An On-line Test Strategy and Analysis for a 1T1R Crossbar Memory

Manuel Escudero-Lopez, Francesc Moll and Antonio Rubio
Electronics Engineering Department
Universitat Politecnica de Catalunya
Barcelona, Spain
manuel.escudero@upc.edu, francesc.moll@upc.edu, antonio.rubio@upc.edu

Ioannis Vourkas
Dept. of Electrical Engineering
Pontificia Universidad Catolica de Chile
Santiago, Chile
iovourkas@uc.cl

***Abstract*—Memristors are emerging devices known by their nonvolability, compatibility with CMOS processes and high density in circuits density in circuits mostly owing to the crossbar nanoarchitecture. One of their most notable applications is in the memory system field. Despite their promising characteristics and the advancements in this emerging technology, variability and reliability are still principal issues for memristors. For these reasons, exploring techniques that check the integrity of circuits is of primary importance. Therefore, this paper proposes a method to perform an on-line test capable to detect a single failure inside the memory crossbar array.**

I. Introduction

Resistive switching devices (memristors) are emerging electronic devices that are receiving significant attention because of their promising properties including being passive nonvolatile memory elements, storing data in the form of resistance. Altough they were postulated by L. Chua back in 1971 [1], Chuas theory was connected with practice only in 2008 [2]. Some other interesting characteristics of memristors are the compatibility with CMOS technology and the highest possible device integration density [3]. It is expected that memristive device speed could match that of CMOS devices. Nonetheless, currently it isn't a mature enough technology and there is space for improvement in several aspects. For instance, variability and reliability are considered among the most critical issues [4].

Memristors are suitable devices for applications such as memories and computing, both digital and analog. As memristors are nonvolatile, they are ideal to store data even when not powered. This feature and their potential high device density are two key properties for memory applications. However, variability and reliability must be controlled in these applications and strategies to cope with these issues must be developed. On-line testing is a useful technique as memristors may fail eventually due to its improvable reliability. There are some works about on-line testing circuits with memristor crossbars; e.g. [5] applies design for testability to detect open faults in crossbar, whereas [6] takes advantage of sneak-path currents to perform the testing procedure faster and [7] designs to detect bridge defects.

In this context, our work presents a simple fault model for a single one transistor one memristors (1T1R) cell more focused in the possible malfunction of the cell devices than the previous works, as well as the interferences between cells when faults occur and a method to detect them during the system normal operation. The paper is organized as follows. Section II introduces the device model used in the work, Section III depicts the memory system, Section IV shows the fault model considered, Section V presents the on-line test procedure and Section VI shows the simulation results. Finally, Section VII concludes the paper.

II. Memristor fundamentals in brief

The memristor is a passive two-terminal device, a passive circuit element with a characteristic parameter named memristance. Memristance is conceptually defined as the derivative of flux with respect to electric charge, but it is commonly explained as a resistance that depends on the previous history of the electric charge that passes through it. There are different types of memristors: bipolar or unipolar, filamentary or homogeneous switching [8]. For instance, in the resistive RAM (ReRAM) device assumed in this paper, the change in its memristance is attributed to the creation or destruction of one or more conductive filaments inside the metal-insulator-metal device structure (in line with the soft breakdown of an oxide intermediate layer).

The memristor model adopted in this work is presented in [9]. The model is based on the formation of a two-dimensional conductive filament, described as a cylinder that modulates its length and width according to the voltage applied to the terminals of the device. It features variability, parasitic elements and temperature effects, among other features. The used parameter values have been extracted from TiN/HfOx/TiOx/Pt devices of 10 nm feature size. The typical I-V pinched curve observed in memristors is shown in Figure 1, where V_M and I_M are the voltage across the memristor and the flowing current, respectively. The I/V curve was generated using a triangular signal of amplitude 2 V and period 40 ns. The memristor switches to low resistive state (LRS) when a positive voltage is applied. Then a negative voltage gradually increases the memristance, finally leading to a high resistive state (HRS).

