01 Aug 2008

A BIST Approach for Configurable Nanofabric Arrays

Mandar V. Joshi

Waleed K. Al-Assadi
Missouri University of Science and Technology, waleed@mst.edu

Recommended Citation

M. V. Joshi and W. K. Al-Assadi, "A BIST Approach for Configurable Nanofabric Arrays," *Proceedings of the 8th IEEE Conference on Nanotechnology, 2008. NANO'08*, Institute of Electrical and Electronics Engineers (IEEE), Aug 2008.
The definitive version is available at https://doi.org/10.1109/NANO.2008.210

A BIST Approach for Configurable Nanofabric Arrays

Mandar V. Joshi and Waleed K. Al-Assadi, *Senior Member, IEEE*
Department of Electrical and Computer Engineering
Missouri University of Science and Technology
Rolla, MO 65409

Abstract **—This work proposes a Built-in Self Test (BIST) approach to test crossbars for a defined set of faults. The BIST can classify the different programmable elements in the crossbars as non-defective or defective with a certain fault type. The logic synthesis can then configure the crossbar by avoiding these defective elements.**

Index Terms— **Crossbar architecture, BIST, Nanofabrics, Nanowires, Recovery, Redundancy, Defects**

I. INTRODUCTION

Bottom-up techniques that enable us to fabricate circuits of molecular dimensions, exploiting mechanical and electronic properties of CNTs and SiNWs have been suggested for digital systems [1] [2] [3]. A junction of two SiNWs or CNTs is termed a "crosspoint." A complete NanoPLA architecture uses the "stochastic addressing" developed by De Hon and takes advantage of programmable crosspoints [4]. All such architectures assume a certain assembly of NWs or CNTs, but crossbar (also referred to as "nanofabric") architectures are the most common of all. The key idea of configurability is that each NW can be uniquely addressed with a very high probability by introducing redundancy in terms of the number of wires. Redundancy ensures that even in the presence of a very high number of defects (nominally 13% to 20%); the desired digital circuits can be synthesized.

Our previous work pertaining to PLA architectures introduced the new concept of introducing fixed [5] or variable nanowire (NW) redundancy [6] to obtain higher yields than most of the other proposed logic blocks in a PLA.

In this work, however, we invoke a higher level of abstraction in which we divide our crossbar into a number of Programmable Blocks (PBs) equal in size to each other and equidistant, as shown in Figure 1. In the BIST procedure, the researchers configure the nanofabric array in a defined sequence of macros (logic circuits) and observe the outputs of neighboring logic blocks to find and analyze defects. The performance of such BIST procedures is governed by the types of configurations they need and the number of configurations. Each PB can be thought of as a PLA block, which has a rich interconnect. Each PB is either defective or defect free for a given configuration. If a PB is found to be defective, BIST techniques will tag it and the synthesizer will not be allowed to use it for the corresponding configuration.

The entire nanofabric array gets divided into sets of blocks that act either as the Blocks Under Test (BUTs) or Checker Blocks that test the validity of the outputs of the BUTs. A NanoBlock is a configurable block and a SwitchBlock is used as interconnect between different NanoBlocks.

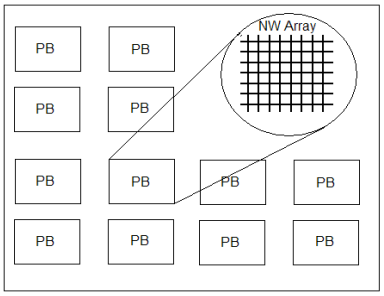


Fig.1 : Nanofabric PLA as an Array of Programmable Blocks

II. OVERVIEW OF PREVIOUS BIST TECHNIQUES PROPOSED FOR NANOFABRICS

M. Tehranipoor proposed a Built-in Self Test procedure for nanofabrics [7]. In this procedure, the nanofabric is split into NanoBlocks and SwitchBlocks that perform logical and routing operations, respectively. In the self-test, each NanoBlock is configured either as a *Pattern Generator (PG)* or a *Response Generator (RG)*. A *Test Group* is created using a set of PGs, RGs and SwitchBlock(s) between the two. Test Groups of the same kind form a *Test Architecture,* or *TA*, as shown in Figure 2. TAs are generated based upon the direction of fault in each NanoBlock and SwitchBlock. The NanoBlock configured as a PG tests itself and generates the test pattern for RG. An external device is needed to program the NanoBlocks and read the RGs' responses. $3n^2/4$ devices are configured. In the test configurations, stuck-at, stuck-open, forward biased and reverse biased diode and AND & OR bridging faults are targeted. A specific configuration of PGs and RGs is used for every type of fault. The main disadvantage of this scheme is that it requires an external tester. Moreover, faults in the SwitchBlocks are not considered.

