Memristor-Specific Failures: New Verification Methods and Emerging Test Problems

Baishakhi Rani Biswas and Sandeep Gupta
Electrical and Computer Engineering
University of Southern California
Los Angeles, USA
bbiswas@usc.edu, sandeep@usc.edu

2022 IEEE 40th VLSI Test Symposium (VTS) | 978-1-6654-1060-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/VTS52500.2021.9794274

***Abstract*—We study two types of memristor-specific logic failures in Memristor Ratioed Logic (MRL), an extensively studied Memristor-CMOS hybrid logic design style. Cascading failures have been previously observed as causing significant voltage degradation, and hence logic values errors, in certain MRL logic circuits. Here, we present the first systematic study of this type of logical error and identify its key properties as a function of circuit structure and patterns applied. We then propose a method to generate patterns that cause the worst-case output voltage for a given MRL circuit and hence facilitate pre-fabrication verification. We then present the first study of another type of logic failure for voltage controlled memristor devices, namely a race when memristors in series, and with the same polarity, switch states from Ron to Roff. We show that such a race can cause non-deterministic behavior depending on circuit structure, values of memristor parameters, and the initial states of memristors when a pattern is applied. We then generate patterns and initial states that excite such race and hence potentially cause logic errors to enable verification.**

***Index Terms*—Memristor Ratioed Logic, Cascading, Logic failure, Verification, Test pattern**

I. Introduction

The physical scaling limit of conventional CMOS devices is accelerating the introduction of several emerging materials and devices [1]. A memristor [2], or a memory resistor, is one of the most promising emerging devices. Although first theoretically predicted in 1971, the research on memristors accelerated after 2008, when HP Labs realized the first nano-scale memristor [3]. Memristor supports non-volatile operation, i.e., it can maintain indefinitely its state (the resistance value) assigned via a write operation. Further, this two-terminal device has excellent scalability and low power consumption. Memristors have already found applications in non-volatile memory blocks [4], [5], digital logic circuits [6]–[8], analog filters [9], and neuromorphic computation [10].

In the domain of digital logic, pure memristor logic like MAGIC [6] and IMPLY [7] are difficult to integrate with conventional CMOS digital logic, since these implement combinational logic functions over multiple clocks and require complex control circuitry. In contrast, Memrisrtor Ratioed Logic (MRL) is compatible with static CMOS logic due to its single-clock operation and its use of voltage levels for logic-0 and logic-1. Hence, MRL allows memristor-only logic cells — namely AND and OR — to be integrated with static CMOS logic cells to obtain hybrid memristor-CMOS logic blocks which have similar delays and lower area, compared to conventional CMOS logic.

978-1-6654-1060-1/22/$31.00 ©2022 IEEE

In MRL logic family, AND and OR are the only two basic memristor-only logic gates and CMOS inverters are used to implement logic inversion. CMOS inverters are also used to restore voltage levels. As is common for new devices and logic families, MRL faces new electrical challenges which degrade circuit operation, and can cause logic errors, especially when large numbers of memristor-only MRL gates (AND and OR) are *directly cascaded*, i.e., without inserting CMOS inverters, or other CMOS gates, between them.

This paper presents a systematic analysis of two major logic failure mechanisms in memristor-only MRL logic blocks, i.e., blocks where MRL AND and OR gates are directly cascaded. First, some memristor-only MRL logic blocks can produce incorrect logic values due to degradation of voltages. This failure mechanism is named *cascading failures*. While previous research has identified such failures [11], [12], only some examples of MRL blocks with logic errors are reported and general methods for verification and test of any given block are not provided.

Second, a race condition occurs when series-connected voltage controlled memristors with the same polarity switch their states from ON to OFF. We call this failure mechanism as *race failures* and present the first study of such failures at logic-level. To the best of our knowledge, the only previous research on this are [13]–[15] which observe this race condition when studying devices and consider its effects during the derivation of the equivalent circuit model for two memristors in series.

In this paper, we carry out the first systematic study of these two memristor-specific logic failure mechanisms. We identify properties of such failures in memristor-only MRL logic blocks in terms of structure of the circuit and the patterns applied. We then use these properties to generate patterns for each failure mechanism for pre-fabrication verification of memristor-only MRL logic blocks. We demonstrate that our methods generate small numbers of patterns and invoke the worst-case for each failure mechanism.

