

# A Novel “Divide and Conquer” Testing Technique for Memristor based Lookup Table

Veeresh A. Hongal<sup>1</sup>, Raghavendra Kotikalapudi<sup>2</sup>, Yong-Bin Kim<sup>3</sup> and Minsu Choi<sup>1</sup>

<sup>1</sup>Department of Electrical and Computer Engineering, {vah284,choim}@mst.edu

<sup>2</sup>Department of Computer Science, rkyvb@mst.edu

Missouri University of Science & Technology, Rolla, MO 65409, USA

<sup>3</sup>Department of Electrical & Computer Engineering, ybk@ece.neu.edu

Northeastern University, Boston, MA 02115, USA

**Abstract**—The recently proposed nanoscale asynchronous crossbar architecture based on memristor-based look up table (MLUT) combines the advantages of memristor technology and asynchronous design for viable nanoscale computing. In spite of having numerous merits over the clocked counterparts and previous asynchronous designs, it is bound to have inevitable defects due to nondeterministic nanoscale assembly. In order to assess the reliability of MLUT, there is a need to develop efficient testing techniques. Typical approach so far has been to test every crosspoint on each crossbar MLUT exhaustively; this is not only tedious but is also prohibitively time consuming for designs involving large number of MLUTs. This paper introduces a novel testing scheme based on “Divide and Conquer” approach to efficiently locate the defective memristors in a MLUT. The proposed testing scheme leverages upon a special current additive property of the memristor based multiplexer. It performs binary isolation of regions, reducing the search space by half whenever applicable. Numerical simulations clearly demonstrate that the approach is generic, deterministic, and scalable.

## I. INTRODUCTION

Recent research trends show an increase in popularity of asynchronous designs for FPGA. This is due to the fact that elimination of the clock solves many timing related issues such as clock skew and race conditions. The recently proposed Asynchronous Nanowire Crossbar Architecture [1] uses Memristor LUT, formed by an array of memristors [2], [3], as its building block and works on the principle of NCL [4] (Null Conventional Logic) - an asynchronous paradigm - thereby bringing asynchronous LUT designs to nanoscale regime. In spite of these advantages, due to the non-deterministic nature of nanoscale crossbar assembly, a defect rate of up to 10% is anticipated in nanoscale crossbar assembly [5]. In order to develop reliable circuits at the nano level, it is important to develop efficient testing methodologies.

Since the introduction of the memristor based crossbar arrays [6], very little work has been done in realizing efficient testing scheme(s). A popular approach to test MLUT is the Raw testing scheme, involving exhaustive checking of every crosspoint for defects [1]. However, exhaustive checking is tedious and is prohibitively time consuming for designs involving large number of MLUTs. Although an efficient testing scheme for Asynchronous Nanowire Crossbar has been proposed before, it is probabilistic in nature and is specific to

diode based LUTs [7]. Such a scheme cannot be applied to the MLUT as the memristors follow a one to one correspondence with every test tuple input. In this paper, we introduce an efficient testing scheme for MLUT, by utilizing the special current additive property of a memristor based multiplexer.

The paper makes multiple contributions to the field of memristor based testing: 1) The ability to select variable regions within the MLUT - utilizing the special property of memristor based multiplexors - is novel and could lead to innovative applications; 2) The proposed Divide and Conquer testing methodology, not only applies to MLUT testing, but can also be applied to any memristor based crossbar designs in general; 3) Unlike previous probabilistic testing approaches, the proposed technique is deterministic in nature; 4) The proposed scheme is not only generic, but also has a good scaling behavior.

## II. DEFECTS IN MLUT

Defects in a nanowire crossbar can either occur due to nanowire faults or switch faults [5]. In this paper, we focus on identifying switch faults due to defective memristors in the crossbar array since nanowire shorts and breaks are easily tested and detected.

The memristor uses a very thin film of  $\text{TiO}_2$  sandwiched between two Pt contacts. The  $\text{TiO}_2$  film contains two regions - a high conductance doped region and a high resistance undoped region - as illustrated in Fig. 1. When a positive voltage is applied across the device, the dopants drifts towards the undoped region, increasing the proportion of the conductive region. Similarly, the application of negative voltage increases the resistance [3].

