a cell expecting a logic *L* value), where $0 \le L \le 3$ [11]. We introduce a new way to determine the next cell that needs to be tested denoted by "$\Updownarrow_{a0}$" and "$\Updownarrow_{a1}$." $\Updownarrow_{a0}$ represents a sequence that proceeds to alternate memory elements starting from address location 0. Similarly, $\Updownarrow_{a1}$ is a sequence that performs the test to alternate memory locations starting from address 1. The addressing sequence is unique to each fault type and is explained later in this section. The fault sequence is tabulated in Table III.

To test a memristor memory, we first determine the proper March sequence that can detect all faults defined in Section II-C. Next, we improve the test efficiency by leveraging sneak-paths. Based on the fault models in Section II-C, the proper March sequence to test a memristor memory is

$$\begin{aligned}&\{\text{M1: } \Updownarrow (\text{w0, w0}); \text{ M2: } \Updownarrow_{a0} (\text{w3}); \text{ M3: } \Updownarrow_{a1} (\text{r0});\\ &\text{M4: } \Updownarrow_{a1} (\text{w3}); \text{ M5: } \Updownarrow (\text{r3}); \text{ M6: } \Updownarrow (\text{w1, r1, w0, w2,}\\ &\text{r2, w0, r0, w1, r1, w3, w3}); \text{ M7: } \Updownarrow_{a1} (\text{w0});\\ &\text{M8: } \Updownarrow_{a0} (\text{r3}); \text{ M9: } \Updownarrow_{a0} (\text{w0}); \text{ M10: } \Updownarrow (\text{r0})\}. \qquad (3)\end{aligned}$$

We have labeled each March element with a label, "*Mx*." Table II describes how the proposed algorithm detects the various multilevel memristor faults (shown in Table I).

Testing a $m \times n$ crossbar requires 18 mn memory accesses. In order to minimize test time we determine and exploit the RoD for each fault type to maximize test area.

Using the sneak-path testing described in Section III, we determine the RoD for each type of fault.

A. MLC Fault Types

1) Stuck-at Fault: All stuck-at faults can be detected in two steps. First, write any logic level followed by a read. Second, write any other logic level that is at least two levels away and perform another read. For example, all stuck-at faults can be detected by either by $\{w0, r0, w2, r2\}$ or $\{w1, r1, w3, r3\}$.

Fig. 7(a) illustrates the RoD for the SA1 fault. Each square represents a memory cell. The memory cell at the center of the RoD (red square) is the memory cell under test. Faults in the surrounding cells (yellow squares) are detected by testing the addressed cell and the variation in output current for a fault occurring at that location. $\mathrm{I}_{\mathrm{ideal0}}$ is the expected current at the output. If any fault exists in the RoD the output current is greater than I_{ideal0} and the sense amplifier output is seen as logic 1.

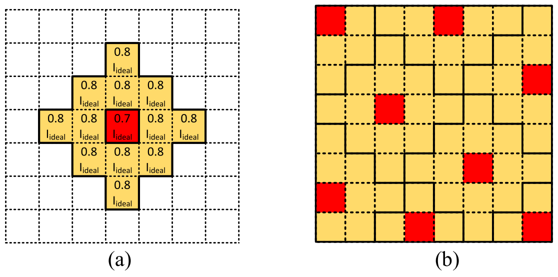


Fig. 7. (a) RoD for stuck-at fault. (b) Test points and coverage for a SA fault by tiling the RoD in a 8×8 crossbar. Red squares are the addressed memory cell. Yellow squares represent memory cells in RoD whose faults can be sensed at output. Number in each cell is the variation in I_{output} as a function of I_{ideal} for a fault at that location.

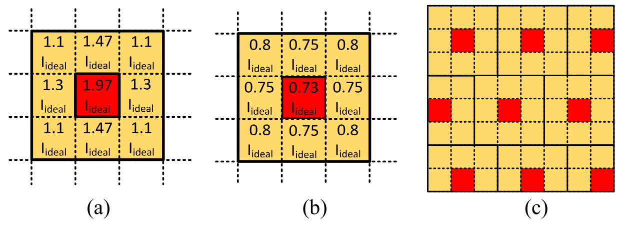


Fig. 8. RoD for a single (a) deep-0 fault using $\mathrm{I}_{\mathrm{ideal0}} = 91.9\ \mu\mathrm{A}$. (b) Deep-3 fault using $\mathrm{I}_{\mathrm{ideal3}} = 36\ \mu\mathrm{A}$ for a 8×8 crossbar. (c) Tiling the deep fault RoD in a 8×8 crossbar. Number in each cell is the variation in I_{output} as a function of I_{ideal} for a fault at that location.

In order to test the entire memory we test specific points on the memory, ensuring that every memory cell falls into the RoD of at least one test point. Hence, we "tile" the RoD to minimize the number of measurements, and thus the test time [Fig. 7(b)]. The order in which the cells need to be tested is denoted by "$\Updownarrow_{\mathrm{SA}}$." We estimate the number of memory accesses to test a $m \times n$ memory as $2\ \mathrm{mn} + (2\mathrm{mn}/13)$.

2) Deep Fault: To detect a "deep-0," we need to force the cell into the deep-0 state using $\{\Updownarrow (w0, w0, w1); \Updownarrow (r1)\}$ with a reference current of I_{ideal1}. The RoD for a deep-0 is shown in Fig. 8(a). Similarly, for $q = 4$, a deep-3 is detected by $\{\Updownarrow (w3, w3, w1); \Updownarrow (r1)\}$ with a reference current of $\mathrm{I}_{\mathrm{ideal3}}$. The RoD for a deep-3 fault is shown in Fig. 8(b). Fig. 8(c) shows how the RoDs can be tiled to minimize the number of test points. The order in which the cells need to be tested is denoted by "$\Updownarrow_{\mathrm{deep}}$." Note that the RoD for a deep fault is a subset of the RoD of a SA fault and the sequence $\Updownarrow_{\mathrm{deep}}$ can be used instead of $\Updownarrow_{\mathrm{SA}}$ if necessary.