We start with a brief overview of memristor devices and MRL family in Section II. The underlying reasons that cause voltage degradation and logic failures in memristor-only MRL logic blocks are presented in Section III. Section IV and V describe the proposed methods to generate test patterns

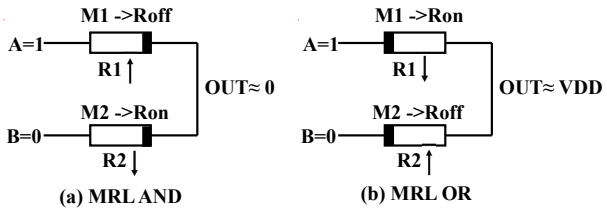

Fig. 1. Operation of MRL AND, OR gates

to verify the logic-level correctness of memristor-only MRL blocks against these two types of memristor-specific logic failures. Finally, Section VI concludes the paper by summarizing results and ongoing and future research opportunities and challenges. In particular, we share our plans for extending these approaches for post-fabrication testing of chips with MRL logic blocks by considering process variations and defects, and for tackling the major challenge posed by the non-deterministic behavior caused by race failures.

II. Background: Memristor model and MRL logic

For a memristor, the state or the resistance (in the range [Ron, Roff]) depends on the voltage applied across the device or the charge passing through it. In this paper, we study *voltage controlled* memristors with voltage thresholds. The resistance of such a device changes when a voltage beyond a threshold is applied (Von and Voff). Here, we use MOS transistors from TSMC 65 nm technology and the VTEAM model [16] for memristors with parameter values shown in Table I [17], [18]. Further, we use VDD = 1.2V and GND = 0V.

For MRL AND and OR gates [17], memristors are connected as shown in Fig. 1. (In both gates, an input-to-input path passes via two memristors in series with opposite polarities.) When identical logic values are applied to both inputs (both high or both low) no current flows via the memristors and hence the values of memristances do not change. (This is true for the AND as well as the OR gate.) Despite this, the output voltage is equal to the voltage applied to the inputs. For MRL AND gate with different logic values at its two inputs, the resistance of the memristor with the controlling value (0) decreases, i.e., approximately becomes Ron, and the resistance of the memristor with non-controlling value (1) increases, i.e., approximately becomes Roff. Since Roff$>>$Ron, the output is approximately 0 for AND gate. (The MRL OR gate's operation is a dual of the above.)

Since memristor is a passive device (i.e., has no gain), it is not possible to build NOT gate from only memristors. Hence CMOS inverters are used to complete the logic family as well as to restore voltages at various lines in a logic circuit.

TABLE I
VTEAM Fitting parameters for a memristor [17]

Parameter	Value	Parameter	Value
R_{on}	1 KΩ	p	2
R_{off}	200 KΩ	α_{on}	3
D	3 nm	α_{off}	3
K_{on}	-0.0162 m/s	V_{on}	-0.16 V
K_{off}	0.0162 m/s	V_{off}	0.16 V
x_{on}	0 nm	x_{off}	3 nm

III. Memristor-specific failures in MRL logic

Although several works [17], [19] have proposed design of digital circuits by cascading MRL AND and OR gates directly (i.e., without inserting a CMOS gate in between), the output voltage for directly-cascaded MRL AND and OR logic gates degrades and logic error occurs in some such circuits. We study two types of memristor-specific logic failures.

A. Cascading failures

As shown in Section II, a single multi-input MRL AND/OR gate provides logically correct output voltages. Despite this cascading MRL AND and OR gates directly may not always be viable [11], [12]. For example, a 2-input AND gate cascaded with a 3-input OR gate (shown in Fig. 2) fails to produce correct logical output for the pattern (1100). Here, if the 2-input AND gate was an isolated gate, for input pattern (11) we would get a logic 1 (and perfect voltage, i.e., VDD) at the output and the two memristors in the AND gate would not go through any changes in memristance. But when this 2-input AND gate is cascaded with the 3-input OR gate, the memristors in the AND gate now interact with the memristors in the OR gate. Due to the resulting current flow (assuming that memristances change in the expected direction), for the parameter values shown in Table I, M1, M2, M4, and M5 become OFF and the internal node X becomes slightly higher than $\mathrm{VDD}/2$. As node X is an input to the OR gate, the voltage at the output (OUT) is further degraded. Instead of the expected high output voltage, the voltage at the final output (OUT) is below $\mathrm{VDD}/2$. This demonstrates that this cascading structure is not logically correct.

To avoid above interactions between the memristors in the two gates, and for restoring the voltage level, it is necessary to insert a CMOS inverter between them.

We systematically study such interactions and develop models of such failures in terms of the structure of the circuit and the patterns applied. The goal is to develop methods to generate patterns for any given memristor-only MRL logic block to verify its logical correctness. Simulation of the patterns we generate, using the given memristor parameters, will verify whether the block will work correctly or require re-design.