978-1-4244-2104-6/08/$25.00 ©2008 IEEE.

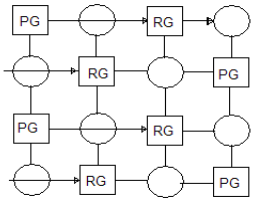
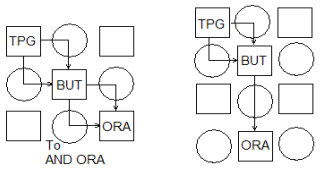


Fig. 2: Test Architecture for BIST

Z. Wang proposed a BIST approach that is similar in many ways to the approach discussed above [8]. In this BIST procedure, NanoBlocks can be configured as Test Pattern Generators (TPGs), Block Under Test (BUTs) or Output Response Analyzers (ORAs) as shown in Figure 3. These blocks, along with the corresponding SwitchBlocks, comprise a TG (Test Group) similar to one discussed in [7]. In a TG, the TPG generates the testing patterns for a BUT and ORAs examine the BUT output response. A TG and a set of Fault Detecting Configurations (FDCs) are used where different BUT faults can be tested. The metric defined for the quality of the test is called "recovery," which is defined as the ratio of non-defective blocks identified to the actual number of non-defective blocks. The separate test procedure for each type of TG needed to achieve full fault coverage results in three partial defect maps. The types of FDCs used in the test sequence are identical to test configurations used in [7]. A NanoBlock is defect free when it bypasses all three partial defect maps. It is assumed in the test sequence that ORAs can be read out using the mechanism that configured the fabric. The test results show that a 10x10 nanofabric with a 10% defect density yields a recovery of 76.9%.



(a) AND-OR ORA **(b) OR ORA**

Fig. 3: Different Test Groups Implementing FDCs

III. New Approach For Nanofabric BIST

A. Test Configuration

In the new approach, we model the nanofabric as a set of NanoBlocks similar to those in [7]. The types of blocks that can be targeted are single stuck-at and bridging faults. A test architecture consists of three blocks: two BUTs and one Comparator (denoted as "C"), as shown in Figure 4. Therefore, all the NanoBlocks take part in each test, and the test for a particular set of faults is completed in two configuration sequences. The BIST configures the blocks externally using the device's I/O interface.

Every block in the nanofabric has the ability to store the result of the comparison. It is assumed that a comparator always generates correct results storing a "0" for a successful comparison and a "1" for an unsuccessful comparison. In other words, the BIST remembers which comparison went wrong and reports it to the external tester. This helps generate an intermediate defect map called the "Raw Defect Map" and, in turn, the final defect map. A test is run for each type of fault to be targeted to create Raw Defect Maps for corresponding faults. Combining all the Raw Defect Maps gives the final defect map, which the logic synthesizer can use to synthesize a given logic by avoiding the defective blocks in the nanofabric.

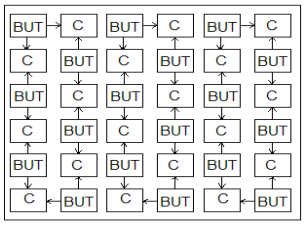


Fig. 4: Test Configuration for Proposed BIST

B. BIST Algorithm and Illustration

A block is declared fault-free only if it does not manifest any of the faults targeted. If the BIST can generate tests for "f" number of faults, 2f sets of test vectors are needed to test all blocks. These vectors create a Raw Defect Map for every fault targeted. All Raw Defect Maps are then read together to create the final defect map. The following algorithm describes the BIST sequence:

FOR *i=1 to* ***types of faults targeted ---BIST STEP 1 to f***
Generate test vectors for fault (i) in the first fault Detection loop;
Generate Raw_Defect_Map(i);
END FOR
Initialize final_defect_map=NULL;
FOR *i=1 to types of faults* ***------BIST STEP f+1***
Final_defect_map = final_defect_map + Raw_Defect_Map(i);
END FOR

During the initial "f" steps, the BUTs are configured for a certain logic that detects the targeted fault.

Figure 5 illustrates the two possible Fault Detection Loops which the whole nanofabric can be divided. To enable the conversion of every block in a nanofabric into a BUT, two such loops are needed. A Comparator compares the outputs of two neighboring BUTs and stores the results of comparison. Differences in the outputs of the two BUTs indicate the presence of a fault. At this time, the Comparator does not know which of the BUTs possesses the fault. Thus, the Comparator marks both BUTs as defect *suspects*. When the next Fault Detection Loop is applied, the actual faulty member is identified and is marked as "1." The Raw Defect

Map is updated accordingly.