In a memory application memristors are tipically excited with voltage pulses. The width and voltage level are important to achieve desired resistance states. A long, high voltage pulse is needed to bring the memristor to a given state, while a

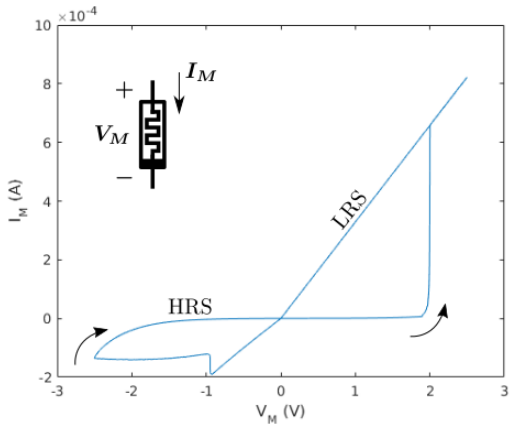


Fig. 1. I/V characteristic curve of the used memristors model. The inset shows the memristors symbol and polarity convention used for the applied voltage and the flowing current.

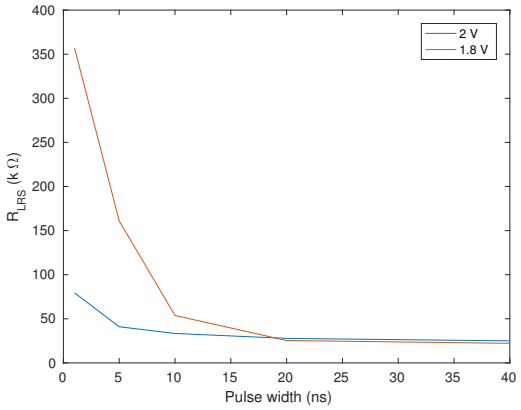


Fig. 2. Low resistance states reached with voltage pulses of 2 V and 1.8 V, for pulse widths from 1 ns to 40 ns.

short, low voltage pulse is sufficient to get the resistance of the device without modifying the state adequate for a reading phase. To illustrate the change of state due to different voltage pulses a simulation is carried out. A memristor is initialized at a HRS state (about 1.5 MΩ) and excited with voltage pulses of 1.8 V and 2 V and different width pulses. Figure 2 shows the obtained results: as voltage and pulse width increase the change in the memristance is higher. This is especially clear for small pulse widths. In order to read and write memristors, voltage pulses of $V_{read} = 1$ V 10 ns long and $V_{write} = 2$ V 120 ns have been chosen to read and write the memristor, respectively.

III. Target System Architecture

Before moving to the complete memory system considered in this work, Figure 3 shows a single memory cell with simple peripheral circuitry which we analyze first. The 1T1R cell of interest consists of a memristor M connected in series with an NMOS select transistor [10]. Memristor M stores 1 bit of information, where LRS in our case stands for logic '1' and

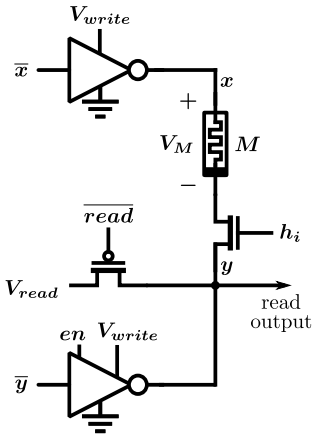


Fig. 3. 1-bit storage unit consisting of a memristor as a memory cell, and peripheral circuitry which enable writing and reading the content of the cell.

HRS logic '0', respectively. The NMOS device enables the peripheral circuitry to perform the following three operations: write '1' or '0' to the cell, and read the state stored in the cell.

Two inverters work as drivers to write the memristor M. They are powered to V_{write} level. It is assumed that the inverters can supply enough current to memristors. The one at the bottom of the figure is a tri-state inverter, so it must provide a high-impedance output to not interfere with the PMOS transistor used for reading. To write cell to '1', $\overline{x} =$ '0' and $\overline{y} =$ '1' so $V_M = V_{write}$. To write cell to '0', the signal values are switched, $\overline{x} =$ '1' and $\overline{y} =$ '0' so $V_M = -V_{write}$. When writing, both inverters must be working, hence $en =$ '1'.