B. Race failures

In an memristor-only MRL logic block, memristors interact with each other and change in specific directions to provide

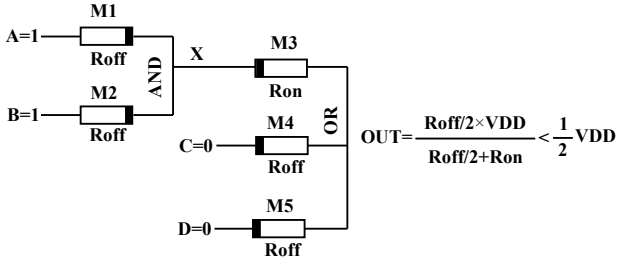

Fig. 2. Incorrect output voltage for a 3-input MRL OR gate cascaded with a 2-input MRL AND gate

the correct output voltage. But if the memristors are voltage controlled (we use VTEAM [16] model) and are connected in series with the same polarity, they *exhibit non deterministic behaviour*. This can cause a serious effect on cascaded MRL logic gates. We first study the behaviour of series connected memristors with the same polarity.

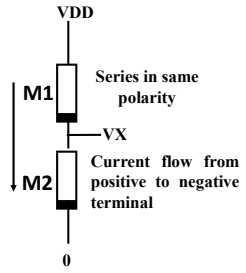

Fig. 3. Non deterministic behavior of two series connected same polarity memristors switching from ON to OFF state

As shown in Fig. 3, memristors M1 and M2 are in series with the same polarity and, for the voltages applied to the circuit, both are trying to switch from ON to OFF state. In a perfectly matched case, the resistance values of both memristors change at exactly equal rates and the final memristance will be Roff for both.

However, in practice, the perfectly matched case is highly unlikely for three reasons. First, process variation during fabrication is likely to create minor differences between the parameter values for the two memristors, e.g., Ron, Voff or Koff. Second, even if we have zero process variations, a mismatch between the initial states (resistance values) of the memristors is likely, as these depend on previous pattern(s) and the resistances are not always exactly Ron. Third, as we illustrate in detail ahead, in certain circuit configurations, one of the two memristors in the race may be in fact an equivalent circuit for 2+ memristors in parallel while the other memristor may be a single memristor. Clearly, in such a case there will be significant mismatch between the parameter values.

Consider a scenario where, due to any of the above reasons, memristor M2 is switching faster than memristor M1. In this scenarios, V(M2)>V(M1) and as the switching rate depends on the applied voltage [16], resistance of M2 changes faster and faster. Eventually, after some time, M2 becomes Roff and M1 lags behind at some intermediate resistance Rx (between Ron and Roff). Two possible cases arise.

Case-1: Voff $> \frac{\text{Rx}}{\text{Rx+Roff}}$VDD, M1 cannot switch further and stabilizes at an intermediate resistance. (Voff constraint case)

Case-2: Voff $< \frac{\text{Rx}}{\text{Rx+Roff}}$VDD, M1 will also switch to Roff if sufficient time is allowed, which may not be possible for a high speed circuit.

This non-deterministic behaviour of a memristor's resistance value can have serious effect on MRL logic.

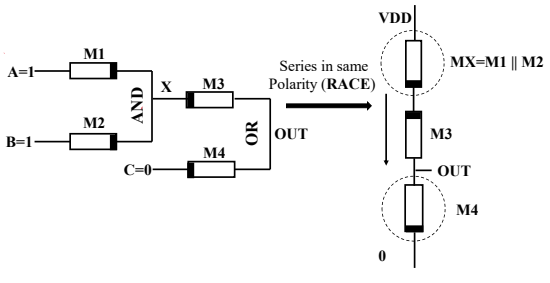

Fig. 4. Effect of race phenomenon on cascaded MRL gate

As shown in Fig. 4, a 2-input MRL AND gate is directly cascaded with a 2-input MRL OR gate. When the input pattern (110) is applied, memristors M1, M2, and M4 are *expected* to become Roff and M3 is *expected* to become Ron. Thus, the *expected* output voltage is $\frac{\text{Roff}}{\text{Ron}+\frac{3}{2}\text{Roff}} \times \text{VDD} \approx \frac{2}{3}\text{VDD}$ which can be considered as logically high.

Now consider a case where M1, M2, and M4 were initially in ON state when the input pattern (110) is applied. Memristors M1 and M2 are parallel for these inputs and their equivalent memristance is MX. MX and M4 act like two series-connected memristors with the same polarity, and both are trying to switch their states from ON to OFF. As MX is the equivalent memristor model for two memristors in parallel while M4 is a single memristor, their ON state resistance will be different. Hence, a race is likley to occur between MX and M4. If M4 switches its state to Roff, then the circuit will produce correct logic value output. However, in the other situation, where MX switches faster, M4 is not guaranteed to switch all the way to Roff. Say at t = tx, MX finishes switching and M4 becomes Rx (consider it as a small value). As discussed in (case-2), if Voff $> \frac{\text{Rx}}{\text{Rx+Roff/2+Ron}}$VDD, M4 is unable to switch and stabilizes at Rx. Hence, under this particular condition, the output voltage is evaluated as: $\frac{\text{Rx}}{\text{Ron+Rx+Roff/2}}$VDD, which is no longer a high output voltage. In other words, the output logic fails in this directly cascaded structure due to a race.