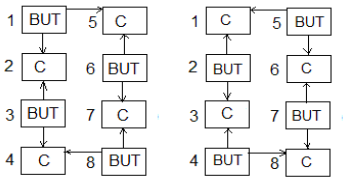


Fig. 5: An Illustration of Fault Detection Loops

Raw Defect Maps are obtained by sequencing through all the possible fault types. A final defect map is obtained by combining all of their 0s. The presence of a "1" essentially signals the inability of the corresponding block to be configured for a given logic function.

IV. Accuracy Enhancement Of Comparator

Comparator plays the most vital role in the proposed BIST, since its results are assumed to be correct. Because the comparator itself contains defective crosspoints, it is of essence that it produces results with a very high accuracy. This section explains how the accuracy of a comparison is improved. For a given defect rate:'d' the expected number of defects present in the BUT equals the product of BUT array size and d. A BUT cannot be tagged as defect suspect for the presence of a single fault, as all the BUTs are expected to contain a non-zero number of faults. Therefore, we tag the BUTs as defect-suspects when the total number of expected faults in them exceeds the above product. The presence of faults at the comparator would actually alter the number of unsuccessful comparisons. Therefore, the number of unsuccessful comparisons that gives the most accurate estimate for BUT defects needs to be found. Its analysis is as follows.

A comparison is successful when:

Non-defective element in a BUT is compared with non-defective element in the other BUT under comparison

OR

Defective element in a BUT is compared with defective element in the other BUT under comparison

Therefore,

P_1 = P(successful comparison)
$= (1-d).(1-d) + d.d$
$= 1- 2d + d^2 + d^2$
$= 1 – 2d + 2d^2$

And,

P_2=P(unsuccessful comparison)
$= 1- P_1$
$=1 - (1 – 2d + 2d^2)$
$= 2d (1-d)$

Where, d is the defect density in the nanofabric. It follows that the presence of d defects in an array gives rise to P2 number of unsuccessful comparisons. Therefore, if the number of unsuccessful comparisons is exceeded by the product of size of BUT array and P2, the BUTs are tagged as defective and otherwise they are tagged as non-defective by the comparator. This improves the comparator's probability of obtaining a successful comparison.

V. Results And Analysis

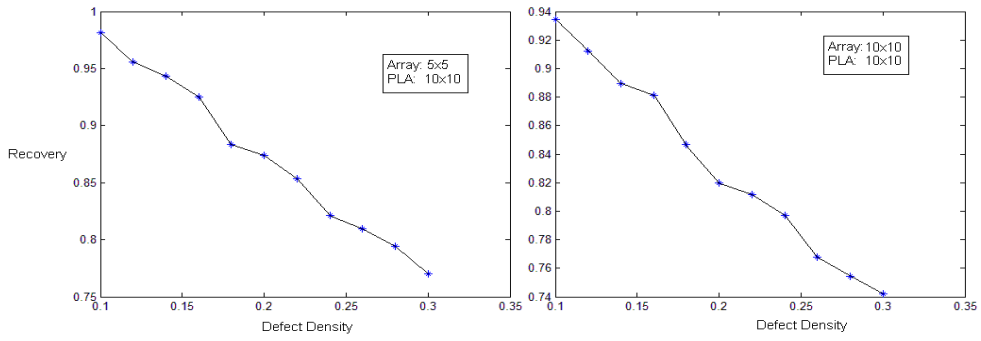
The coding and simulations were carried out using MATLAB for the proposed BIST on a machine with the following configuration:

AMD Turion 64 Processor 1.6 GHz, 1280 MB RAM

The fault universe consisted of two to five faults at a given time. The results were obtained in terms of recovery and computation time. The Defect Density or Defect Rate was varied from 10% to 30% for all faults in the fault universes to obtain the output parameters, namely recovery and computation time. All the results display the results based on the average of 1000 simulations.

$$\text{Recovery} = \frac{\text{Number of identified non - defective Logic Blocks}}{\text{Actual Number of non - defective Logic Blocks}} \quad (1)$$

$$\text{Defect Density} = \frac{\text{Number of defective Logic Blocks}}{\text{Total number of Logic Blocks}} \quad (2)$$



(a) (b)

