# Detection, Diagnosis, and Repair of Faults in Memristor-based Memories

Sachhidh Kannan, Naghmeh Karimi, Ramesh Karri
Department of Electrical and Computer Engineering
NYU Polytechnic School of Engineering
Brooklyn, United States of America
sachhidh@nyu.edu, {nkarimi, rkarri}@poly.edu

Ozgur Sinanoglu
Department of Engineering
New York University, Abu Dhabi
Abu Dhabi, United Arab Emirates
ozgursin@nyu.edu

***Abstract*— Memristors are an attractive option for use in future memory architectures due to their non-volatility, high density and low power operation. Notwithstanding these advantages, memristors and memristor-based memories are prone to high defect densities due to the non-deterministic nature of nanoscale fabrication. The typical approach to fault detection and diagnosis in memories entails testing one memory cell at a time. This is time consuming and does not scale for the dense, memristor-based memories. In this paper, we integrate solutions for detecting and locating faults in memristors, and ensure post-silicon recovery from memristor failures. We propose a hybrid diagnosis scheme that exploits sneak-paths inherent in crossbar memories, and uses March testing to test and diagnose multiple memory cells simultaneously, thereby reducing test time. We also provide a repair mechanism that prevents faults in the memory from being activated. The proposed schemes enable and leverage sneak paths during fault detection and diagnosis modes, while still maintaining a sneak-path free crossbar during normal operation. The proposed hybrid scheme reduces fault detection and diagnosis time by ~44%, compared to traditional March tests, and repairs the faulty cell with minimal overhead.**

**Keywords—Memristor; Memory; Testing; Sneak-paths.**

## I. Introduction

The International Technology Roadmap for Semiconductors highlights that the performance characteristics of emerging resistive random access memory (RRAM) and PCM technologies at the advanced technology nodes (<65nm), 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, excellent scalability and low-power consumption [4][5].

Nanoscale devices are prone to defects due to process variations, design marginalities, and corner operating condition. This requires precise fault modeling and efficient test design in order to keep test time and cost within economically acceptable limits. To screen defects in memristor-based memories several fault models have been considered in literature. These include stuck-at 0/1, open, short and bridging faults [6][7]. A comprehensive fault model, including faults unique to memristors has been proposed in [7]. A physics-based analysis of the memristor physical structure and fault models to represent all possible defects in memristors has been proposed in [8]. All these approaches (except [8]) use the conventional March memory test [6]. 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 [8][9]. In this paper we propose diagnosis (determining the exact type and location of the fault) and repair schemes for memristor-based memories. Specifically,

1) We use sneak-path testing to determine the location of faults and type of defects in the memory. This not only involves the derivation of fault diagnostic sequences, but also their adaptation into a highly parallel sneak-path based framework.

2) We develop a repair scheme to handle failures by modifying the peripheral circuitry and the memory write mechanism.

The rest of the paper is organized as follows: in Section II we review memristor defect mechanisms and the fault models for each defect. The memristor crossbar and the fault detection technique are briefly explained in Section III. The fault diagnosis technique is described in Section IV. The proposed fault repair scheme is discussed in Section V. Section VI evaluates the diagnosis and repair schemes. Section VII concludes the paper.

## II. Metal-oxide Memristor

The metal-oxide memristor is a two-terminal passive element in which the memristance ($M$; measured in Ohms) of the device is determined by the voltage ($V$) as a function of time ($t$) applied between the terminals. The $M$ of the device is expressed as:

$$V(t)/I(t) = M \tag{1}$$

We consider the $Pt/TiO_2/Pt$ memristor (Figure 1(a)) fabricated by HP Labs [3]. The $TiO_2$ film contains - a low resistance region doped with oxygen vacancies and a high resistance undoped region. Applying a voltage ($V(t) > 0$) across the memristor causes an increase in the size of the doped region, yielding the lowest possible resistance $R_{on}$. Conversely, applying a negative voltage ( $V(t) < 0$ ), increases the

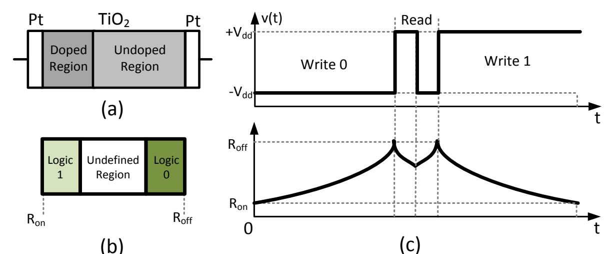


Figure 1. (a) Memristor physical structure, (b) Logic levels and range of memristance as detected by a sense amplifier, (c) Read and write operation in a memristor.