IV. Verification of MRL logic blocks against cascading failures

It has already been demonstrated that certain cascading structures of MRL AND and OR gates are not feasible without CMOS gates inserted between them. In a given digital MRL logic block (an example block shown in Fig. 5), memristor-only blocks (M_i-blocks) are connected via CMOS gates (C_j-blocks), where the CMOS gates are required to implement logic inversion logic, or to restore voltages, or to provide electrical isolation. Since M_i-blocks contain only two types of memristor-only gates, namely AND and OR, these do not have reconvergent fan-outs. Without any loss of generality, we consider all M_i as being fanout-free circuits. In this section we assume that each memristor changes its state completely according to the direction of the current. To enable this, in this section, we use the parameter values in Table I for evaluations but reduce the values of Von and Voff. (In the next section, when we study race, we use the parameter values as shown in Table I.)

The method we describe here targets any given M_i-block and generates patterns for creating worst case voltages at each line in the block. To extend this method for verification of the complete circuit, where only primary inputs and outputs of the entire MRL block are considered, justification of the patterns from inputs of the target M_i-block and propagation of logic error (if any) from the output of the M_i-block is required.

A given memristor-only block should be verified via simulation of input patterns that cause the most degraded output voltages and check whether this degradation is acceptable. Exhaustive simulation for all possible patterns is expensive.

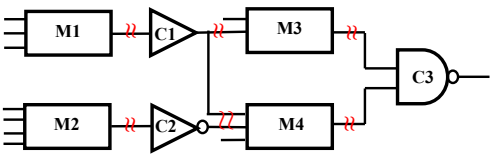

Fig. 5. Partitioning of MRL logic block in memristor-only MRL blocks (M_i-blocks) and CMOS blocks (C_j-blocks)

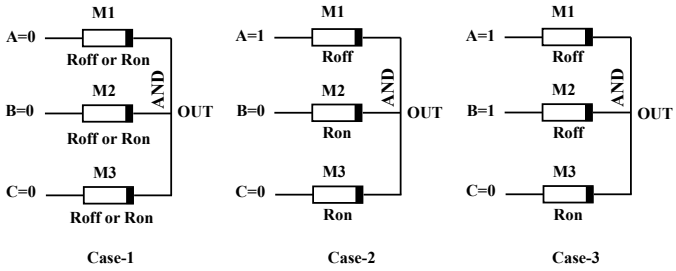

Fig. 6. Output voltage behaviour of a 3-input MRL AND gate under different applied patterns

Hence, here we propose a method to generate patterns that will cause the weakest output voltage in a given memristor-only MRL logic block. In the context of MRL logic, *weak 1* means a degraded voltage which is lower than VDD and *weak 0* means degraded voltage which is higher than 0V.

The degradation of output voltage in any MRL circuit depends on circuit configuration and also on the pattern applied to the circuit. We introduce the key properties using a 3-input AND gate. The set of all possible 8-input patterns is divided into 4 sets: S1={000}, S2={100, 010, 001}, S3={101, 011, 110}, and S4={111}. The output logic is 0 for the patterns in S1, S2, and S3. For the pattern in S1 (Case-1 in Fig. 6), the output voltage is perfect 0 and doesn't depend on the states of the memristors. Any pattern in S2 (Case-2 in Fig. 6) results in slightly degraded output voltage: $\frac{\text{Ron/2}}{\text{Ron/2+Roff}} \times \text{VDD}$. For any pattern in S3, the output voltage is $\frac{\text{Ron}}{\text{Ron+Roff/2}} \times \text{VDD}$ (Case-3 in Fig. 6) which is the weakest 0 output.

Since we are studying cascaded gates, some or all of the inputs A, B, C are driven by outputs of other MRL gates. These driving gates can be asymmetric in structure as shown in Fig. 8. In this context, consider the case where our objective is to derive an input pattern that causes the worst case output voltage degradation for the AND gate, i.e., OUT=0_w, where *0_w denotes logic 0 that is as weak as possible*. (Also, 0_s denotes logic 0 as strong as possible and 1_w and 1_s are duals.)