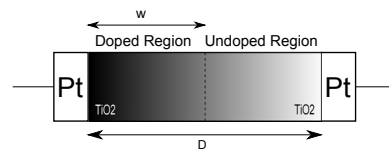


Fig. 1: Memristor Structure

Due to the memristor’s dependence on doping, the following defects can arise:

1) *Stuck at 1 defect (SA1)*: If  $\text{TiO}_2$  is doped excessively with positively charged oxygen vacancies, the memristor remains stuck in the ON state, even when a negative voltage - within the operating range - is applied across it.

2) *Stuck at 0 defect (SA0)*: This is analogous to SA1 defect. Due to the deficiency of the positively charged vacancies, the memristor remains stuck in the OFF state, even when a positive voltage is applied across it.

3) *Non Programmable Defects (NPD)*: A Stuck at defect occurs when the proportion of doped/undoped region is *largely* greater than the other region. On the other hand, a Non Programmable Defect occurs when one region is *slightly* greater than other region, resulting in one the following behaviors:

a) When the doped region is slightly greater than the undoped region, even though the positive voltage switches the memristor to the ON state, the negative voltage - within the operating threshold - may not be sufficient to switch it to the OFF state. Being analogous to SA1 defect, we call this NPD1 defect.

b) On the same lines, NPD0 is defined when the proportion of undoped region is slightly greater than the doped region.

Considering the upper and lower threshold cutoff currents  $I_{on}(=0.8I_{max})$  and  $I_{off}(=0.2I_{max})$ , where  $I_{max}$  is the current flowing through a completely doped memristor, Table I shows how Test 1 and Test 2 can be used to distinguish between SA1, SA0, and Non Programmable Defects.  $I_1$  and  $I_2$  are the output currents when the memristor is programmed to ON and OFF state respectively.

TABLE I: Tests on memristors with different types of defects

Defect Type	$I_1$	Test 1 ( $I_1 \geq I_{on}?$ )	$I_2$	Test 2 ( $I_2 \leq I_{off}?$ )
Ideal	$0.9I_{max}$	Pass	$0.1I_{max}$	Pass
SA0	$0.1I_{max}$	Fail	$0.1I_{max}$	Pass
SA1	$0.9I_{max}$	Pass	$0.9I_{max}$	Fail
NPD0	$0.6I_{max}$	Fail	$0.1I_{max}$	Pass
NPD1	$0.9I_{max}$	Pass	$0.4I_{max}$	Fail

From Table I, we can observe that (SA0, NPD0) show similar characteristics and shall be referred to as SA0 defects. Similarly, (SA1, NPD1) will be referred to as SA1 defects.

### III. TESTING TECHNIQUE

Every memristor in MLUT follows a one to one correspondence with every test tuple input. With current and voltage as the only measurable quantities, one possible way to outperform the raw scheme is to measure multiple currents or voltages simultaneously across various memristors. This can be achieved by measuring the summation of individual memristor current outputs, capturing information about multiple memristors in a single measurement. In particular, by comparing the actual current summation  $I_{actual}$  with the ideal current summation  $I_{ideal}$  - the current if the region were completely free of defects - we can deduce whether or not a region contains defective memristor(s). By discarding unpromising regions

through current summations, we can converge quickly towards more promising regions of interest. Section III-A describes how the special property of the multiplexer can be used to compute current sums in a particular region. Section III-B details an efficient Divide and Conquer approach for converging towards promising regions. Sections III-C and III-D describes the application of the proposed Divide and Conquer approach to pre-programmed and post-programmed testing.

#### A. Special Property of Memristor-Based Demultiplexer

In an MLUT, the 1:4 demultiplexer comprising of a  $1 \times 4$  memristor array is normally used to select a single column by enabling the memristor corresponding to that column. However, multiple columns can be selected by simultaneously switching the corresponding memristors in a demultiplexer to the ON state. Similarly, multiple rows can also be selected simultaneously. Therefore, any region defined by the tuple  $\langle RowStart, RowEnd, ColStart, ColEnd \rangle$  can be selected. Furthermore, as all memristors in the selected region are parallel to each other, the same voltage appears across all the memristors. By measuring the current at the output, the summation of currents through all the individual memristors can be obtained. The current summation, however, suffers from the *sneak path* problem, but can be avoided using complementary resistive switches [8].