Fig.6: Defect density vs. Recovery for (a) 5x5 array and (b) 10x 10 arrays

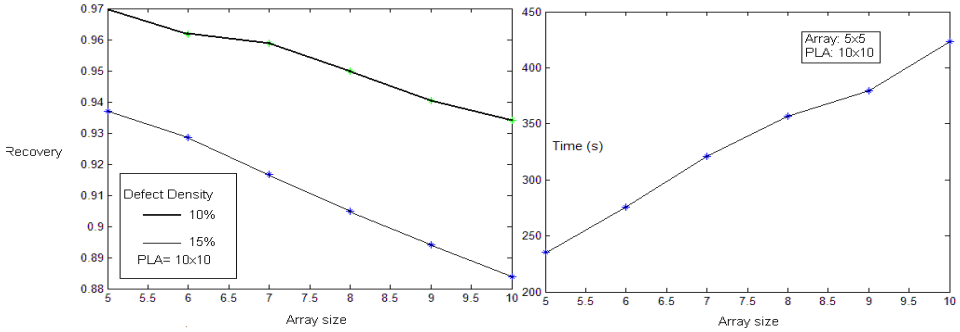


Fig.7: Array size vs. Recovery and Array size vs. Computation time

A. Effect of Defect Density

As the defect density increases, the ambiguity between the blocks marked as suspects and the actual defective blocks increases. Moreover, the BIST assumes that only one of the two blocks being compared is defective. This assumption ceases to hold true for higher defect densities, resulting in lower recovery rates at higher defect densities. This trend is independent of the array sizes and, thus, results in identical recovery values for all array sizes, as seen in Figure 6. It can be seen that recovery figures are much better than previous work in section II.

B. Effect of Array size

The simulations were conducted on square arrays to maintain symmetry. Since the computational complexity has a linear relationship to the size, the computation time grows continuously as size increases, as seen in Figure 7. The mathematical relationship of array size and computation will be established in the next section.

C. Computation Time

The following computations are involved in the completion of self-test:

1. Configuration time for each Fault Detection Loop $=T_{cfg}$
2. Comparison time consumed by each comparator $=T_{com}$
3. Calculation time consumed by external Tester when computing the final defect map $=T_{calc}$

It follows that the configuration time, $T_{cfg,}$ is taken by each fault type and is repeated twice because there are two Fault Detection Loops per fault. Similarly, the comparison takes place twice. Given the above considerations, the time complexity "T" is given by

$$T = O(\text{Array size}*(T_{cfg} \times f) + (T_{com} \times f) + T_{calc}) = O(\text{Array size}*(T_1+T_2+T_{calc})) \quad (3)$$

Where:

f = Number of faults in the fault universe

$T_1 = T_{cfg} \times f$, $T_2 = T_{com} \times f$

In the current scenario, the fault universe contains four faults. As the array size increases, the computation time also increases, as seen in Figure 7.

VI. Conclusion

Only two configurations are needed to cover all the NanoBlocks to test a particular block, whereas the other techniques require a set of configurations depending on size and the type of fault targeted. In our technique, the number of blocks tested at any time is a constant and equals half the total blocks. This technique is much more area efficient because two of the three NanoBlocks configured in our technique are tested at a time and there is no need to dedicate two blocks exclusively to pattern generation and response analysis. It is flexible in terms of fault analysis. The fault set can be previously defined, and the configurations can be developed based on each fault. The entire nanofabric is tested in just two configuration sequences, which reduces the overall time required to test the complete fabric for a given fault. Another advantage of the new BIST approach is its constant recovery rate with respect to array size. Scaling of arrays without loss of recovery becomes possible.

REFERENCES

[1] M. Mishra and S. Goldstein, "Defect Tolerance at the End of the Roadmap," International Test Conference (ITC) 2003, pp. 1201-1210.

[2] Y. Cui, Z. Zhong, D. Wan, W. Wang and C. Lieber, "High Performance Silicon nanowire Field Effect Transistors," Nano Letters 2003, vol. 3, no. 2, pp. 149-215.

[3] S. Cha, J. Jang, Y. Choi, G. Amaratunga, D. Kang, D. Hasko, J. Jung and N. Kim, "Fabrication of a Nanoelectromechanical Switch Using a Suspended Carbon Nanotube," Applied Physics Letters 2005, vol. 86, no. 8, id. 083105.

[4] A. DeHon and M. J. Wilson, "nanowire-Based Sublithographic Programmable Logic Arrays," International Symposium on Field Programmable Gate Arrays (FPGA'04), pp. 123-132.

[5] M. V. Joshi and W. K. Al-Assadi, "Nanofabric PLA Architecture to Minimize Configuration Time Complexity," 2007 IEEE Region 5 Technical Conference, pp. 32-36, April 2007

[6] M. V. Joshi and W. K. Al-Assadi, "Nanofabric PLA architecture with Flexible Nanowire-Redundancy," NSTI Nanotech conference, pp. 116-169, May 2007.

[7] M. Tehranipoor, "Defect Tolerance for Molecular Electronics-Based NanoFabrics Using Built-In Self-Test Procedure," Proceedings of the 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), 2005.

[8] Z. Wang and K. Chakraborty, "Built-in Self-Test of Molecular Electronics-Based nanofabrics," Proceedings of European Test Symposium (ETS), 2005.

[9] J. G. Brown and R. D. Blanton, "CAEN-BIST: Testing the NanoFabric," Proceedings of International Test Symposium (ITC), pp. 462-471, 2004.