3) Slow-Write Fault: SW3-1 and SW0-1 faults are detected by using $\{\Updownarrow(w3, w1); \Updownarrow(r1)\}$ and $\{\Updownarrow (w0, w1); \Updownarrow(r1)\}$, respectively. However, as we see from (4), the tests for deep-0 and deep-3 faults cover these types of faults. The RoD for a SW2-0 and SW0-2 faults shown in Fig. 9(a)–(c) demonstrates how we can tile the RoD within a memory. The order in which the cells need to be tested is denoted by "$\Updownarrow_{\mathrm{SW}}$."

4) Fast-Write Fault: FW0-1 can be detected by using $\{\Updownarrow(w0, w1); \Updownarrow(r1)\}$. We see from (4), the tests for deep-0 and deep-3 faults covers these fault types.

5) Coupling Fault: Coupling faults are detected by using $\{\Updownarrow (w0); \Updownarrow_{a0} (w3); \Updownarrow_{a1} (r0)\}$ and $\{\Updownarrow (w3); \Updownarrow_{a1} (w0); \Updownarrow_{a0} (r3)\}$. The first sequence detects any fault with the

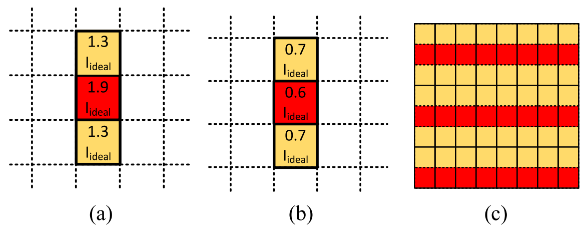


Fig. 9. RoD and variation in output current for a single (a) SW2-0 fault using $I_{\text{ideal0}} = 7.9\ \mu$A. (b) SW0-2 fault using $I_{\text{ideal3}} = 28\ \mu$A for a 8×8 crossbar. (c) Test points and coverage for a SW fault by tiling the RoD. Number in each cell is the variation in I_{output} as a function of I_{ideal} for a fault at that location.

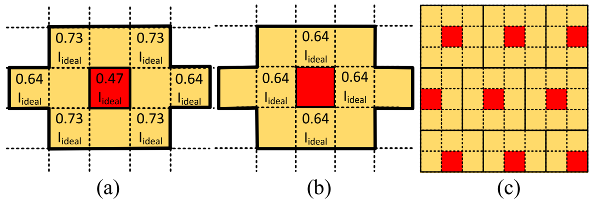


Fig. 10. RoD and variation in output current for a single coupling fault using $I_{\text{ideal0}} = 91.9\ \mu$A using (a) $\Updownarrow_{a0}$ and (b) $\Updownarrow_{a1}$ for a 8×8 crossbar. (c) Tiling the RoD in a 8×8 crossbar. Number in each cell is the variation in I_{output} as a function of I_{ideal} for a fault at that location.

TABLE III
DESCRIPTION OF TEST SEQUENCES

Sequence Notation	Sequence
$\Uparrow$	All cells from smallest to largest address.
$\Downarrow$	All cells from largest to smallest address.
$\Updownarrow$	Either $\Uparrow$ or $\Downarrow$.
$\Updownarrow_{\text{a0}}$	Alternate cells starting from location 0.
$\Updownarrow_{\text{a1}}$	Alternate cells starting from location 1.
$\Updownarrow_{\text{SA}}$	All cells to detect stuck-at faults using RoD.
$\Updownarrow_{\text{SW}}$	All cells to detect slow-write faults using RoD.
$\Updownarrow_{\text{deep}}$	All cells to detect deep faults using RoD.
$\Updownarrow_{\text{ra0}}$	Sequence to detect coupling faults froma0→a1
$\Updownarrow_{\text{ra1}}$	Sequence to detect coupling faults froma1→a0

aggressor cell in memory address $a0$ and victim cell in $a1$. Similarly, the second sequence detects any faults with memory address $a0$ as the aggressor cell. Fig. 10 illustrates the RoD for the coupling fault. The sequence in which the cells need to be tested are denoted by "$\Updownarrow_{ra0}$" or "$\Updownarrow_{ra1}$."

B. Sneak-Path Detection of MLC Faults

To minimize test time, we replace the read operations with sneak-path testing. Modifying (3), we get

$$\begin{aligned}&\{\Updownarrow (w0,\ w0); \Updownarrow_{\text{a0}} (w3); \Updownarrow_{\text{ra1}} (r0); \Updownarrow_{\text{a1}} (w3), \Updownarrow_{\text{deep}} (r3);\\ &\ \Updownarrow (w1); \Updownarrow_{\text{SW}} (r1); \Updownarrow (w0,\ w2); \Updownarrow_{\text{SW}} (r2); \Updownarrow (w0);\\ &\ \Updownarrow_{\text{SW}} (r0); \Updownarrow (w1); \Updownarrow_{\text{SW}} (r1); \Updownarrow (w3,\ w3); \Updownarrow_{\text{a1}} (w0);\\ &\ \Updownarrow_{\text{ra0}} (r3); \Updownarrow_{\text{a0}} (w0); \Updownarrow_{\text{deep}} (r0)\}. \qquad (4)\end{aligned}$$

The sequence notation is explained in Table III. Using the test sequence shown in (4) reduces the total test time to 11 mn writes + (2.56) mn reads, which is a 24.69% reduction in test time compared to the March sequence shown in (3). Table VIII provides the test time for each type of fault and compares them to the March tests in (3).

If there are several faults in a single RoD, the output current moves further away from I_{ideal} making it easier to detect. Faults with opposite effects do not mask each other since they can never be sensitized at the same time.

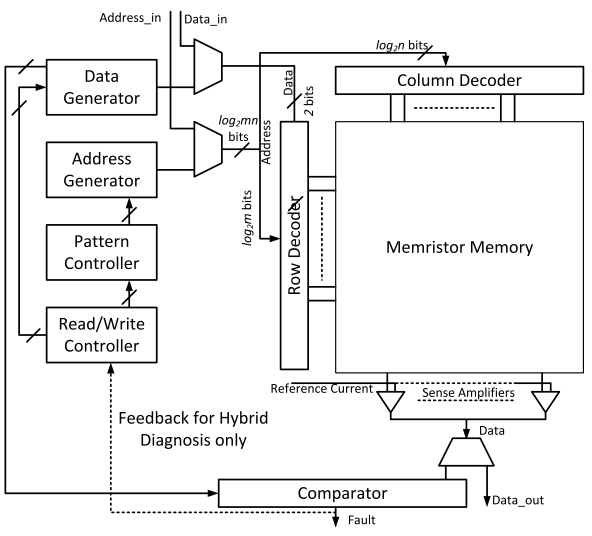


Fig. 11. BIST architecture to implement sneak-path testing.