#### B. Divide and Conquer Testing Scheme

The algorithm begins by measuring the current sum of the entire crossbar in a single measurement, utilizing the special property described in the previous section. In the first iteration, by comparing  $I_{actual}$  and  $I_{ideal}$ , the total number of defects in the MLUT can be estimated. In the following iterations, the given region is split into two equal halves, each of which is examined recursively. The recursive procedure disregards regions that are free of defects, reducing the search space by half, whenever applicable. The general testing scheme for an  $m \times n$  crossbar is described in Algorithm 1.

Consider the example of an  $8 \times 4$  MLUT shown in Fig. 2 in which all the memristors are programmed to the ON state. Initially,  $I_{actual}$  for the entire circuit is calculated by selecting all rows and columns of the memristor. In this example, by observing the difference between  $I_{ideal}$  ( $8 \times 4 \times I_{on}$ ) and  $I_{actual}$ , the existence of a single defect is identified. Following the recursive procedure, group 2 is discarded from the search space as it is defect free. Continuing from group 3, the defective memristor is located in 4 more measurements, discarding half the search space in each iteration. In the end, examination of 23 memristors is avoided, effectively discarding 72% of the search space. As the scheme involves the application of voltage across different memristors, no additional control circuit complexity or area overhead is incurred when compared to the raw testing scheme.

#### C. Pre-programmed Testing scheme

In pre-programmed testing, all the memristors in the MLUT are programmed to the ON state; SA0 faults are identified by

**Algorithm 1** DAC\_Test: The proposed divide and conquer subroutine

```

1: Input: RS (Row Start), RE (Row End), CS (Column Start), CE (Column End), TestSA1
2: if TestSA1 then
3:    $I_{ideal} := (RE - RS) \times (CE - CS) \times I_{off}$ 
4: else
5:    $I_{ideal} := (RE - RS) \times (CE - CS) \times I_{on}$ 
6: end if
7:  $I_{actual} :=$  Measure current sum from the region enclosed in (RS, RE, CS, CE) using the special multiplexer property
8: NumDefects :=  $(I_{actual} - I_{ideal}) / (I_{on} - I_{off})$ 
// Base case of the recursion
9: if NumDefects = 1 and (CE - CS) = 1 and (RE - RS) = 1 then
10:  print Defect location found at (RE, CE)
11:  return
12: end if
// If defects exist, recurse..
13: if NumDefects > 0 then
14:  // Perform column wise split
15:  if (CE - CS) > 1 then
16:    DAC_Test(RS, RE, CS, (CS + CE)/2, TestSA1)
17:    DAC_Test(RS, RE, (CS + CE)/2, CE, TestSA1)
18:  // Perform row wise split
19:  else if (RE - RS) > 1 then
20:    DAC_Test(RS, (RS + RE)/2, CS, CE, TestSA1)
21:    DAC_Test((RS + RE)/2, RE, CS, CE, TestSA1)
22:  end if
23: end if

```

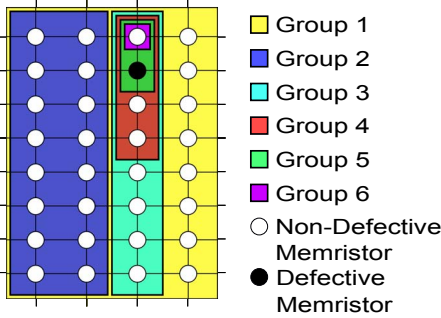


Fig. 2: Illustration of convergence process for the proposed Divide and Conquer approach

applying the Divide and Conquer testing scheme described in the previous section. SA1 faults are identified in a similar manner. Algorithm 2 lists the procedure for an  $8 \times 4$  MLUT.

**D. Post-programmed Testing scheme**

Post-programmed testing scheme is used to assess the reliability of pre-configured MLUTs. Unlike pre-programmed testing scheme, we cannot reconfigure the MLUT as it would destroy its current configuration.

The procedure for post-programmed testing is explained with an example of TH23 gate, illustrated in Fig. 3. Initially, rows with similar configuration(s) are grouped together. In the TH23 gate, Rows 1-2, 3-6, and 7-8 are separated into three groups. Within each group, the 0's and 1's are segregated to form the 0-plane and the 1-plane. Each 0-plane is tested for SA1 errors while the 1-plane is tested for SA0 errors.