978-1-4799-2611-4/14/$31.00 ©2014 IEEE

IEEE computer society

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on June 14,2021 at 17:36:17 UTC from IEEE Xplore. Restrictions apply.

TABLE 1. SUMMARY OF MEMRISTOR DEFECTS AND FAULT TYPES [8][9]

| Fault Type | Defect type/source | | | | Manifestation |
|---|---|---|---|---|---|
| | *Doping* | *Length* | *Area* | *Others* | |
| *Ideal* | Normal | L | A | - | Defect-free memristor. |
| *SA0* | Undoped | L | A | - | Always at logic 0 irrespective of the voltage applied across it. |
| | - | - | - | Open | |
| *SA1* | Fully doped | L | A | - | Always at logic 1 irrespective of the voltage applied across it. |
| | - | - | - | Short to $V_{dd}$ | |
| *SW0* | Underdoped | L | A | - | Slow transition from a logic 1 to a logic 0. |
| | Normal | L | A | Open defect [17] | |
| *SW1* | Excessively doped | L | A | - | Slow transition from a logic 0 to a logic 1. |
| | Normal | L | A | Open defect [17] | |
| *Deep-0* | Normal | L + ΔL | A | - | Enters 'deep 0' state when resistance $R > R_{off}$. A write logic 1 to a memristor in the deep 0 state is not long enough to switch from deep 0 to logic 1. |
| | | L | A − ΔA | | |
| | | L + ΔL | A − ΔA | | |
| *Deep-1* | Normal | L − ΔL | A | - | Enters 'deep 1' state when resistance $R < R_{on}$. A write logic 0 to a memristor in the deep 1 state is not long enough to switch from deep 1 to logic 0. |
| | | L | A + ΔA | | |
| | | L + ΔL | A − ΔA | | |
| *Deep-1/0* | Underdoped | L − ΔL | A | - | Demonstrates the characteristics of both *Deep-1* and *Deep-0* faults. |
| | | L | A + ΔA | | |
| | | L + ΔL | A − ΔA | | |
| *UR* | Excessively doped | L + ΔL | A | - | Memristor is constantly in undefined region, and may randomly be either logic 1 or logic 0 irrespective of the voltage applied across it. |
| | | L | A − ΔA | | |
| | | L + ΔL | A − ΔA | | |
| | Normal | L | A | Open defect [17] | |
| *Coupling* | Normal | L | A | Short between row/columns [9] | Adjacent memory cell switches during write. |

resistance to a maximum of $R_{off}$. The range of resistance values exhibited by the memristor is used to represent different logic levels. The logic level of a memristor is determined by using a sense amplifier that measures the current relative to a reference current ($I_{ref}$). Any current greater than $I_{ref}$ is a logic 1, and any current less than $I_{ref}$ is a logic 0. A noise margin (or undefined region) is added to enhance the reliability of the memory cell (shown in Figure 1(b)). The output of the sense amplifier may randomly be logic 0 or 1. Figure 1(c) demonstrates the read and write operation of a memristor.

**Memory write operation:** To write a logic 1, a positive pulse is applied across the memristor. Similarly, to write a logic 0, a negative voltage is applied across the device.

**Memory read operation:** A two-stage read operation is used [4]. The ideal read pattern is a negative pulse followed immediately by a positive pulse with the same magnitude and duration, creating a zero net change in resistance.

The memristor fault models proposed in [8], [9] are summarized in Table 1.

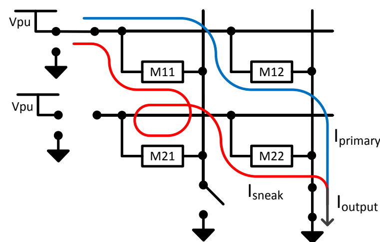


Figure 2. Sneak path testing in a 2×2 crossbar. $M_{12}$ is the addressed memory cell. A sneak path current $I_{sneak}$ flows through $M_{11}$, $M_{21}$ and $M_{22}$. Any variations due to defects in the memristors are reflected as a measurable variation in $I_{output}$.

## III. FAULT DETECTION

In this section, we review our proposed fault detection scheme [9] as a basis for our new fault diagnosis and repair methods. For the sake of clarity, we first provide a brief description of memristor-based crossbars.