V. BIST

Embedded memories can be considered as the densest circuitry on a chip, and are expected by the end of 2014 to occupy around 94% of the silicon area in SoCs [23]. Thus, BIST will be more efficient than external testing. The BIST architecture is shown in Fig. 11. The main purpose of the address generator is to generate the address sequences required by sneak-path testing ($\Updownarrow, \Updownarrow_{\text{a0}}, \Updownarrow_{\text{a1}}, \Updownarrow_{\text{deep}}, \Updownarrow_{\text{SW}}$). The pattern controller provides the required control signals for the address generator. The read/write controller stores the test sequence and provides the location on the memory where data is to be written during the write operation and data is to be read during the read operation. The read/write controller determines how many cycles a given address is maintained before the address counter is incremented and which address sequence is required. The data generator is controlled by the read/write controller and provides the correct data to the memory corresponding to the particular element of the particular test pattern. The comparator compares the output of the sense amplifier with the expected output, provided by the data generator, to detect a fault. In order to perform hybrid diagnosis (described in Section VI-A) the output of the comparator forms a feedback loop with the read\write controller allowing us a more dynamic approach to generate the address sequence. Multiplexers are used to toggle between test and normal mode. The area of the BIST is 1260 + 430 mn transistors.

The address generator that is controlled by the pattern controller should be capable of generating the address sequences required by sneak-path testing ($\Updownarrow, \Updownarrow_{a0}, \Updownarrow_{a1}, \Updownarrow_{\text{deep}}, \Updownarrow_{\text{SW}}$). Fig. 12 demonstrates the different control signals utilized

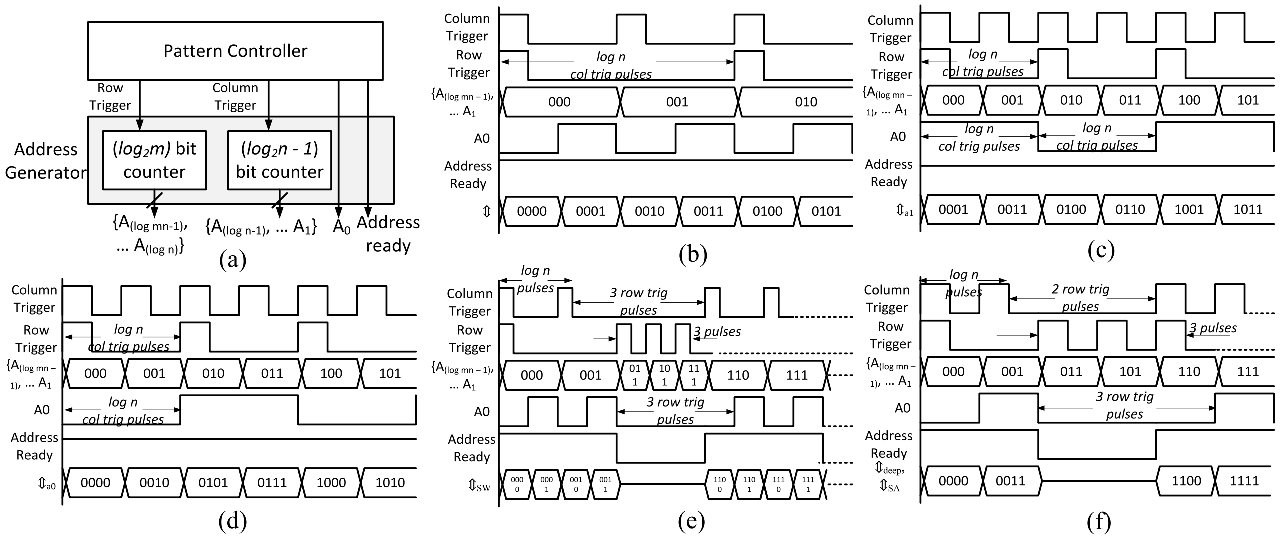


Fig. 12. (a) Address generator structure for a $m \times n$ crossbar. Pattern control signal and address generator output for a 4×4 crossbar (m, n = 4) (b) $\Updownarrow$ sequence, (c) $\Updownarrow_{a1}$ sequence, (d) $\Updownarrow_{a0}$ sequence, (e) $\Updownarrow_{\mathrm{SW}}$ sequence, and (f) $\Updownarrow_{\mathrm{SA}}$, $\Updownarrow_{\mathrm{deep}}$ sequence (RoD of deep fault is a subset of the RoD of a SA fault and $\Updownarrow_{\mathrm{deep}}$ can substitute $\Updownarrow_{\mathrm{SA}}$).

TABLE IV
REDUCTION IN NUMBER OF TRANSITIONS IN ADDRESS SEQUENCE FOR POWER-AWARE TESTING IN A 64 × 64 CROSSBAR AND THE CORRESPONDING AREA OVERHEAD

Sequence	**Sequence Length**	**Transitions**	**Modification**	**Transitions after modification**	**% reduction in transitions**	**Area Overhead**
$\Updownarrow$	mn	16339	Gray Code for row and column address	4096	74.9	7%
$\Updownarrow_{\mathrm{a0}}$, $\Updownarrow_{\mathrm{a1}}$	$mn/2$	6124	-	6124	0	0
$\Updownarrow_{\mathrm{SW}}$	$mn/3$	5052	Gray code for column address	2026	59.9	3.5%
$\Updownarrow_{\mathrm{deep}}$, $\Updownarrow_{\mathrm{SA}}$, $\Updownarrow_{\mathrm{ra0}}$, $\Updownarrow_{\mathrm{ra1}}$	$mn/9$	2116	-	2116	0	0
Test Sequence (Eqn. 4)	$13.56mn$	151202	Gray Code for row and col. depending on sequence	65640	56.58	7%

by the pattern controller to generate the required address sequences.