By testing the 0-plane for SA1 errors, we ignore the presence of SA0 errors. Analogously, SA1 errors are ignored

**Algorithm 2** Preprogrammed\_Testing: The proposed pre-programmed testing scheme

```

1: Program all 32 memristors to OFF state
2: DAC_Test(0,8,0,4,true) // To test SA1 errors
3: Program all 32 memristors to ON state
4: DAC_Test(0,8,0,4,false) // To test SA0 errors

```

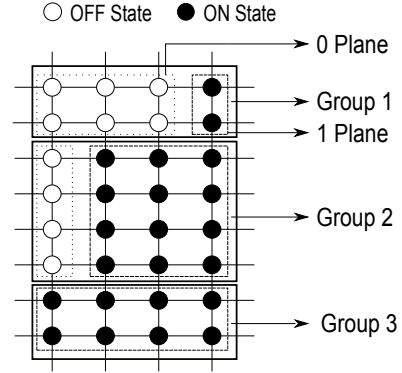


Fig. 3: Illustration of groups and planes in TH23 gate

in the 1-plane. For the post-programmed testing, this is not a concern as we can still assess the operational reliability of the pre-configured MLUT. A similar approach can be applied to test any of the 27 TH gates defined in NCL [4].

**IV. PERFORMANCE ANALYSIS**

In this section, we present the simulation results for comparing the performance of the proposed Divide and Conquer approach and the Raw scheme.

1) *Simulation Setup:* The number of measurements for both methods were evaluated by varying the defect rate from 0 to 10%. To obtain reliable estimates, current measurements for each defect rate was averaged over 100,000 runs, randomizing defect locations for each run.

In all the simulations, two ways of generating defects were adopted: 1) Mixed method; 2) Pure method. In the mixed method, a defect type is randomly chosen between SA0 and SA1 with equal probability, irrespective of the programmed state of the memristor while pure method selects SA1 defect for a memristor programmed to the OFF state and vice-versa.

2) *Pre-programmed testing:* Fig. 4 illustrates the comparison between the proposed pre-programmed testing scheme and the raw scheme. For pre-programmed testing, defects were generated using the mixed method as pure method does not apply. While the raw testing scheme always takes 64 iterations to locate the defects, the number of test iterations for our scheme grows logarithmically with increase in defect rates making it faster - especially for lower defect ranges. Even in the worst case, our scheme is 3.2 times faster.

3) *Post-programmed testing:* Post-programmed testing was conducted with both mixed and pure methods. Mixed method allows defect hiding, simulating real world scenario while the pure method is more rigid as it prevents defect hiding.

Fig. 5 shows the best, worst and average number of test iterations using both defect generation methods. To obtain a more



Fig. 4: Comparison of best, average, and worst number of test iterations in proposed scheme to Raw Testing Scheme

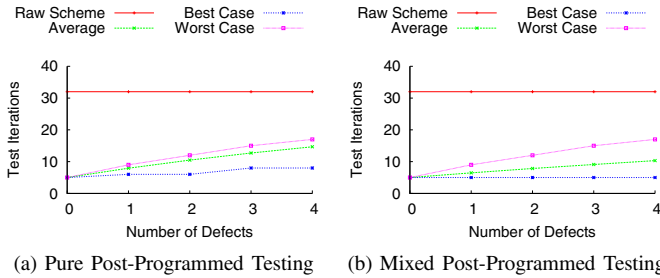


Fig. 5: Performance evaluation on TH23 gate

comprehensive overview, the average number of test iterations for all 27 NCL gates were evaluated and are summarized in Fig. 6. The whisker plot shows minimum, maximum, average, the upper and lower quantiles for defects varying from 1-4. From this, we observe that a logarithmic trend similar to pre-programmed testing is maintained for all the 27 NCL gates, demonstrating the flexibility of the proposed approach, even when applied to a programmed MLUT.