### A. Memristor-based Crossbar Memory

Nanoscale memories are constructed using nanoscale crossbar structures [5]. Crossbar architectures built using memristors suffer from sneak-paths which corrupt the output current and result in incorrect read and write operations. Figure 2 illustrates the currents flowing through a 2×2 crossbar when the memory cell $M_{12}$ (memristor in row one, column two) is addressed. 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 ($I_{sneak}$). In practice, sneak-paths are eliminated by adding active components, such as diodes or transistors in series with the memristor making them unidirectional [10]. We consider the 1-Transistor/1-Memristor (*1T1M*) crossbar [11].

In a 1T1M crossbar, during a read/write operation, the rows and columns are enabled based on the outputs of the row and column decoders. The output current is measured at the bottom of each column using a current sense amplifier. When addressing a certain memory cell all transistors on the addressed row are turned ON. For example, while writing to memory cell $M_{i,j}$, all transistors on row '*i*' are turned ON.

### B. March Testing

The direct approach to memory testing is to use the March sequence as shown in [12]:

$$\{M1{:}\Updownarrow(w0, w0, w1);\ M2{:}\ \Updownarrow(r1);\ M3{:}\Updownarrow(w1, w0);\ M4{:}\Updownarrow(r0)\} \quad (2)$$

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on June 14,2021 at 17:36:17 UTC from IEEE Xplore. Restrictions apply.

March elements are labelled 'M$x$'. The following describes how the proposed March algorithm detects the various metal-oxide memristor faults shown in Table I:

***SA1*:** sensitized by *w0* in M3 and detected by M4.
***SA0*:** sensitized by w1 in M1 and detected by M2.
***SW1*:** sensitized by {w0, w1} of M1 and detected by M2.
***SW*0:** sensitized by M3 and detected by M4.
***Deep-0*:** sensitized by M1 and detected by M2.
***Deep-1*:** sensitized by {w1} of M1, M3 and detected by M5.

March exhaustively tests a memory for faults, one memory cell at a time. However, this technique has a long test time and is not scalable for large memristor-based memories.

### C. Sneak-path Testing

S*neak-path testing* [8][9][12] provides two capabilities: 1) the ability to distinguish between a logic level and an undefined state, 2) improve test time and minimize cost by using sneak-paths to test multiple memristors simultaneously.

Sneak-path testing leverages sneak-paths to capture information about multiple memristors in a single measurement. In the 2×2 crossbar shown in Figure 2, the output current is:

$$I_{output} = V_{pu}/M_{12}||^{1}(M_{11} + M_{21} + M_{22}) \quad (3)$$

Hence, any variation in the resistance of $M_{12}$, $M_{11}$, $M_{21}$ or $M_{22}$ results in a change in $I_{output}$. Comparing $I_{output}$ with the ideal current $I_{ideal}$ (output current of a defect-free crossbar), in the presence of sneak-paths, detects faults in multiple memristors in a single step. Obviously, there are limits to the number of memristors that can be tested simultaneously. A group of memristors that can be tested simultaneously is referred to as the '*Region of Detection*' (*RoD*).

Figure 3(a) shows the RoD for a *SA1* fault. The RoD is determined using the techniques in [8], [9]. The dark grey cell is the addressed memory cell. The grey region represents the RoD for a *SA1* fault. In Figure 3(b), we see that we can 'tile' the RoD to maximize coverage area and test the entire memory. In order to detect all *SA1* faults in an 8×8 crossbar, sneak-path testing requires 8 read operations (compared to 64 in a March Test). However, the large number of parallel sneak-paths decreases the effective resistance of the crossbar, resulting in a large output current and excessive power dissipation (0.04W compared to 32µW during normal use). To overcome this problem, we limit the sneak-paths to one row above and below the test cell (shown in Figure 3(c)). Limiting the sneak-paths allows us to control the total output current, and in turn, reduces the test power consumption from 0.04W to 47µW. However, it also reduces the size of the RoD, increasing the number of test points from 8 to 9.

To minimize test time we have proposed the following test sequence by substituting the read operations in Equation 2 with *sneak-path* based access:

$$\{\Updownarrow(w0, w0, w1);\ \updownarrow_{deep}(r1);\ \Updownarrow(w1, w0);\ \updownarrow_{deep}(r0)\} \quad (4)$$

where $\updownarrow_{deep}$ is the sequence in which we address the memory cells to tile the RoD.