A. *Power-Aware Test*

Test power is an important issue that should be taken into consideration. It has been shown that in memory testing the primary source of test power is the transitions of the address sequence [24]. Hence, we need to modify (4) to minimize the number of transitions. The $\Updownarrow$ sequence is the most frequently used sequence and also has the largest number of transitions. Instead of using a counter, we may use a gray code generator to minimize the number of transitions in the sequence. Similarly, the $\Updownarrow_{\mathrm{SW}}$sequence may use a gray code generator as well, to generate the column address. The $\Updownarrow$a0, $\Updownarrow$a1, $\Updownarrow_{ra0}$, and $\Updownarrow_{\mathrm{ra1}}$ sequences are used to sensitize and detect coupling faults and are very sensitive to the order in which they are applied, and thus cannot be optimized. Table IV summarizes the address sequences and the reduction in address transitions when a gray code generator is used. As shown in Table III, using a gray code generator instead of a counter for $\Updownarrow$ and $\Updownarrow_{\mathrm{SW}}$ reduces the total number of transitions by 56.58% at an area overhead of 7%.

VI. FAULT DIAGNOSIS AND REPAIR

Faults are either: 1) recoverable or 2) nonrecoverable [13]. A typical repair solution for nonrecoverable faults is to use redundant rows and columns when designing the memory [25]. Columns with defective memristors are bypassed and replaced using the redundant columns. On the other hand, recoverable faults can be repaired using a variety of techniques (Section VI-B). Hence, a diagnostic sequence is required to distinguish between the fault types so that the appropriate repair technique may be applied.

A. *Memory Diagnosis*

In order to repair a fault, we first perform fault diagnosis to determine: 1) fault location and 2) fault type. The March test (3) determines the location of the faulty cell at the same time it detects a fault since it tests one cell at a time, i.e., fault detection and diagnosis are performed simultaneously. However, sneak-path testing (4) detects a fault within the RoD, and the exact location of the fault is unknown. Hence, a separate diagnosis technique is required to determine the location of the fault.

TABLE V
FAULT SENSITIZATION AND DETECTION SEQUENCES IN MARCH DIAGNOSIS SEQUENCE (5)

Fault	**Sensitized by**	**Detected by**
SA0	{*w3*, *w3*} of M6	{*r3*} of M6
SAL	{*w0*, *w0*} of M1	{*r0*} of M1
SW0-2	{*w0*, *r0*, *w2*} of M6	{*r2*} of M6
SW0-1	{*w0*, *r0*, *w1*} of M6	{*r1*} of M6
SW3-1	M2, M4, M5, {*w1*} of M6	{*r1*} of M6
SW2-0	{*w2*, *r2*, *w0*} of M6	{*r0*} of M6
FW0-1	*w0*, *r0*, *w1*} of M6	{*r1*} of M6
Deep-3	{*w3*, *w3*} of M6, M7, M9	M10
Deep-0	M1, M2, M4	M5

1) Diagnosis Using March Sequence: In traditional March testing, a SA1 fault is typically detected by {$w0$, $r0$}. However, if the initial state is logic 1 the $w0$ operation may sensitize either a SA1 or SW1-0 fault. Hence, to distinguish between these faults, the initial state of the cell-under-test is considered. We use {$w0$, $w0$, $r0$} and {$w1$, $r1, w0$, $r0$} to sensitize and detect a SA1 and SW*1-0* fault, respectively. The $r1$ in {$w1$, $r1, w0$, $r0$} is a part of the sensitizing sequence to ensure that the memristor is in logic 1 before applying the $w0$. Similarly, we use {$w3$, $w3$, $r3$} and {$w0$, $w1$, $r1$} to sensitize and detect the SA0, SA1, SA2, and SW0-1 faults.

Table VI provides the test sequences used to distinguish between different types of faults. To diagnose the SW0-2 fault sequence $S7$ can be used. However, to diagnose a deep-0 fault, two test sequences (S2 and S5) are required. A fault is diagnosed as deep-0 if it is detected by sequence S5 but not detected by sequence S2. The last row of the table summarizes the diagnostic sequences that is used for each fault type. The March sequence to diagnose all faults is

$$\begin{aligned}\{&M1\text{: } \Updownarrow (w0, w0, r0);\ M2\text{: } \Updownarrow_{a0} (w3);\ M3\text{: } \Updownarrow_{a1} (r0);\\ &M4\text{: } \Updownarrow_{a1} (w3),\ M5\text{: } \Updownarrow (r3);\\ &M6\text{: } \Updownarrow (w1, r1, w0, r0, w2, r2, w0, r0, w1, r1, w3, w3, r3);\\ &M7\text{: } \Updownarrow_{a1} (w0);\ M8\text{: } \Updownarrow_{a0} (r3);\ M9\text{: } \Updownarrow_{a0} (w0);\ M10\text{: } \Updownarrow (r0)\}.\end{aligned} \tag{5}$$

The fault sensitization and detection sequence for each fault is shown in Table V. We will show in Section VI-B that it is unnecessary to distinguish between SA1*,* SA2*,* etc., since the same technique is used to repair both types of faults.

2) Diagnosis Using Sneak-Paths: The sneak-path diagnosis technique is a modified version of [13]. In [13], we only consider defects in a two-level memristor. We now extend the diagnosis technique to multilevel memristor faults and coupling faults. In order to minimize diagnosis time we leverage sneak-paths and use the following sequence:

$$\begin{aligned}\{&\Updownarrow (w0, w0);\ \Updownarrow_{\text{SA}} (r0);\ \Updownarrow_{a0} (w3);\ \Updownarrow_{a1} (r0);\ \Updownarrow_{a1} (w3),\\ &\Updownarrow_{\text{deep}} (r3);\ \Updownarrow (w1);\ \Updownarrow_{\text{SA}} (r1);\ \Updownarrow (w0);\ \Updownarrow_{\text{SA}} (r0);\ \Updownarrow (w2);\\ &\Updownarrow_{\text{SA}} (r2);\ \Updownarrow (w0);\ \Updownarrow_{\text{SA}} (r0);\ \Updownarrow (w1);\ \Updownarrow_{\text{SA}} (r1);\ \Updownarrow (w3, w3);\\ &\Updownarrow_{\text{SA}} (r3);\ \Updownarrow_{a1} (w0);\ \Updownarrow_{a0} (r3);\ \Updownarrow_{a0} (w0);\ \Updownarrow_{\text{deep}} (r0)\}.\end{aligned} \tag{6}$$

Equation (6) can diagnose the type of fault, but the only information about the fault location is that it is somewhere within the RoD. However, we can exploit the structure of the RoD to determine the exact location of the fault.