As we want to determine the required primary inputs, we start traversing backward from line OUT and perform the next task, namely determine the values at lines X, C, and D (XCD) to satisfy our primary objective, namely OUT=0_w. We first understand the strengths of the input patterns as shown in Fig. 7. It can be inferred from superposition law that (XCD) being (011) produces weaker 0 at output than (XCD) being (010) and (001). But as the circuit is asymmetric, we cannot derive any such relationship between every pattern in S3 and every pattern in S2. For example, we cannot say that (101) produces weaker 0 than (010) for an arbitrarily asymmetric circuit structure. It is evident from Fig. 7 (a), for AND gate

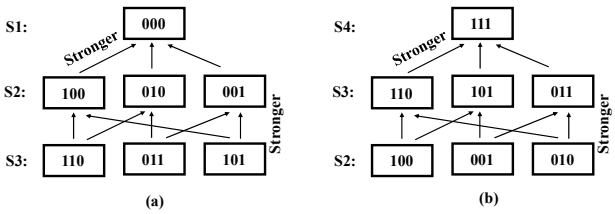

Fig. 7. Input pattern strength relationship for 3-input MRL (a) AND gate producing 0, and (b) OR gate producing 1, considering asymmetric sub-circuits driving their inputs

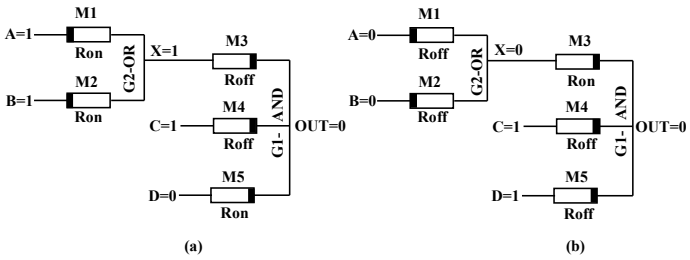

Fig. 8. Deriving verification input patterns for 3-input AND gate cascaded with a 2-input OR gate

at the output, irrespective of the circuit structure, the *weakest 0 will be caused by one of the three patterns from S3={101, 011, 110}*. In our example circuit, for weakest 0 at OUT, XCD value has two possible patterns: {110, 011}, since we can reduce one pattern because C and D legs are symmetric.

For intermediate lines, such as X in this example, the voltage can again be strong or weak depending on the inputs applied to the inputs of the previous gate. As our primary objective is to obtain the weakest 0 at OUT, any intermediate line should be made as strong as possible if it is 1 and as weak as possible if it is 0. Hence, we specify our next objective as Objective$_2$: (XCD)={$1_s1_s0_w$, $0_w1_s1_s$}. As C and D are primary inputs, no further analysis is needed for those values. For the internal line X, A and B are both assigned 1 to satisfy X=1_s (Fig. 7 (b) shows the strength relationship). X=0 can only be satisfied by setting A=0, B=0. Hence, our proposed algorithm generates (ABCD)={1110, 0011} as the set of two input patterns that should be applied for worst case analysis. This algorithm is outlined next.

An outline of the algorithm for generating test patterns to verify for cascading failures

Initialize
Set Logic value at all circuit lines to Unknown
Set primary Objective:
If output line l driven by AND gate, $v = 0_w$
else if l driven by OR gate, $v = 1_w$
T = Assign_and_Justify (line l to value v)
Print test set T

Assign_and_Justify(line l to value v)
From Table II, identify new objectives for the fan-in lines of the gate driving line l
This provides a set of patterns for the fan-in lines
Assign_and_Justify each value at all circuit lines recursively until values assigned at primary inputs
Combine returned patterns using cross-product and return test set

For this 4-input circuit, waveforms for exhaustive simulation

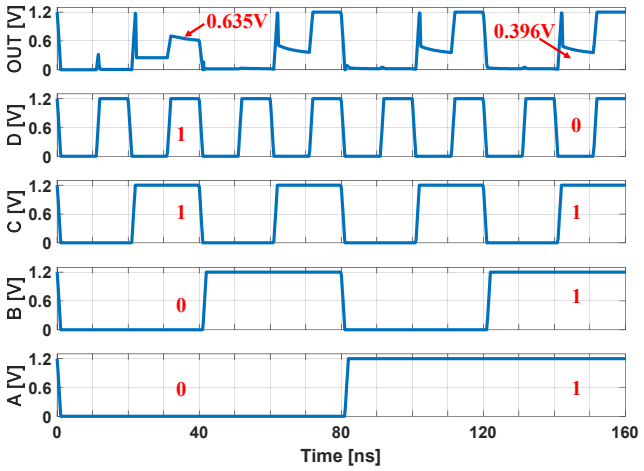

Fig. 9. Output voltage of the 3-input MRL AND gate cascaded with a 2-input MRL OR gate shown in Fig. 8

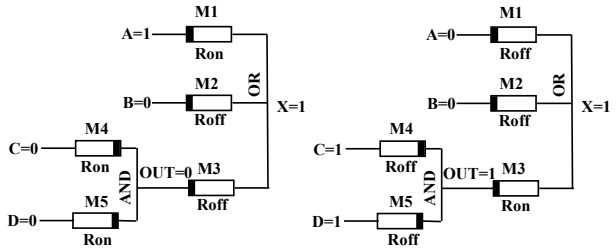

Fig. 10. Worst case analysis of internal node X in Fig. 8

over all possible 16 patterns are shown in Fig. 9 and 1110 indeed causes the worst case output voltage level (For this simulation, we have used parameter set from Table I with very small Von, Voff.)