In order to enable sneak-path testing, the peripheral circuitry of the crossbar is modified to selectively introduce sneak-paths by turning on the transistors across multiple rows at the same time [8]. Such modifications are enabled only during the test phase.

[1] || represents two resistors in parallel. $R_a||R_b = \frac{R_a.R_b}{R_a+R_b}$

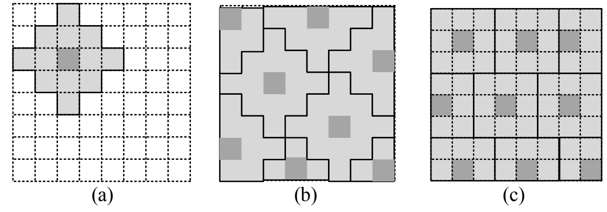


Figure 3. (a) Single RoD. (b) Test points and coverage for a *SA1* fault using RoD. (c) Tiling RoD with sneak-paths restricted to three rows. Dark grey square is memory cell being addressed. Grey region is other memory cells in RoD whose faults can be sensed at output.

During the functional mode, sneak-paths are eliminated using transistors ensuring a sneak-path free operation.

## IV. Fault Diagnosis

The fault diagnosis scheme aims to determine 1) Fault location, and 2) Fault type. The March test (Equation 2) is one of the main approaches used to test memories. In a March test, one memory cell is tested at a time. Therefore, the location of the faulty cell is determined as soon as a fault is detected, i.e., fault detection and diagnosis are performed simultaneously. Also, a typical solution for crossbar repair is to use redundant rows and columns when designing the memory [13]. Columns with defective memristors are bypassed and replaced using the redundant columns. Hence, there is no need to distinguish between different fault types. For example, in the March sequence shown in Equation 2, a sequence of write operations sensitizes the fault and a read operation detects the fault. Each read operation may detect one or more faults. For example, *Deep-0*, *SA1* and *SW1* are detected by March sequence M2. However, our proposed repair scheme (Section V) requires differentiation between the different fault types. Hence, we develop a new March sequence that can distinguish between the fault types.

### A. Diagnostics 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 SW-0 fault. Hence, to distinguish between these faults, the initial state of the cell-under-test is considered. We use $\{w0, w0, r0\}$ and $\{w1, w0, r0\}$ to sensitize and detect a SA1 and SW-0 fault, respectively. Similarly, we use $\{w1, w1, r1\}$ and $\{w0, w1, r1\}$ to sensitize and detect SA0 and SW-1 faults.

Table 2 provides the test sequences used to distinguish between different types of faults. To diagnose *SA1* and *SA0* fault types sequences S1 and S2 can be used, respectively. However, to diagnose a *Deep-0* fault, two test sequences (S4 and S5) are required. A fault is diagnosed as *Deep-0* if it is detected by sequence S5 but not detected by sequence S4. The last row of the table summarizes the diagnostic sequences that should be used for each fault type. The March sequence to detect all faults is:

$$\text{M1: }\Updownarrow(w0, w0, r0);\ \text{M2: }\Uparrow(r0, w1, r1);\ \text{M3: }\Uparrow(w1, r1);\ \text{M4: }\Downarrow(r1, w0, r0)\} \quad (5)$$

***SA1*:** sensitized and detected by M1.
***SA0*:** sensitized by {w1} of M2 and M3. Detected by M3.
***Deep-0/SW1*:** sensitized by {w0, w0} of M1 and {w1} of M2 and detected by M2.
***Deep-1/SW0*:** sensitized by {w1} of M2, {w1} of M3, and {w0} of M4. Detected by M4.
***Coupling:*** sensitized by {w0} of M1 and {w1} of M2 and detected by M2. Also sensitized by {w1} of M3 and {w0} of M4 and detected by M4.

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on June 14,2021 at 17:36:17 UTC from IEEE Xplore. Restrictions apply.