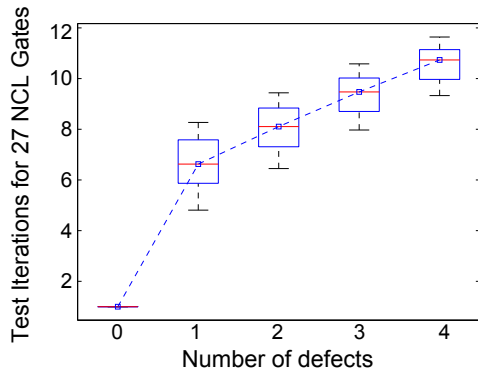


Fig. 6: Box and Whisker plot for all 27 NCL gates

4) *Scalability Study*: To evaluate the scalability of the proposed approach, we evaluate the number of measurements on different circuit sizes with the defect rate ranging from 0% to 10%. The circuit size is increased exponentially in powers of 2 up to 256. Experimental results show that with increasing

circuit size, the number of test iterations increase by the same factor, demonstrating a linear scaling behavior. Furthermore, our method identifies a defect free circuit in one measurement while the raw scheme iterates through the entire circuit. All numerical results are summarized in Fig. 7.

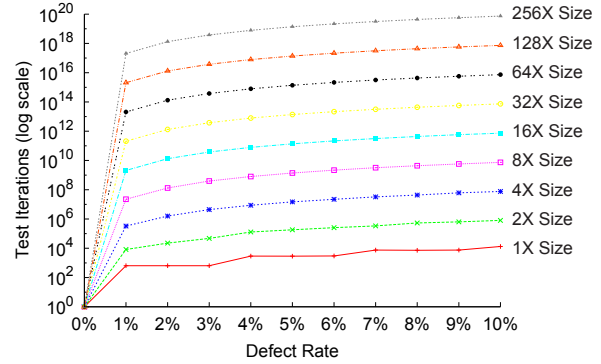


Fig. 7: Scaling Behavior of the proposed testing scheme

## V. CONCLUSION

This paper introduces a novel testing scheme based on Divide and Conquer approach to efficiently locate the defective memristors in an MLUT. By leveraging upon the special current additive property of the memristor based multiplexer, the proposed scheme outperforms the raw testing scheme. In particular, for 10% defect rate, numerical results show that our approach is nearly 3 times faster - both for pre-programmed and post-programmed testing. Scalability studies showed that the proposed testing scheme has a linear scaling behavior with respect to circuit size and a logarithmic scaling behavior with respect to defect rates. Unlike previous probabilistic approaches, our approach is deterministic in nature. Furthermore, the proposed approach is generic, i.e., it can also be applied to any memristor based crossbar array, even for pre-configured arrays.

## REFERENCES

- [1] J. Wu and M. Choi, "Memristor lookup table (mlut)-based asynchronous nanowire crossbar architecture," in *Nanotechnology (IEEE-NANO), 2010 10th IEEE Conference on*, pp. 1100–1103, 2010.
- [2] L. Chua, "Memristor-the missing circuit element," *Circuit Theory, IEEE Transactions on*, vol. 18, no. 5, pp. 507–519, 2002.
- [3] D. Strukov, G. Snider, D. Stewart, and R. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.
- [4] S. Smith and J. Di, "Designing asynchronous circuits using NULL convention logic (NCL)," *Synthesis Lectures on Digital Circuits and Systems*, vol. 4, no. 1, pp. 1–96, 2009.
- [5] J. Huang, M. Tahoori, and F. Lombardi, "On the defect tolerance of nano-scale two-dimensional crossbars," in *Defect and Fault Tolerance in VLSI Systems, 2004. DFT 2004. Proceedings. 19th IEEE International Symposium on*, pp. 96–104, IEEE, 2004.
- [6] P. Vontobel, W. Robinett, P. Kuekes, D. Stewart, J. Straznicki, and R. Williams, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, p. 425204, 2009.
- [7] S. Venkateswaran, J. Lee, and M. Choi, "Novel functional testing technique for asynchronous nanowire crossbar system," in *Instrumentation and Measurement Technology Conference, 2009. I2MTC'09. IEEE*, pp. 1121–1125, IEEE, 2009.
- [8] E. Linn, R. Rosezin, C. Kugeler, and R. Waser, "Complementary resistive switches for passive nanocrossbar memories," *Nature Materials*, vol. 9, no. 5, pp. 403–406, 2010.