This algorithm can also be applied to generate patterns for worst case analysis for any internal line in the circuit (say X in Fig. 8). As shown in Fig. 10, the circuit is folded around X and the primary objective is set as: X=1_w. Here, since two different types of gates are cascaded at node X, in the folded circuit, the output X behaves like the output of an OR gate with an extra input added to the original OR gate.

In contrast, if two gates of the same type are cascaded (Fig. 11), folding creates an extra memristor leg with a fixed resistance (Ron+Roff). This extra fixed-resistance leg can again be analyzed as one extra input added to the original gate. As this is a fixed-resistance leg, we apply a logic value to this leg to make the output weaker according to the superposition principle. For example, in a 3-input OR gate input pattern for weakest 0 at output is (100). So for a 3-input OR gate with an extra fixed-resistance leg, the input pattern is (1000).

When an intermediate node located k levels earlier than the output node is analyzed, we have k folding sites in the circuit (one folding at each level). As at every level there are only four possible types of cascading: AND-AND, AND-OR, OR-AND, and OR-OR, only above two scenarios will occur. Hence, our above properties are sufficient. We use the above properties and folding to generalize our above algorithm to create patterns for verification of cascading failures at all lines.

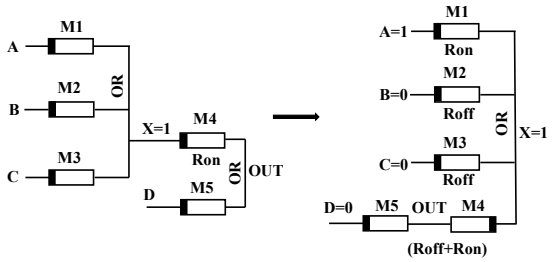

Fig. 11. Worst case analysis of an internal node X in a 2-input OR gate cascaded with a 3-input OR gate

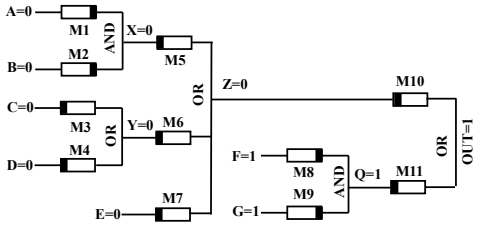

Fig. 12. Deriving verification and test pattern for an example memristor-only MRL block

We apply our above algorithm to a larger circuit shown in Fig. 12. It generates the following minimum set of test patterns guaranteed to cause the worst case 1 (i.e., 1_w) at the primary output (OUT): (ABCDEFG)={0000011, 0010000, 1100000, 0000100}. Simulations show that the circuit fails for 1100000. For this circuit, testing with 4 patterns is guaranteed to detect cascading failure at the output, compared to 2^7=128 patterns for the exhaustive method. Hence, our method significantly reduces the cost of verification while ensuring high quality.

V. Verification for Race Failures

Two series memristors in the same polarity changing from ON to OFF create a race, a condition that occurs when different types of gates (AND and OR gate in MRL) are cascaded in the circuit. Whether there will be a failure due to race or not depends on circuit structure, the applied pattern, the voltage threshold value, initial states, and the dynamic behaviour of the memristors. The initial ON and OFF values of memristors fall in a range of values rather than being perfect Ron or Roff as shown in Table. I. Let us consider the range of memristor ON values as [Ron_{min}, Ron_{max}].

Now we *generate an input pattern plus initial states that activate the race failure at the output node of a memristor only*

TABLE II
Determining New Objectives

Type of gate	Objective for gate output line	Set of new objectives for gate input lines
AND	1_s	$\{(1_s, 1_s, 1_s)\}$
AND	0_s	$\{(0_s, 0_s, 0_s)\}$
AND	1_w	$\{(1_w, 1_w, 1_w)\}$
AND	0_w	$\{(1_s, 1_s, 0_w), (1_s, 0_s, 1_s), (0_w, 1_s, 1_s)\}$
OR	1_s	$\{(1_s, 1_s, 1_s)\}$
OR	0_s	$\{(0_s, 0_s, 0_s)\}$
OR	1_w	$\{(1_w, 0_s, 0_s), (0_s, 1_w, 0_s), (0_s, 0_s, 1_w)\}$
OR	0_w	$\{(0_w, 0_w, 0_w)\}$