TABLE 2. TEST SEQUENCE AND FAULTS DETECTED BY EACH SEQUENCE. DIAGNOSTICS IS PERFORMED USING VARIOUS COMBINATIONS OF TEST SEQUENCE TO DETERMINE TYPE OF FAULT.

| Seq. No. | March Sequence | SA0 | SA1 | Deep-0 | Deep-1 | SW0 | SW1 | Coupling↑ | Coupling↓ |
|---|---|---|---|---|---|---|---|---|---|
| S1 | ⇕{w0,w0,r0} | | √ | | | | | | |
| S2 | ⇕{w1,w1,r1} | √ | | | | | | | |
| S3 | ⇕{w1,w0,r0} | | √ | | | √ | | | |
| S4 | ⇕{w0,w1,r1} | √ | | | | | √ | | |
| S5 | ⇕{w0, w0,w1,r1} | √ | | √ | | | √ | | |
| S6 | ⇕{w1, w1,w0,r0} | | √ | | √ | √ | | | |
| S7 | ⇕{w0}⇑{r0,w1} | | | | | | | √ | |
| S8 | ⇕{w1}⇑{r1,w0} | | | | | | | √ | |
| S9 | ⇕{w0}⇓{r0,w1} | | | | | | | | √ |
| S10 | ⇕{w1}⇓{r1,w0} | | | | | | | | √ |
| Diagnostic Sequence | | **S2** | **S1** | **S5 and not S4** | **S6 and not S3** | **S3 and not S1** | **S3 and not S1** | **S7 or S8** | **S9 or S10** |

We will show in Section V that it is unnecessary to distinguish between *Deep-0* and *SW1* since the same technique is used to repair both types of faults. Similarly it is unnecessary to distinguish between a *Deep-1* and a *SW0*.

### B. Diagnostics using Sneak-paths

In order to minimize diagnostic time we leverage sneak-paths and use the following sequence:

$$\text{M1: }\Updownarrow(w0,\ w0);\ \text{M2: }\uparrow_{SA}(r0);\ \text{M3: }\uparrow_{c}(r0);\ \text{M4: }\Updownarrow(w1);\ \text{M5: }\uparrow_{deep}(r1);\ \text{M6: }\Uparrow(w1);\ \text{M7: }\downarrow_{SA}(r1);\ \text{M8: }\downarrow_{c}(r1);\ \text{M9: }\Downarrow(w0);\ \text{M10: }\downarrow_{deep}(r0)\} \quad (6)$$

***SA1*:** sensitized by M1 and detected by M2.
***SA0*:** sensitized by M4, M6 and detected by M7.
***Deep-0/SW1*:** sensitized by M1, M4 and detected by M5.
***Deep-1/SW0*:** sensitized by M4, M6, M9; detected by M10.
***Coupling:*** sensitized by M1, M3, M4 and detected by M5. Also sensitized by M6, M8, M9 and detected by M10.

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. Figure 4(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 at 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.

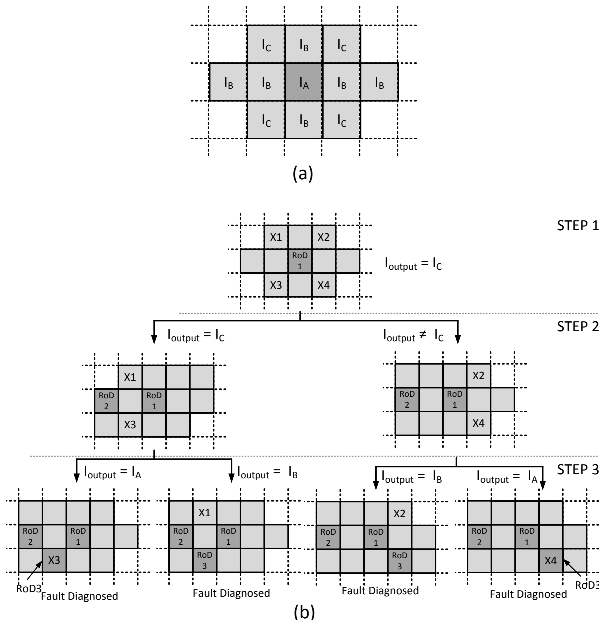


Figure 4. (a) RoD of a *SA1* fault. $I_{output}$ due to a fault at each memory cell is shown. $I_A > I_B > I_C$. (b) Diagnosing a *SA1* fault using binary search. *X* represents possible fault locations.

We use a binary search algorithm with overlapping RoDs to pinpoint the location of the fault. Figure 4(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 (Equation 6) is applied. When *RoD1* is read during M7, $I_{output} = I_C$. This indicates that a *SA1* fault is located at one of the following locations: X1, X2, X3, or X4. In step 2, we narrow down the possible fault locations using *RoD2*. When *RoD2* is read, $I_{output}= I_C$. 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 $I_{output} = I_A$, the fault must be X3. However, if the *SA1* fault was at location X1 then $I_{output}$ would have been equal to $I_B$.

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).