Fig. 14(a) shows a single RoD for a SA1 fault with sneak-paths restricted to three rows. The value in each square represents the I_{output} when a SA1 fault exists in that cell. If the addressed memory cell suffers from a SA1 fault, I_{output} is I_A. Similarly, if I_{output} is I_B, it implies that the fault lies in the cells in the same row or column as the addressed memory cell and an I_{output} of I_C indicates that the fault lies in the cells diagonal to the addressed memory cell.

We use a binary search algorithm with overlapping RoDs to pinpoint the location of the fault.

Fig. 14(b) demonstrates the steps taken to diagnose SA1 faults. For example, let us assume that there is a SA1 fault at the location marked as X3. The diagnostic sequence (6) is applied. Say, when RoD1 is read during M7, $I_{\text{output}} = I_C$. This indicates that a SA1 fault is located at one of the following locations: X1–X4. In step 2, we narrow down the possible fault locations using RoD2. Say, when RoD2 is read, $I_{\text{output}} = I_C$. This narrows down the possible fault locations to X1 or X3. In step 3, we can pinpoint the location of the fault using RoD3. When reading RoD3, if $I_{\text{output}} = I_A$ then the fault must be X3. However, if the SA1 fault was at location X1 then I_{output} would have been equal to I_B.

3) Diagnosis Using Hybrid Technique: In cases where multiple defects exist in the RoD, sneak-path diagnosis is slow to diagnose the location of all faults. Hence, we introduce a hybrid diagnosis technique that can diagnose clustered faults (multiple faults in the same RoD).

The proposed hybrid diagnosis technique combines March and sneak-path diagnosis to minimize diagnosis time. First, fault detection is performed using sneak-paths and I_{output} is compared to I_A, I_B, and I_C. If I_{output} is equal to one of these currents, a single fault must be located in the RoD and we perform sneak-path diagnosis. However, if $I_{\text{output}} \neq \{I_A, I_B, I_C\}$ then multiple faults must be existing in the RoD. In this case, we perform March diagnosis on all memory cells in the RoD sequentially to determine their locations. Our BIST scheme (Section V) supports both sneak-path and hybrid diagnosis technique. The feedback loop, shown in Fig. 11, facilitates hybrid diagnosis. The BIST performs sneak-path diagnosis until a fault is detected. When a fault is detected the feedback loop is used to signal the read\write controller to switch to March diagnosis to pinpoint the location of the fault.

B. Memory Repair

Once the faults are detected and diagnosed the final step is to suppress erroneous behavior of faulty cells. We categorize the faults into: 1) nonrecoverable and 2) recoverable faults.

1) Nonrecoverable Faults: The erroneous behavior caused by stuck-at and coupling faults can be avoided only by using redundant rows and columns. The memristor crossbar is designed with redundant rows and columns, and a switch block is inserted between the row/column decoder and the crossbar. The switch block may be used to bypass rows/columns which

TABLE VI
TEST SEQUENCE AND FAULTS DETECTED BY EACH SEQUENCE. VARIOUS COMBINATIONS OF TEST SEQUENCES USED TO DETERMINE THE TYPE OF FAULT

Seq No	**March Sequence**	**SA0**	**SA1**	**SA2**	**SA3**	**deep -0**	**deep -3**	**SW**				**FW 0→1**	**Coupling (*a0→a1*)**	**Coupling (*a1→a0*)**
								0→2	**0→1**	**3→1**	**2→0**			
S1	⇕{w0,w0,r0}		√	√	√									
S2	⇕{w3,w3,r3}	√	√	√										
S3	⇕{w3} ⇕$_{a0}${w0} ⇕$_{a1}${r3}	a 1	√	√	a 0								√	
S4	⇕{w0} ⇕$_{a1}${w3} ⇕$_{a0}${r0}	a 0	√	√	a 1									√
S5	⇕{w0, w0,w3,r3}	√	√	√		√								
S6	⇕{w3, w3,w0,r0}		√	√	√		√							
S7	⇕{w0, r0, w2, r2}							√						
S8	⇕{w2, r2, w0, r0}										√			
S9	⇕{w3, r3, w1, r1}									√				
S10	⇕{w0, r0, w1, r1}								√			√		
Diagnostic Sequence		(S2 or S1) and not(S3 or S4 or S5 or S6)				S5 and not S2	S6 and not S1	S7	S10	S9	S8	S10	(S3 or S4)and not (S1 or S2)	

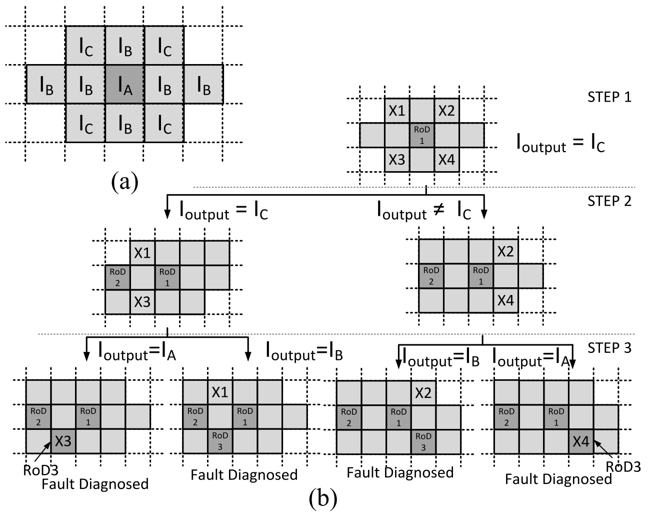


Fig. 13. (a) RoD of a SA1 fault. I_{output} due to a fault at each memory cell is shown. $I_A > I_B > I_C$. (b) Diagnosing a SA1 fault using binary search. X represents possible fault locations.