Finally, to read the state of M, the tri-state inverter output is disabled setting $en =$ '0', the PMOS is enabled setting $\overline{read} =$ '0' and $\overline{x} =$ '1'. Using this configuration, M is put in series with the PMOS and y will act as the read output. A voltage at node y near ground indicates a LRS or '1' while voltage at y near V_{read} means a HRS or '0'.

Figure 4 shows the complete memory architecture considered in this work. The memristive crossbar array is the core of the system and contains m rows and n columns of 1T1R cells, as depicted in the inset. These cells are interconnected using common horizontal and vertical lines named x_i and y_j respectively, where $1 \leq i \leq n$ and $1 \leq j \leq m$. The addition of the NMOS select transistor in every cell prevents from any unwanted changes in the state of memristors due to sneak current paths, as well as facilitates the proper read and write operation. The NMOS transistor is controlled by the gate signal h_i, which is common to all cells belonging to the same row i.

At the boundaries of the crossbar there are peripheral circuits for control and read operations. Again, inverters which drive the memristors are powered to V_{write} voltage as they are intended to write the content of memory cells. Column drivers are tri-state inverters and their enable signal is connected to a single node en. The read circuitry is powered with V_{read} voltage (assuming $V_{read} < V_{write}$). PMOS transistors used to

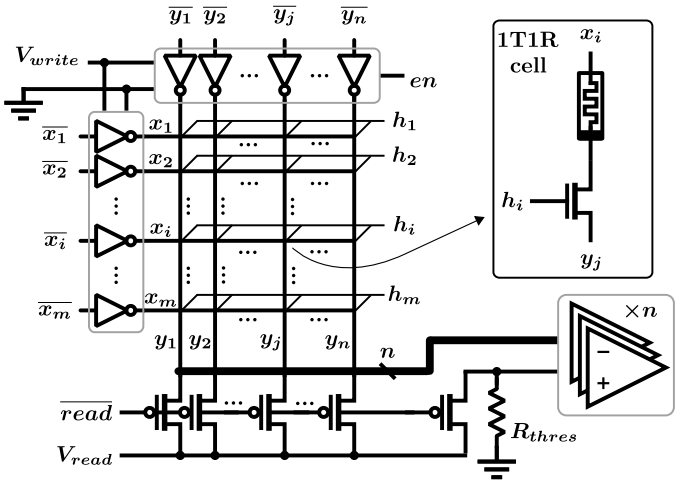


Fig. 4. Memory system overview: $m \times n$ 1T1R crossbar with control and read circuitry.

read memristors are controlled with a single $\overline{read}$ gate signal. The read circuitry also includes n comparators plus a resistor of value R_{thres} that acts as a threshold between LRS and HRS. The logic level at the comparator is in accordance with the stored memristor state, i.e. '0' (0 volts) is HRS and '1' (V_{read}) is LRS.

This memory crossbar is built to operate not only with single cells, but with the entire row. Now, the four operations available are write entire row to '1', write entire row to '0', write a word to an entire row and read an entire row. To write or read an entire row i, the row must be enabled selecting h_i and inverters and PMOS transistors are configured depending the operation and by following similar instructions as in the case of one single cell described previously.

IV. Fault model

The circuit complexity in integrated circuits has been growing very fast but at the cost of an increase of physical failures. Many different types of faults may appear in a circuit, in the manufacturing process or during its useful lifetime. Memristors are not an exception. Their inherent "structure-modifying" nature of operations (i.e. the soft breakdown in the intermediate metal-oxide) and their still immature manufacturing processes are both issues of reliability.

In this work, a single fault scenario inside the memory crossbar cells is assumed, that being neither in the interconnections nor the external circuitry. The fault is assumed only in the memory cell and may arise from two different sources: the select transistor or the memristor. Transistors may be stuck-on (T_{s-on}) or stuck-open (T_{s-open}). In modeling, this corresponds to substituting the transistor by a resistor of a low or high value depending on the type of failure. As for the memristor, it can get stuck in a LRS (called M_{s-LRS}) or HRS (called M_{s-HRS}). Likewise, the modeling approach for these failures concerns using a resistor with a low or high value instead of a memristor. In this case, resistance values depend on the typical HRS or LRS achieved in the write operation. Table I summarizes the considered faults. This notwithstanding, other system faults, e.g. the case of having a memristors not switched completely, are not considered here since we set a single resistance threshold for reading the state and there is no undefined intermediate resistance band.