### C. Diagnostics using Hybrid Technique

We propose a hybrid diagnostic technique that combines March and Sneak-path diagnostics to minimize diagnostic 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 diagnostics. However, if $I_{output} \neq \{I_A, I_B, I_C\}$ 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.

## V. 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 two sets: *recoverable* and *non-recoverable* faults.

### A. Recoverable faults

We can avoid faulty behavior resulting from *slow-write* and *deep* faults. For example consider Figure 5. In this figure, two *w0* pulses followed by a *w1* pulse are applied to a memristor. The blue curve represents a defect-free memristor. The defective memristor (dotted red line) enters the deep state during the second *w0* and *w1* cannot restore the memristor to logic 1. We propose the use of a *strong-write 1* pulse, with a magnitude larger than a typical write pulse, to program the

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on June 14,2021 at 17:36:17 UTC from IEEE Xplore. Restrictions apply.

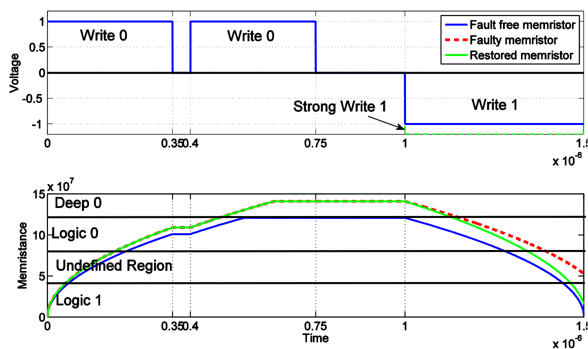


Figure 5. Repair of a Deep-0 fault using a strong-write 1. A faulty memristor enters deep state due to consecutive w0 operation and cannot be properly restored. A strong-write is required to restore to logic 1.

memristor to logic 1. A strong-write 1 can be used to restore a *Deep-0* and a *SW1* fault. Similarly a *strong-write 0* pulse can be used to restore a *Deep-1* or a *SW0* fault. A strong-write pulse has a write energy of 420 fJ/bit, compared to 350 fJ/bit for a typical write.

Figure 6 shows our memristor-based memory repair scheme. Each row has an additional flag bit that stores the state of each row. If a row has a *deep* or *SW* fault, the flag bit is 1. The flag bit is 0 if no faults exist. During normal operation, rows with a flag bit = 1 use a strong-write operation preventing the faults from triggering. Defect-free cells on the same row as faulty cells are also subject to a strong-write. The defect-free cells operate normally, but use more energy compared a typical write. Section VI provides a detailed energy analysis of the proposed repair mechanism.

### B. Non-recoverable Faults

The erroneous behavior caused by *stuck-at*, *coupling*, and *UR* faults can be avoided by using redundant rows and columns. A switch block may be used to bypass columns which have defective memristors use redundant rows. This technique is used in current SRAM/DRAM technologies and has been studied in [13].

## VI. Experimental Analysis

### A. Experimenal Setup

The memristor is modeled using the nonlinear ionic drift model [14][15]. Memristor defects and faults are modeled by changing the physical parameters of the device as shown in [8]. The memristor-based 1T1M crossbars is modeled using SPICE and includes 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, diagnosis and repair methods. The results are categorized to three sets; the first set presents the effectiveness of our fault detection scheme. The second set highlights the efficacy of the proposed fault diagnosis method, and finally the third set helps assess our fault repair scheme.

#### 1) Fault Detection

Table 4 shows the fault detection time using our proposed detection scheme with March test scheme (Equation 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. Coupling faults benefit the largest test time reduction, from using sneak-path testing, i.e. 50% test time reduction compared to March testing. *Deep* and *slow-write* faults show a 20% reduction in test time. On average, to detect all fault types, sneak-path testing shows a test time reduction of 23% compared to the March sequence in Equation 2.

#### 2) Fault Diagnosis

This set of results evaluates the proposed fault diagnosis scheme. Our proposed March sequence (Equation 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 4 (columns 4~6) we show the additional memory accesses required to diagnose a fault. As mentioned in Section III, 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 6 read operations. If the fault lies in a memory cell diagonal from the addressed memory cell, 2 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 diagnostic overhead, we assume a defect rate (number of defective cells in a crossbar) of 1% and 5%. A typical manufacturing process typically seeks a defect rate of < 2%. We show that our technique works with a 5% defect rate and can be used for any memristor crossbar in production. We assume that all defects have an equal probability of occurring and are 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 4, the average diagnostic time reduction, compared to the March sequence, to perform complete diagnosis (differentiate between all faults) is 29.17% for a 1% defect rate, and 26.77% for a 5% defect rate.