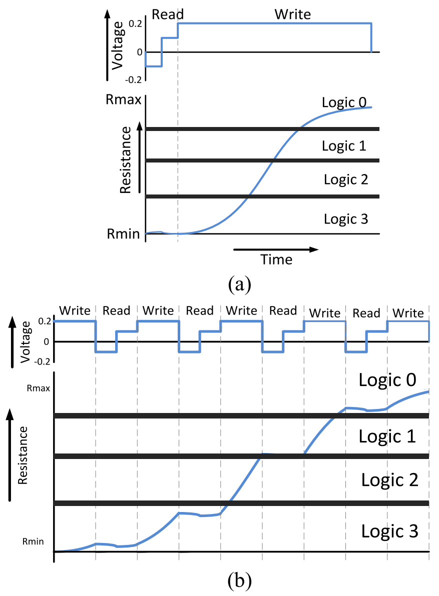


Fig. 14. (a) Typical memristor write. Initial read to determine initial resistance. (b) Read-monitored write [26]. Successive reads followed by short writes ensures required resistance is reached.

have defective memristors and use the redundant rows/columns instead. This technique is used in current SRAM/DRAM technologies and has been studied in [25].

2) Recoverable Faults: We can avoid faulty behavior resulting from fast-write, slow-write, and deep faults. A technique that remedies these faults using a strong write pulse is proposed in [13]. In MLCs each logic level is assigned a narrow resistance band. Strong write pulses result in large changes in resistance and are unsuited for MLCs where accurate programming is required. Hence, a strong write is only compatible with two-level memories. Another technique that may be used to avoid these faults is the read-monitored write [26]. Fig. 13(a) shows a typical memristor write operation for a $3w0$. The read determines the initial resistance of the memristor. This read allows the memory controller to determine the width of the write pulse. The write pulse is applied to the memristor to program it to its required resistance. This long write pulse may be susceptible to SW or FW faults. On the other hand, a read-monitored write [Fig. 13(b)] uses successive reads followed by short write pulses to gradually increase/decrease the resistance of the memristor until the desired resistance is obtained. Even if the short write pulse sensitizes a SW, FW, or deep fault, the read operations provide feedback to the memory controller which ensures that the write pulses are provided until the memristor is programmed to the desired resistance. This technique eliminates all SW, FW, and deep faults, but has a ~5× higher write energy and ~5× to 20× increased write time [26].

TABLE VII
(a) FITTING PARAMETERS AND CONSTANTS USED IN MODELING THE MEMRISTOR [27]. (b) PHYSICAL PARAMETERS USED IN MODELING THE CROSSBAR

(a)

Fitting parameters	Value
x_{on}	0
x_{off}	3×10^{-9}
λ	7.6
p	2

Constants	Value
Voltage (V)	1
R_{on} (Ω)	100
R_{off} (Ω)	0.2 M

(b)

Memristor Parameters	Value
Length	20nm
Width	5nm
Material	TiO_2

Crossbar Parameters	Value
Wire resistance	2.03Ω
Load Resistance	1MΩ

VII. EXPERIMENTAL ANALYSIS

A. Experimental Setup

The memristors are modeled using the TEAM model [15] shown in (1). The fitting parameters used in our model are presented in Table VII.

Memristor defects and faults are modeled by changing the physical parameters of the device as shown in [10]. The 1T1M memristor crossbar is modeled using SPICE while including realistic physical parameters such as wire resistance and peripheral control circuitry. The results are reported for a 16 × 16 crossbar.

B. Experimental Results

In this section, we present an analysis of the proposed fault detection and diagnosis methods. The results are categorized into two sets; the first set presents the effectiveness of our fault detection scheme. The second set highlights the efficacy of the proposed fault diagnosis method.

1) Fault Detection:

a) Accuracy: We built a SPICE model for a 16 × 16 memristor crossbar using the TEAM model described in Section II-B. The crossbar model includes physical parameters such as wire resistance, sneak-path elimination techniques (1T1M) [20] and peripheral control circuitry. To the best of our knowledge, there is no access to the manufactured memristors in the public domains and memristors are only fabricated in a few research laboratories. Hence, there is no available data on the distribution of faults. Accordingly, in our experiments, we assume equal probability for all fault types within the crossbar. To determine accuracy of our test technique, we assume that M_{87} is the addressed memory cell. We consider the following cases.

1) A single fault is injected into a memory cells around M_{87}. We exhaustively consider all fault types at every possible memory location.
2) Multiple faults are injected at locations around M_{87}. We consider from two to five faults injected simultaneously around M_{87}.

We consider all combinations of fault types at all possible combinations of locations. Each logic level is represented by a range of resistances (resistance band) in the memristor. For all simulations, we assume the worst case scenario where the memristor is at the edge of the resistance band. We exhaustively consider all possible combinations of faults types and location. Simulating all possible combinations, we see that 50 out of 38 024 simulated faults are undetected. Hence, we can accurately detect faults with 99.87% success rate.

b) Test time: The test time is determined as a function of the crossbar dimensions (m, n). For each fault type (SA, deep, etc.) we inject the fault into random locations in the crossbar. The average test time is determined over 1000 different runs. While determining the test time with all fault types we assume an equal distribution of all faults. Table VIII (columns 2 and 3) shows the fault detection time using our proposed detection scheme with March test scheme (5). Both March and sneak-path testing utilize the same number of write operations used to detect a fault. However, sneak-path testing significantly reduces the number of read operations. Slow-write and fast-write faults benefit the largest test time reduction, from using sneak-path testing, i.e., 43% test time reduction compared to March testing. Deep faults show a 20% reduction in test time. On average, to detect all fault types, sneak-path testing offers a test time reduction of 24.69% compared to the March sequence in (2).

2) Fault Diagnosis: This set of results evaluates the proposed fault diagnosis scheme. Our proposed March sequence (5) identifies both the type and location of a fault in a single step. However, we use sneak-path diagnosis to reduce diagnosis time. In sneak-path diagnosis, we first use a test sequence to detect a fault and then use additional read operations to locate the fault. In Table VIII (columns 4–6), we show the additional memory accesses required to diagnose a fault. As mentioned in Section VI, if the fault lies in the center of the RoD it is immediately diagnosed. However, if the fault lies in other locations in the RoD, a binary search within the RoD is employed for diagnosis. For example, if a SA0 fault lies in the same row/column as the addressed memory cell, the overhead to diagnose the fault is six read operations. If the fault lies in a memory cell diagonal from the addressed memory cell, two read operations are required to diagnose the fault. Deep and coupling faults have a small RoD that does not extend to the memory cells diagonal to the addressed memory cell.