TABLE I
Fault Models in Crossbar Cells (Single Faults)

Fault	Name	Description
Transistor stuck-at-on	T_{s-on}	Channel remains conductive.
Transistor stuck-at-open	T_{s-open}	Channel remains cut-off.
Memristor stuck-in-LRS	M_{s-LRS}	Unalterable LRS.
Memristor stuck-in-HRS	M_{s-HRS}	Unalterable HRS.

V. On-line Test Strategy

Having established the system architecture and the fault model, in this section the proposed on-line testing algorithm is presented. The main goal is to check a single fault, that being any of the four types mentioned before, occurring inside the memory crossbar. This process must be robust and keep the original stored data unaltered. The strategy, which is similar that in [11], consists in writing some known data in every row, checking them for unexpected data, and evaluating this unexpected data at the end of the algorithm. For each row the following actions take place:

1) Read row content and store the data.
2) Write the entire row to '1'.
3) Check if the content of the entire row is '1'.
4) Write the entire row to '0'.
5) Check if the content of the entire row is '0'.
6) Retrieve stored data and write to the row.

Next, we analyze how each type of fault is detected using the previous scheme.

A. T_{s-open}, M_{s-LRS} and M_{s-HRS} detection

Let's assume that inside the row the algorithm is goint to check there is one of the previously mentioned faults. After writing '1' to an entire row, the read output should be '1' for all cells in the row. If there is a M_{s-HRS} or T_{s-open} fault in the active row it will be detected when reading the entire row, as its high resistance is interpreted as a memristor in a HRS and the read output is '0' for the faulty cell. However, M_{s-LRS} fault in the active row is not detectable as it already shows a LRS and it's mistaken by a functional memristor. Then, after writing the entire row to '0', the read output should be '1' for all cells in the row. If there is a M_{s-LRS} fault in the active row is will be detected when the row is read. That cell will be stuck to a LRS and the read output is '1'. Again, M_{s-HRS} and T_{s-open} in the active row are not detectable as their high resistance shown will be, at least, as high as the other written cells.

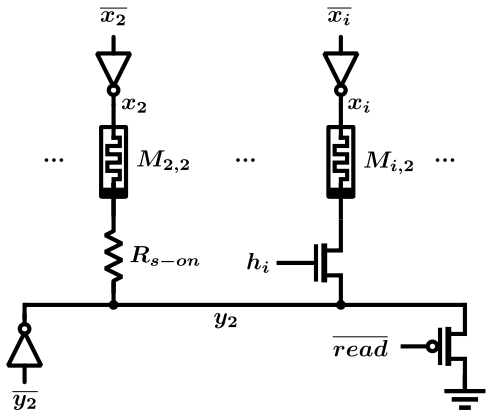


Fig. 5. Cell (2,2) has a fault T_{s-on}. When reading any cell (i,2) of the same column, the faulty cell interferes in the operation.

B. T_{s-on} detection

Note that T_{s-on} is not detectable using write entire row to '0' or write entire row to '1' operations separately. Figure 5 depicts a cell with this fault, for instance the cell (2,2). This schematic will be useful to explain how T_{s-on}.

Writing the entire row to '1' operation, if the row to be written is number 2, the memristor $M_{2,2}$ is written as normally. In the case that the written row is i (another than 2), the memristor of the faulty cell interferes in the read process, but as the memristor $M_{i,2}$ are already in LRS, the total resistance seen by the read circuit is approximately equivalent to the LRS, completely masking the fault.

Writing the entire row to '0', if the written row is number 2, the memristor $M_{2,2}$ is previously written to '0'. When another row i is being written, the memristor $M_{2,2}$ interferes in the read process. Unfortunately the content of this memristor is unknown. Hence if memristor $M_{2,2}$ is in LRS the resistance seen by the read circuit is approximately equivalent to the LRS and not the expected value of HRS (previously written in memristor $M_{i,2}$.