To repair the memory it is sufficient to differentiate between recoverable and non-recoverable faults. This is shown in Table 4 (bottom row). The diagnostic time reduction, compared to the March sequence, is 48.97% for a 1% defect rate, and 46.77% for a 5% defect rate.

To estimate the diagnostic overhead for the hybrid diagnostic technique we assume a defect rate of 1% and 5% as before. We assume that all defects have an equal probability of occurring, and may also occur in clusters (with multiple faults in each RoD). Using the hybrid testing method shows a

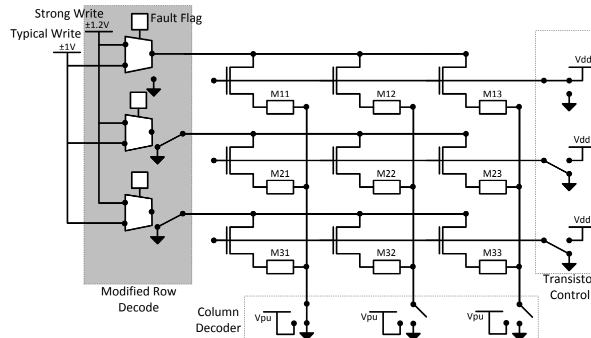


Figure 6. Memory repair scheme for deep and slow-write faults. Grey region shows modifications to row decoder. Fault flags are set during diagnostics if a deep or slow-write fault exists on that row. If fault flag = 1 we use ±1.2V during write. If fault flag = 0 we use a typical ±1V during write.

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on June 14,2021 at 17:36:17 UTC from IEEE Xplore. Restrictions apply.

TABLE 3. AVERAGE WRITE ENERGY PER BIT USING PROPOSED REPAIR TECHNIQUE FOR A RANGE OF DEFECT RATES

| Defect Rate | 0% | 2% | 5% | 8% | 10% |
|---|---|---|---|---|---|
| Write Energy (fJ/bit) | 350 | 352.3 | 355.8 | 359.2 | 361.5 |
| Energy Overhead | 0% | 0.7% | 1.6% | 2.6% | 3.2% |

diagnostic time of ~44% compared the March test.

#### 3) *Memory Repair*

For a $m \times n$ crossbar ($m$ rows and $n$ columns), the overhead incurred by the proposed memory repair technique in terms of transistor count is $18m$.

Table 3 summarizes the average write energy per bit for a defective memory using the proposed repair mechanism. Memristor-based memories are still experimental and fabrication data regarding defect rates are unavailable in literature. Hence, we assume a wide range defect rates between 2-10%. There is no energy overhead to repair *SA* and coupling faults. With a pessimistic assumption that all faults are *deep* or *slow-write* faults, for a 10% defect rate the average increase in write energy is only 3.2%.

## VII. CONCLUSION

Based on our previously proposed memristor fault model and test scheme, we have developed a comprehensive testing solution to detect, diagnose, and repair different faults that occur in the memristor. We showed that sneak-path diagnosis is the fastest way (~48% reduction in time compared to March) to diagnose a memory but is unable to diagnose clustered faults. Our hybrid diagnostic technique can detect all faults (including clustered faults) for a diagnostic time reduction of ~44% compared to traditional March testing. We also developed a repair technique that prevents the activation of the faults in memristor based memories with minimal write energy overhead.

## REFERENCES


[1] J. Hutchby and M. Garner, "Assessment of the Potential and Maturity of Selected Emerging Research Memory Technologies," *International Technology Roadmap for Semiconductors (ITRS)*, 2010.

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

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

[4] Y. Ho, G. M. Huang, and P. Li, "Nonvolatile memristor memory: device characteristics and design implications," *International Conference on Computer-Aided Design*, pp. 485 - 490, November, 2009.

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

[6] J. Wu and M. Choi, "Memristor lookup table (MLUT)-based asynchronous nanowire crossbar architecture," *IEEE Conference on Nanotechnology*, pp. 1100 - 1103, August 2010.

[7] N.Z. Haron and S. Hamdioui, "On Defect Oriented Testing for Hybrid CMOS/Memristor Memory," *IEEE Asian Test Symposium*, pp. 353 - 358, November 2011.