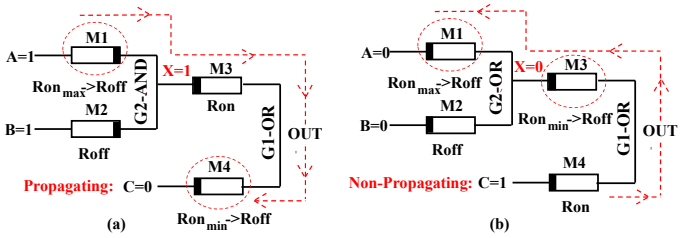

Fig. 13. Pattern to verify against race failure for (a) 2-input MRL OR gate cascaded with a 2-input MRL AND gate (b) 2-input MRL OR gate cascaded with a 2-input MRL OR gate

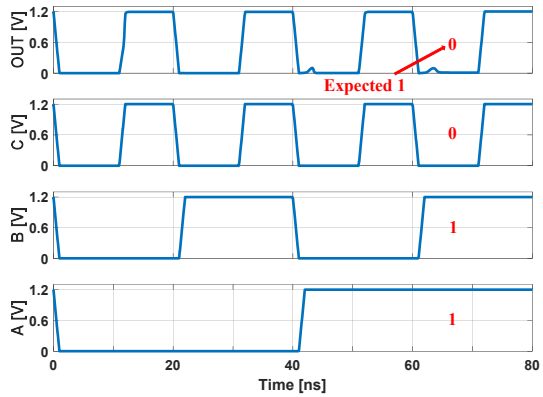

Fig. 14. Output voltage of the 2-input MRL OR gate cascaded with a 2-input MRL AND gate shown in Fig. 13 (a)

MRL gate shown on the next page. (The cascaded circuits in Fig. 13 is used as an example.)

An outline of the algorithm for generating input test pattern and initial states to verify for race failure
For every input-input current path (P_i)
For every memristor-pair (M_j, M_k) w/ the same polarity in P_i
Assign values to inputs of P_i to produce a current that causes (M_j, M_k) to switch from ON to OFF state.
Assign initial states (Ron$_{min}$, Ron$_{max}$) to (M_j, M_k)
For other memristors initial states can be considered as Roff
Justify for error propagation and create an input pattern

For example, in Fig. 13(a), M1 and M4 are targeted in the A to C input-input path. Inputs (AC) are assigned (10) to create the required current path. To propagate the error, B is assigned 1. This input condition satisfies error propagation to the output and so input pattern (ABC)=(110) is generated. In a similar fashion, M1 and M3 are targeted in Fig. 13(b) but no pattern is generated for the targeted memristors in this input-input path as C=1 is the controlling input for the OR gate and error cannot be propagated.

As race failure depends on the pattern as well as the initial states, many published memristor based designs fail verification for this type of logic failure. The full adder proposed in [17] has a race failure for the memristor parameter set given in [17]. As shown in Fig. 15, M1 || M3 and M2 can cause a race for the input pattern A=1, B=0. We use M1(initial)=Ron$_{max}$=6 kΩ, M3(initial)=200 kΩ and M2(initial)=Ron$_{min}$=1 kΩ. (These values are determined in a pre-processing step.) The input pattern (AB)=(10) causes M1 to switch faster than M2 and the output of the XOR gate is 0

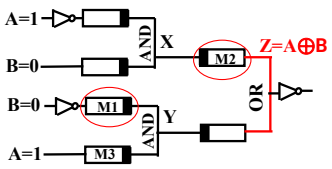

Fig. 15. Race failure in the XOR gate proposed in [17] for a full adder

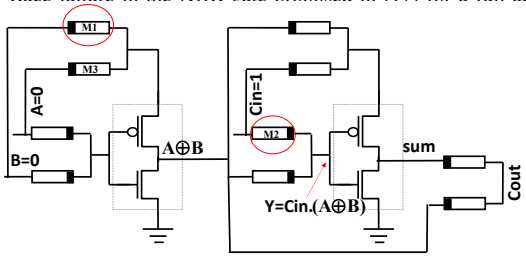

Fig. 16. Race failure in the adder proposed in [20]

due to a race failure.

The adder proposed in [20] also produces incorrect logical value at line Y for memristor parameters in Table. I for input pattern (A B Cin)=(001) and initial states- M1(initial)=6 kΩ, M3(initial)=200 kΩ and M2(initial)=1 kΩ. This shows that our method is needed even when we use published circuits in our designs.

VI. Conclusion

In this paper, we systematically studied two memristor-specific logic failure phenomena and identified approaches to generate patterns for these failure modes for memristor-only MRL blocks in MRL logic. Specifically, we propose systematic methods to generate reduced set of patterns to (a) invoke the worst-case voltage at output, and (b) activate race failures. Our methods provide a failure-directed approach for identifying a small yet effective set of patterns for pre-fabrication verification.

We have developed core methods that analyze the structure of the circuit, the properties of these failure types, and generate patterns for verification for specific targets. One additional key innovation is that our method for race failures also generates initial states of memristors.