To evaluate the total overhead for diagnosis, we assume a defect rate (number of defective cells in a crossbar) of 1% and 5%. We assume that all defects have an equal probability of occurring and injected at random locations on the crossbar, with a constraint that no more than two defects can exist within a single RoD. As shown in Table VIII, the average diagnosis time reduction, compared to the March sequence, to perform complete diagnosis (differentiate between all faults) is 31.96% for a 1% defect rate, and 26.77% for a 5% defect rate.

To estimate the diagnosis overhead for the hybrid diagnosis technique we assume a defect rate of 1% and 5% as before.

TABLE VIII
FAULT DETECTION TIME FOR MARCH VERSUS SNEAK-PATH TESTING. DIAGNOSIS OVERHEAD OF SNEAK-PATH AND HYBRID DIAGNOSIS COMPARED TO THE MARCH TESTING

	Fault	**Fault Detection Time (memory accesses)**		**Sneak-path Diagnosis**					**Hybrid Diagnosis**		
				Overhead (reads)			**Diagnosis Time Reduction (%)**		**Overhead (reads)**	**Diagnosis Time Reduction (%)**	
		March	**Sneak-path Testing**	**Center**	**Diagonal**	**Same row/ col**	**1% defect rate**	**5% defect rate**		**1% defect rate**	**5% defect rate**
Non-Recoverable	SA	$6mn$	$4mn + 2mn/13$	0	2	6	30.16	27.74	1~13	29.6	24.94
Non-Recoverable	Coupling	$4mn$	$3mn + mn/9$	0	2	6	21.22	17.22	1~11	20.72	14.72
Recoverable	Deep-0	$4mn$	$3mn + mn/5$	0	×	2	19.67	18.33	1~5	19.25	16.25
Recoverable	Deep-3	$4mn$	$3mn + mn/5$	0	×	2	19.67	18.33	1~5	19.25	16.25
Recoverable	SW0-2	$4mn$	$2mn + 18mn/65$	0	2	3	41.90	37.17	1~11	41.57	35.58
Recoverable	SW0-1	$4mn$	$2mn + 18mn/65$	0	2	3	41.90	37.17	1~11	41.57	35.58
Recoverable	SW3-1	$4mn$	$2mn + 18mn/65$	0	2	3	41.90	37.17	1~11	41.57	35.58
Recoverable	SW2-0	$4mn$	$2mn + 18mn/65$	0	2	3	41.90	37.17	1~11	41.57	35.58
Recoverable	FW0-1	$4mn$	$2mn + 18mn/65$	0	2	3	41.90	37.17	1~11	41.57	35.58
All Fault types		$21mn$	$12mn + 61mn/65$	-	-	-	31.96	26.77	-	35.27	28.02

All defects have an equal probability of occurring, and may also occur in clusters (with multiple faults in each RoD). The hybrid testing method offers a diagnosis time reduction of ~28% compared the March test.

VIII. CONCLUSION

We introduced new fault models for multilevel metal-oxide memristors that covers a range of parametric defects unique to the device and unique to multilevel applications of the device. We also developed an efficient test scheme that detects these faults by leveraging sneak-paths to test multiple memory elements simultaneously. We show that by using sneak-path testing we can reduce the test time by ~24% compared to March tests. We also developed a BIST scheme that implements sneak-path testing.

REFERENCES

[1] J. Hutchby and M. Garner, "Assessment of the potential and maturity of selected emerging research memory technologies," in *Proc. Int. Technol. Roadmap Semiconduct. (ITRS)*, Barza, Italy, 2010, pp. 1–24.

[2] L. O. Chua, "Memristor—The missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, Sep. 1971.

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

[4] Y. Ho, G. M. Huang, and P. Li, "Nonvolatile memristor memory: Device characteristics and design implications," in *Proc. Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 2009, pp. 485–490.

[5] P. O. Vontobel *et al.*, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, no. 42, pp. 42–52, Oct. 2009.

[6] X. Cong, D. Niu, N. Muralimanohar, N. P. Jouppi, and Y. Xie, "Understanding the trade-offs in multi-level cell ReRAM memory design," in *Proc. Design Autom. Conf.*, Austin, TX, USA, 2013, pp. 1–6.

[7] J. Wu and M. Choi, "Memristor lookup table (MLUT)-based asynchronous nanowire crossbar architecture," in *Proc. IEEE Conf. Nanotechnol.*, Seoul, Korea, 2010, pp. 1100–1103.

[8] N. Z. Haron and S. Hamdioui, "On defect oriented testing for hybrid CMOS/memristor memory," in *Proc. IEEE Asian Test Symp.*, New Delhi, India, 2011, pp. 353–358.

[9] N. Z. Haron and S. Hamdioui, "DfT schemes for resistive open defects in RRAMs," in *Proc. Design Autom. Test Europe*, Dresden, Germany, 2012, pp. 799–804.

[10] S. Kannan, J. Rajendran, R. Karri, and O. Sinanoglu, "Sneak-path testing of memristor-based memories," in *Proc. Int. VLSI Design Conf.*, Pune, India, 2013, pp. 386–391.

[11] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. New York, NY, USA: Springer, 2000.

[12] S. Kannan, J. Rajendran, R. Karri, and O. Sinanoglu, "Sneak path testing of crossbar-based non-volatile random access memories," *IEEE Trans. Nanotechnol.*, vol. 12, no. 3, pp. 413–426, May 2013.

[13] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, "Detection, diagnosis, and repair of faults in memristor-based memories," in *Proc. IEEE VLSI Test Symp.*, Napa, CA, USA, 2014, pp. 1–6.

[14] C. Xu, X. Dong, N. P. Jouppi, and Y. Xie, "Design implications of memristor-based RRAM cross-point structures," in *Proc. Design Autom. Test Europe Conf. Exhibit.*, Grenoble, France, 2011, pp. 1–6.