Indeed, T_{s-on} is not detectable using single operations. Even more, this fault is ruining the reading process. In order to be detectable, the fact that the faulty cell interferes the reading of another row is exploited. Write '1' and write '0' operations are performed slightly different.

In write '1' operation, inverters are configured to write each row to '1'. Theoretically, only the active row is written because other rows are disabled. However, if there is a cell with T_{s-on} in another row than the active one, its memristor will be written to '1'. When checking the active row, the T_{s-on} fault is masked as explained before.

In write '0' operation, inverters are configured to write only all the active row to '0'. Doing so keeps the memristor of the failing cell in LRS. When checking the active row, the reading reveals a fault if T_{s-on} isn't located at the active row and is masked if it's in the active row.

With this tricky sequence of operations, a T_{s-on} appears as a column with $m-1$ faulty cells in a column and only one functional. Obviously, a final evaluation is required to mark the functional cell as the real faulty cell. Finally, T_{s-on} is recognizable once modified the write operations. Input signals applied in each operation of the online test are detailed in Table II. Note that this procedure doesn't affect the detection of the other faults T_{s-open}, M_{s-LRS} and M_{s-HRS}.

TABLE II
INPUT SIGNALS CONFIGURATION FOR EACH OPERATION

Signal	Read	Write '1'	Write '0'
h_i	1	1	1
$h_k, \forall k \neq i$	0	0	0
$\overline{read}$	0	1	1
en	0	1	1
$\overline{x_i}$	1	0	1
$\overline{x_k}, \forall k \neq i$	1	0	0
$\overline{y_j}, \forall j$	X	1	0

C. Online test algorithm

The complete on-line test process is shown in the following algorithm. The first *for* loop is in charge of detecting unexpected data among the rows. Content of each row is stored in $s_{n-1} \dots s_0$ before it is written. Also $d_{n-1} \dots d_0$ is used to save temporarily the read data to be analysed. The second *for* loop evaluates the unexpected data. If there is more than one cell mark as faulty, the last *for* loop looks for the only one that is correct (T_{s-on} fault case).

1: **for** i from 0 to $m-1$ **do**
2: read row $i \rightarrow s_{n-1} \dots s_0$
3: write entire row i to '1'
4: read row $i \rightarrow d_{n-1} \dots d_0$
5: **for** j from 0 to $n-1$ **do**
6: **if** ($d_j =$ '0') **then**
7: mark cell (i,j) as faulty cell
8: **end if**
9: **end for**
10: write entire row i to '0'
11: read row $i \rightarrow d_{n-1} \dots d_0$
12: **for** j from 0 to $n-1$ **do**
13: **if** ($d_j =$ '1') **then**
14: mark cell (i,j) as faulty cell
15: **end if**
16: **end for**
17: write entire row i to $s_{n-1} \dots s_0$
18: **end for**
19: **for** j from 0 to $n-1$ **do**
20: **if** (all cells in column j are faulty but one cell(k,j)) **then**
21: mark cell (k,j) as faulty and unmark other cells in column j
22: **end if**
23: **end for**

Finally, note that this technique is still useful when there are more faulty cells as long as there is only one faulty cell per column.

VI. Results

In this section, the circuitry used in the system is checked through simulation (using Cadence Virtuoso ADE) in order to verify errors are detectable as mentioned in section IV. Inverters and pass transistors are simulated using a predictive transistor model for 65 nm node [12]. The comparator is implemented as a Verilog-A block. Values used in simulations are $V_{write} = 2$ V, $V_{read} = 1$ V, $R_{thres} = 200$ kΩ and $R_{s-on} = 1$ kΩ, where R_{s-on} is the resistance of the transistor stuck-at-on. Memristors are always initialized at HRS.

First, a simulation of the circuit shown in Figure 3 is performed. Inverters are designed to drive enough current to write succesfully the memristor M. The performance of the circuit is shown in Figure 6a. The memristor starts at HRS. In the read stage, a '0' is at read output. After writing a '1', a LRS is observed. Finally, a '0' is written in the memristor and the read output is HRS again as expected. The write pulse width is 120 ns, long enough to change the memristor state, and the read one is 10 ns, short enough to keep the state. With these voltage pulses, HRS value is 1.5 MΩ and LRS 21 kΩ.