This research has exposed several additional opportunities and challenges. Due to non-deterministic nature of race failures, values at internal lines must be monitored during simulation, by adding assertions to check specific properties of waveforms. In terms of verification, we are studying the impact of the simultaneous excitation of both these failure modes by a single pattern. Systematic generation of all possible targets for each failure mechanism and methods to compact the set of generated patterns must also be developed. Finally, for race condition, use of initial states may lead to pessimistic evaluation, since the initial state we generate may not be achievable during normal circuit operation. This requires extension of our approach to generate a sequence of one or more patterns to be applied before the pattern we currently generate, where the goal of the additional sequence will be to generate achievable worst-case initial state.

REFERENCES

[1] D. E. Nikonov and I. A. Young, "Benchmarking of Beyond-CMOS Exploratory Devices for Logic Integrated Circuits," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 1, pp. 3–11, Dec. 2015.

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

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

[4] I. E. Ebong and P. Mazumder, "Self-Controlled Writing and Erasing in a Memristor Crossbar Memory," *IEEE Transactions on Nanotechnology*, vol. 10, no. 6, pp. 1454–1463, Nov. 2011.

[5] H. Kim, M. P. Sah, C. Yang, and L. O. Chua, "Memristor-based multilevel memory," in *2010 12th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2010)*, Feb. 2010, pp. 1–6, iSSN: 2165-0152.

[6] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "MAGIC—Memristor-Aided Logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.

[7] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in *2009 IEEE/ACM International Symposium on Nanoscale Architectures*, Jul. 2009, pp. 33–36, iSSN: 2327-8226.

[8] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, "MRL — Memristor Ratioed Logic," in *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*, Aug. 2012, pp. 1–6.

[9] A. Ascoli, R. Tetzlaff, F. Corinto, M. Mirchev, and M. Gilli, "Memristor-based filtering applications," in *2013 14th Latin American Test Workshop - LATW*, Apr. 2013, pp. 1–6, iSSN: 2373-0862.

[10] G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, and T. Prodromakis, "Integration of nanoscale memristor synapses in neuromorphic computing architectures," *Nanotechnology*, vol. 24, no. 38, p. 384010, Sep. 2013.

[11] S. Yamtim and S. Tooprakai, "Full Adder Circuit using Multi-Input MRL," in *2020 6th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, Jul. 2020, pp. 1–4.

[12] X. Wang, R. Yang, Q. Chen, and Z. Zeng, "An improved memristor-CMOS XOR logic gate and a novel full adder," in *2017 Ninth International Conference on Advanced Computational Intelligence (ICACI)*, Feb. 2017, pp. 7–11.

[13] R. K. Budhathoki, M. P. Sah, S. P. Adhikari, H. Kim, and L. Chua, "Composite Behavior of Multiple Memristor Circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 10, pp. 2688–2700, Oct. 2013.

[14] X. Hu, G. Feng, L. Liu, and S. Duan, "Composite Characteristics of Memristor Series and Parallel Circuits," *International Journal of Bifurcation and Chaos*, vol. 25, no. 08, p. 1530019, Jul. 2015.

[15] P. Li, X. Zhang, J. Eshraghian, H. Ho, C. Lu, and X. Wang, "Spice modelling of a tri-state memristor and analysis of its series and parallel characteristics," *IET Circuits, Devices and Systems*, Jun. 2021.

[16] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "VTEAM: A General Model for Voltage-Controlled Memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015.

[17] K. A. Ali, M. Rizk, A. Baghdadi, J. Diguet, and J. Jomaah, "MRL Crossbar-Based Full Adder Design," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Nov. 2019, pp. 674–677.

[18] H. Y. Lee, Y. S. Chen, P. S. Chen, P. Y. Gu, Y. Y. Hsu, S. M. Wang, W. H. Liu, C. H. Tsai, S. S. Sheu, P. C. Chiang, W. P. Lin, C. H. Lin, W. S. Chen, F. T. Chen, C. H. Lien, and M. Tsai, "Evidence and solution of over-RESET problem for HfOX based resistive memory with sub-ns switching speed and high endurance," in *2010 International Electron Devices Meeting*, Dec. 2010, pp. 19.7.1–19.7.4, iSSN: 2156-017X.

[19] G. Liu, L. Zheng, G. Wang, Y. Shen, and Y. Liang, "A Carry Lookahead Adder Based on Hybrid CMOS-Memristor Logic Circuit," *IEEE Access*, vol. 7, pp. 43 691–43 696, 2019.

[20] M. Teimoori, A. Ahmadi, S. Alirezaee, and M. Ahmadi, "A novel hybrid CMOS-memristor logic circuit using Memristor Ratioed Logic," in *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, May 2016, pp. 1–4.