[8] S. Kannan, J. Rajendran, R. Karri, and O. Sinanoglu, "Sneak-path Testing of Memristor-based Memories," *International VLSI Design Conference*, January 2013.

[9] S. Kannan, J Rajendran, R. Karri, and O. Sinanoglu, "Sneak Path Testing of Crossbar-based Non-Volatile Random Access Memories," *IEEE Transactions on Nanotechnology*, vol. 12, no. 3, May 2013.

[10] H Manem, G.S Rose, X He, and W. Wang, "Design considerations for variation tolerant multilevel CMOS/Nano memristor memory," *ACM Great lakes symposium on VLSI*, pp. 287 - 292, 2010.

[11] C. Yakopcic, T.M. Taha, G. Subramanyam, R.E. Pino, and S. Rogers, "Analysis of a memristor based 1T1M crossbar architecture," *The International Joint Conference on Neural Networks*, pp. 3243 - 3247, August, 2011.

[12] S. Kannan, R. Karri, and O. Sinanoglu, "Sneak path Testing and Fault Modeling for Multilevel Memristor-based Memories ," *IEEE International Conference on Computer Design*, October, 2013.

[13] A. Coker et al., "Multijunction Fault-Tolerance Architecture for Nanoscale Crossbar Memories," *IEEE Transactions on Nanotechnology*, vol. 7, no. 2, March, 2008.

[14] D. B. Strukov and R. S. Williams, "Exponential Ionic Drift: Fast Switching and Low Volatility of Thin-Film Memristors," *Applied Physics A: Materials Science and Processing*, vol. 94, no. 3, pp. 515 - 519, March, 2009.

[15] C. Yakopcic, T.M. Taha, G. Subramanyam, R.E. Pino, and S. Rogers, "A Memristor Device Model," *IEEE Electron Device Letters*, vol. 10, no. 32, pp. 1436 - 1438, October, 2011.

[16] M. Bushnell and V. Agrawal, *"Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits"*.: Springer, 2000.

[17] N. Z. Haron and S. Hamdioui, "DfT Schemes for Resistive Open Defects in RRAMs," *Design, Automation and Test in Europe*, 2012.


TABLE 4. DIAGNOSTIC OVERHEAD OF SNEAK-PATH AND HYBRID DIAGNOSIS FOR A 16×16 CROSSBAR ($m$, $n$ = 16) COMPARED TO THE MARCH TESTING.

| | Fault | Fault Detection Time (memory accesses) | | Sneak-path Diagnostics | | | | | Hybrid Diagnostics | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Overhead (reads) | | | Diagnostic Time Reduction (%) | | Overhead (reads) | Diagnostic 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 Faults* | *SA0* | $3mn$ | $2mn + mn/13$ | 0 | 2 | 6 | 29.64 | 25.13 | 1~13 | 29 | 11.67 |
| | *SA1* | $3mn$ | $2mn + mn/13$ | 0 | 2 | 6 | 29.64 | 25.13 | 1~13 | 29 | 11.67 |
| | *UR* | $6mn$ | $3mn + mn/5$ | 0 | 2 | 6 | 45.86 | 42.67 | 1~5 | 49.17 | 45.83 |
| | *Coupling Faults* | $6mn$ | $3mn$ | × | × | 3 | 50 | 50 | 1 | 0 | 0 |
| *Recoverable Faults* | *Deep-0* | $4mn$ | $3mn + mn/5$ | 0 | × | 2 | 19.6 | 18 | 1~5 | 23.75 | 18.75 |
| | *Deep-1* | $4mn$ | $3mn + mn/5$ | 1 | × | 2 | 19.55 | 17.75 | 1~5 | 23.75 | 18.75 |
| | *SW1* | $4mn$ | $3mn + mn/5$ | 1 | × | 2 | 19.55 | 17.75 | 1~5 | 23.75 | 18.75 |
| | *SW0* | $4mn$ | $3mn + mn/5$ | 1 | × | 2 | 19.55 | 17.75 | 1~5 | 23.75 | 18.75 |
| *Full Diagnostics* | | $11mn$ | $5mn + 274mn/260$ | - | - | - | 29.17 | 26.77 | - | 25.27 | 18.02 |
| *Differentiate between Recoverable and non-recoverable* | | $10mn$ | $6mn + 36mn/65$ | - | - | - | 48.97 | 46.77 | - | 44.46 | 44.11 |


Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on June 14,2021 at 17:36:17 UTC from IEEE Xplore. Restrictions apply.