After it is shown that all the single cell operations work as expected, now the most interesting case of fault detection is when there is a T_{s-on} fault and the corresponding memristors interferes with another cell found in the active crossbar row. As shown in Figure 5, $M_{2,2}$ is the faulty cell of the memory array and the faulty transistor is represented using the equivalent model presented in Table I. The on-line test starts to check the cells of the first row ($i = 1$) and this first iteration is simulated. Instead of simulating the entire system with just one faulty cell, only the target cell to be read and the faulty cell are included in the simulation. Input and output signals are plotted in Figure 6b. Interconnect resistance and capacitance are neglected for simplicity as our purpose is first to check validity of the proposed approach and signal trends. $M_{2,2}$ and $M_{i,2}$ are initialized at HRS. Reading the output after the write '1' operation is correct but reading after the write '0' operation detects an unexpected output, i.e. a LRS where it should be a HRS. The test marks the $M_{1,2}$ cell as faulty. When the second row is checked, as explained in the Section 3c, there is any unexpected output and the faulty cell is marked as non-faulty. As every other row is checked, each $M_{i,2}$ is marked as faulty. At the end of the algorithm, the faulty and non-faulty marks are swapped. Hence, T_{s-on} is detected in $M_{2,2}$ cell as expected.

Finally, it is important to spend few words on the memory used by the algorithm to achieve its goal. Obviously, there is a need for an auxiliary memory. When storing the data contained in the cells before starting the on-line test only the data of an entire row is stored. However, when the algorithm marks faulty cells the cost may be high as the system scales. The algorithm can be improved to overcome this issue, i.e. to store only few last faulty and non-faulty marks and make a decision of the fault scenario taking into account the single failure assumption.

VII. Conclusion

An on-line test strategy has been proposed here able to detect single failures of the memory cells in 1T1R crossbar-based memory systems. The memory crossbar can operate normally and testing can be performed anytime. Although variability is critical in memristors, the proposed method allows easily detecting different types of faults. Simulation results confirm that the interference from a faulty cell in the on-line test procedure doesn't mislead the result.

Acknowledgment

This work has been funded by the Spanish MINECO and ERDF (TEC2013-45638-C3-2-R).

References

[1] L. Chua, "Memristor - The Missing Circuit Element", *IEEE Trans. Circuit Theory*, vol. CT-18, pp.507-519 Sept. 1971.

[2] D. B. Strukov, G. S. Snider, D. R. Stewart, S. Williams, "The missing memristor found", *Letters*, Nature, vol. 453, pp. 80-84, May 2008.

[3] H.-S. Philip Wong, H.-Y. Lee, S. Y., Y.-S. Chen, et. al., "Metal-Oxide RRAM", *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951-1970, June 2012.

[4] P. Pouyan, E. Amat, A. Rubio, "Reliability Challenges in Design of Memristive Memories", *5th European Workshop on CMOS Variability (VARI)*, pp. 1-6, Sept. 2014.

[5] S. Hamdioui, M. Taouil, N. Z. Haron, "Testing Open Defects in Memristor-Based Memories", *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 247-259, Jan. 2015

[6] S. Kannan, J. Rajendran, R. Karri, "Sneak-path Testing of Memristor-based Memories", *International Conference on Embedded Systems (VLSID)*, Jan. 2013.

[7] N. Arshad, F. Salehuddin, S. I. Salim, N. Soin, "Defect-oriented Test and Design-for-Testability Technique for Resistive Random Access Memory", *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 7, no. 2, July-December 2015.