[15] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: Threshold adaptive memristor model," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 211–221, Apr. 2012.

[16] 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. IEEE Eur. Test Symp.*, Seville, Spain, 2009, pp. 61–66.

[17] M. M. Ziegler and M. R. Stan, "Design and analysis of crossbar circuits for molecular nanoelectronics," in *Proc. IEEE Conf. Nanotechnol.*, Washington, DC, USA, 2002, pp. 323–327.

[18] H. Manem, J. Rajendran, and G. S. Rose, "Design considerations for multi-level CMOS/Nano memristive memory," *ACM J. Emerg. Technol. Comput.*, vol. 8, no. 6, pp. 1–22, 2012.

[19] H. Manem, G. S. Rose, X. He, and W. Wang, "Design considerations for variation tolerant multilevel CMOS/Nano memristor memory," in *Proc. ACM Great Lakes Symp. VLSI*, Pittsburgh, PA, USA, 2010, pp. 287–292.

[20] C. Yakopcic, T. M. Taha, G. Subramanyam, R. E. Pino, and S. Rogers, "Analysis of a memristor based 1T1M crossbar architecture," in *Proc. Int. Joint Conf. Neural Netw.*, San Jose, CA, USA, 2011, pp. 3243–3247.

[21] V. A. Hongal, R. Kotikalapudi, Y.-B. Kim, and M. Choi, "A novel 'divide and conquer' testing technique for memristor based lookup table," in *Proc. IEEE Int. Midwest Symp. Circuits Syst.*, Seoul, Korea, 2011, pp. 1–4.

[22] J. Rajendran, H. Manem, R. Karri, and G. S. Rose, "An approach to tolerate process related variations in memristor-based applications," in *Proc. IEEE Symp. VLSI Design*, Chennai, India, 2011, pp. 20–23.

[23] T. F. Chien *et al.*, "BIST design optimization for large-scale embedded memory cores," in *Proc. IEEE Conf. Comput.-Aided Design (CAD)*, San Jose, CA, USA, 2009, pp. 197–200.

[24] S. Hamdoui and Z. Al-Ars, "Scan more with memory scan test," in *Proc. Int. Conf. Design Technol. Integr. Syst.*, Cairo, Egypt, 2009, pp. 204–209.

[25] A. Coker *et al.*, "Multijunction fault-tolerance architecture for nanoscale crossbar memories," *IEEE Trans. Nanotechnol.*, vol. 7, no. 2, pp. 202–208, Mar. 2008.

[26] H. Manem and G. S. Rose, "A read-monitored write circuit for 1T1M multi-level memristor memories," in *Proc. IEEE Int. Symp. Circuits Syst.*, Rio de Janeiro, Brazil, 2011, pp. 2938–2941.

[27] C. Yakopcic, T. M. Taha, G. Subramanyam, R. E. Pino, and S. Rogers, "A memristor device model," *IEEE Electron Device Lett.*, vol. 10, no. 32, pp. 1436–1438, Oct. 2011.

[28] S. Kannan, R. Karri, and O. Sinanoglu, "Sneak path testing and fault modeling for multilevel memristor-based memories," in *Proc. IEEE Int. Conf. Comput. Design*, Asheville, NC, USA, 2013, pp. 215–220.

Sachhidh Kannan (M'10) received the M.S. and Ph.D. degrees in electrical engineering from the Polytechnic Institute of New York University, Brooklyn, NY, USA, in 2009 and 2014, respectively.

He was with IBM Research Laboratory, Zurich, Switzerland, in 2012, where he was involved in characterization and modeling of phase-change memories. His current research interests include hardware security and implementation and applications of emerging memories. He has published over ten conference and journal papers, at various international conferences and publications.

Mr. Kannan was the recipient of the IBM Great Minds Internship.

Naghmeh Karimi (M'05) received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering from the University of Tehran, Tehran, Iran, in 1997, 2002, and 2010, respectively.

From 2007 to 2009, she was a Visiting Researcher at Yale University, New Haven, CT, USA. She was a Post-Doctoral Researcher at Duke University, Durham, NC, USA, for one year and served as a Visiting Assistant Professor at New York University, Brooklyn, NY, USA, for two years. She is currently a Visiting Assistant Professor with the Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, NJ, USA. Her current research interests include design-for-testability, design-for-security, concurrent testing, fault tolerance, reliability enhancement, and hardware security.

Ramesh Karri (SM'95) received the Ph.D. degree in computer science and engineering from the University of California at San Diego, La Jolla, CA, USA.

He is a Professor of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, NY, USA. His current research interests include trustworthy ICs and processors, high assurance nanoscale IC architectures and systems, very large-scale integration design and test, and interaction between security and reliability. He has published over 150 journal and conference papers in the above areas, written two invited articles in the IEEE Computer on Trustworthy Hardware, and an invited article on Digital Logic Design using Memristors in the PROCEEDINGS OF THE IEEE and the IEEE COMPUTER ON RELIABLE NANOSCALE SYSTEMS.

Prof. Karri was the recipient of the Humboldt Fellowship and the National Science Foundation CAREER Award. He co-founded the IEEE/ACM Symposium on Nanoscale Architectures. He has been an IEEE Computer Society Distinguished Visitor from 2013 to present.

Ozgur Sinanoglu (M'01) received the B.S. degrees in electrical and electronics engineering and computer engineering, both from Bogazici University, Istanbul, Turkey, in 1999, and the M.S. and Ph.D. degrees in computer science and engineering from the University of California at San Diego, La Jolla, CA, USA, in 2001 and 2004, respectively.

He is an Associate Professor of Electrical and Computer Engineering, New York University Abu Dhabi, Abu Dhabi, UAE. He has industry experience at Texas Instruments, Dallas, TX, USA, IBM, Armonk, NY, USA, and Qualcomm, San Diego, CA, USA. His current research interests include design-for-test, design-for-security, and design-for trust for VLSI circuits. He has published over 120 conference and journal papers and 15 issued and pending U.S. patents.

Dr. Sinanoglu was the recipient of the IBM Ph.D. Fellowship Award twice and the Best Paper Awards at the IEEE VLSI Test Symposium 2011 and ACM Conference on Computer and Communication Security 2013. He has given over a dozen tutorials on hardware security and test at various IEEE conferences.