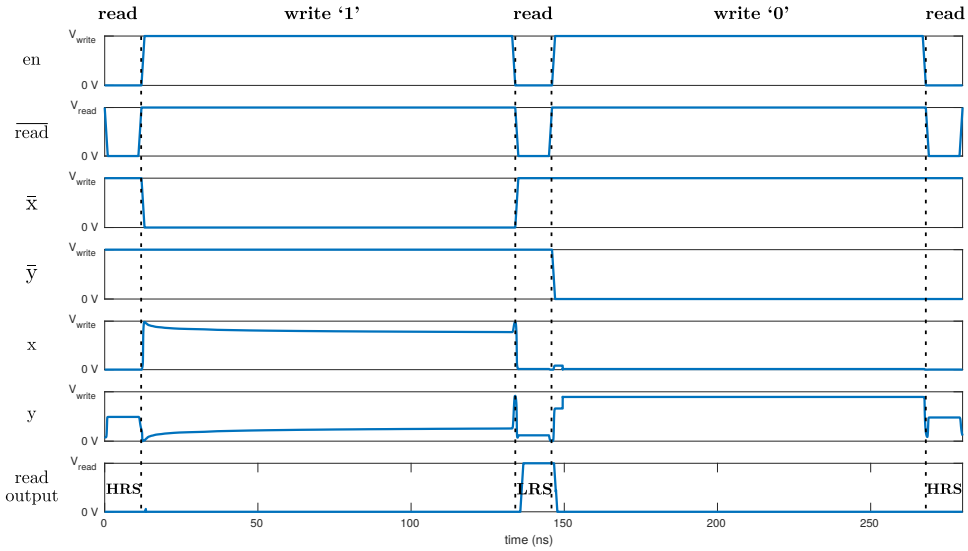
[8] I. Vourkas and G. C. Sirakoulis, "Memristive crossbar-based nonvolatile memory" in *Memristor-Based Nanoelectronic Computing Circuits and Architectures*, 1st ed. Switzerland: Springer, 2016, pp 101147.

[9] H. Li, Z. Jiang, P. Huang, Y. Wu, H. Y. Chen, et. al., "Variation-aware, reliability-emphasized design and optimization of RRAM using SPICE model", *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2015.

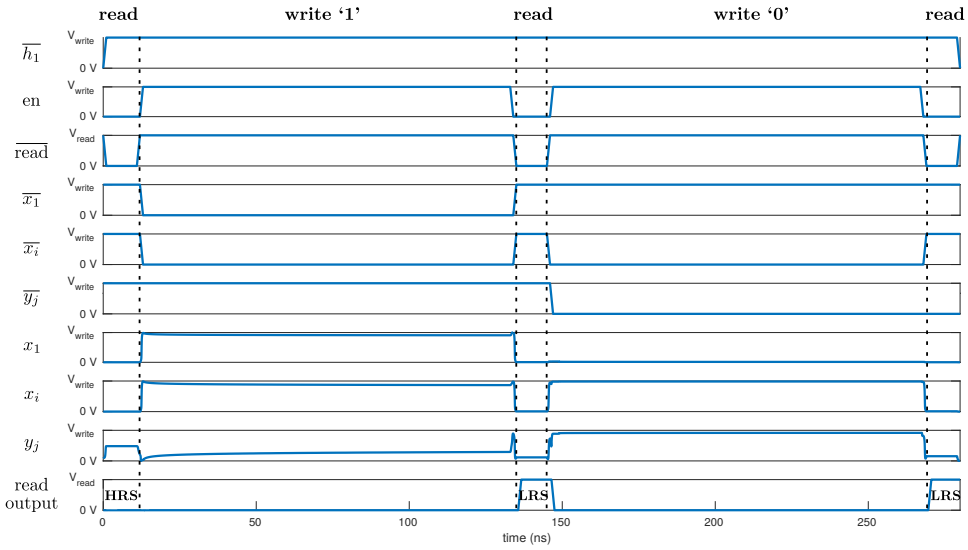
[10] J. Y. Seok, et al., "A Review of Three-Dimensional Resistive Switching Cross-Bar Array Memories from the Integration and Materials Property Points of View", *Advanced Function Materials*, vol. 24, no.34, pp. 5316-5339, 2014.

[11] M. Nicolaidis, "Transparent BIST for RAMs", *International Test Conference Proceedings*, Baltimore, MD, pp. 598-607, Oct. 1992.

[12] Predictive Transistor Model Website from Arizona State University: http://ptm.asu.edu/



(a) Simulation of a single cell.



(b) Simulation of a cell with the interference of a faulty cell.

Fig. 6. Simulation results for a target cell with the interference of a faulty cell.