Built-in Self-Test of Molecular Electronics-Based Nanofabrics

Zhanglei Wang and Krishnendu Chakrabarty
Department of Electrical and Computer Engineering
Duke University, Durham, NC 27708, USA
Email: {zw8, krish}@ee.duke.edu

Abstract— **We propose a built-in self-test (BIST) procedure for nanofabrics based on chemically-assembled electronic nanotechnology. We also present a recovery procedure through which we can identify defect-free nanoblocks and switchblocks in the nanofabric under test. The proposed BIST and recovery procedures are based on the reconfiguration of the nanofabric to achieve complete fault coverage of different types of faults. We show that a large fraction of defect-free blocks can be recovered using a small number of BIST configurations. The proposed BIST procedure is well suited for regular and dense architectures that have high defect densities.**

I. Introduction

Although complementary metal-oxide semiconductor (CMOS) chips are projected to continue their dominance for another 10-15 years [1], CMOS technology today faces a number of challenges. Quantum effects will soon make it nearly impossible to further scale devices. Moreover, the high cost associated with chip masks and next-generation fabrication plants poses a formidable economic barrier to commercial nanometer-scale lithography.

Chemically-assembled electronic nanotechnology (CAEN) is a nanoelectronic technology that is under intense investigation as a possible alternative to CMOS integrated circuits [2], [3]. It has the potential to achieve high density, and it can be fabricated using low-cost chemical synthesis processes. CAEN uses self-assembly and self-alignment to construct electronic circuits out of nanometer-scale devices. CAEN-based systems, referred to as the nanofabric, can achieve a density of more than 10^8 gate-equivalents per cm^2 by using interconnected 2D-arrays of nano-scale wires that can be electronically configured as logic networks, memory units, and signal-routing cells [4]. The 2D arrays, referred to as nanoblocks, are the fundamental units of the nanofabric.

While CAEN-based systems offer the advantage of low manufacturing cost and high density, they are inherently unreliable. The low reliability is a direct consequence of the stochastic nature of self-assembly. It has been predicted that the defect density of CAEN-based systems can be as high as ten percent [5]; therefore it is not economically feasible to discard a nanofabric once a fault is detected. Defect tolerance is needed to make such nanofabrics commercially viable.

Defect tolerance refers to the ability to detect and locate fault sites on a chip, and then avoid the faults through reconfiguration methods. The first step in defect tolerance is to map designs to usable sets of resources; this step leads to increased yield and reduced manufacturing cost. New methods must therefore be devised to diagnose defective sections of the nanofabric. Unlike many testing and diagnosis techniques intended for CMOS chips, testing methods for nanofabrics cannot simply assume a small number of defects or use conventional fault models that only target a few fault sites.

A nanofabric system is similar to a field-programmable gate-array (FPGA) because of its regular 2D-array architecture and reconfigurability. In FPGA-BIST presented in [6], [7], programmable logic blocks (PLBs) are configured as test pattern generators (TPGs), blocks under test (BUTs) and output response analyzers (ORAs) for built-in self-test (BIST). A TPG applies exhaustive test patterns to a BUT and output responses are fed to an ORA. Multiple configurations are needed to ensure that every PLB is tested as a BUT. However, the problem of nanofabric testing is different from FPGA testing due to two main reasons: (1) nanofabric systems are expected to have much higher defect densities and a larger number of resources, and (2) as detailed in Section II, the fundamental units (nanoblocks) in the nanofabric are very simple compared to PLBs in FPGA. It is difficult to create complex comparators or LFSR signature generators using primitive units that are likely to be defective. Therefore, new methods must be devised to address these problems.

In this work, we propose a BIST and recovery procedure for the nanofabric that uses simple single-nanoblock TPGs, BUTs and ORAs. This fine-grained test method allows us to handle high defect densities. We exploit the fact that even for high defect densities, a small number of neighboring simple nanoblocks can be expected to be defect-free. Instead of specifying a set of complex test patterns, our procedures rely on a set of configurations to test the nanofabric. Complex test patterns cannot be generated by simple TPGs in the nanofabric, and due to limited routing resources, it is difficult to feed test patterns using external testers. Since nanoblocks are tested in parallel, the testing time using the proposed procedure is independent of the size of the nanofabric.

The remainder of this paper is organized as follows. We discuss the nanofabric architecture in Section II. Some related prior work on nanofabric testing is reviewed in Section III. The proposed BIST and recovery procedures are presented in Section IV. The configurations for fault detection are presented in detail in Section V. We present our simulation results in Section VI and conclude the paper in Section VII.

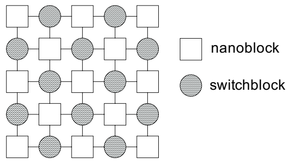


Fig. 1. The nanofabric architecture.

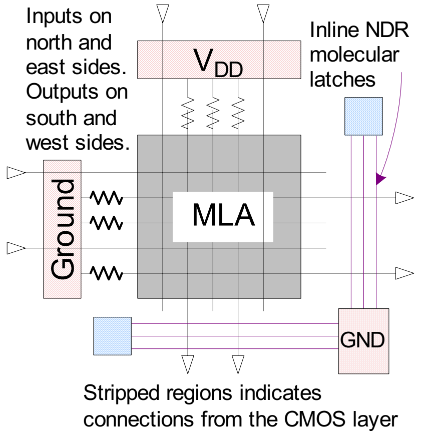

Inputs on north and east sides. Outputs on south and west sides.
V_{DD}
Inline NDR molecular latches
Ground
MLA
GND
Stripped regions indicates connections from the CMOS layer

Fig. 2. Schematic of a nanoblock [2].

II. NanoFabric

A feasible fabrication process for CAEN systems is bottom-up manufacturing, where basic components such as wires and switches are first obtained through chemical self-assembly, and then aligned and grouped into regular structured arrays through self-assembly to form complete systems [2]. Two planes of aligned wires are combined to form a two-dimensional grid with configurable molecular switches at the cross-points. The resulting grid is of the order of a few microns. A post-fabrication configuration step is used to create useful circuits out of these grids [2].

The nanofabric architecture has been proposed for a CAEN-based system in [2], [3], [5]. The self-assembly process does not allow precise end-to-end connections between nanoscale wires. The nanofabric architecture requires that all connections be made only at the cross-points between two orthogonal wires. Molecular latches based on resonant tunneling diodes, referred to as RTDs, are also incorporated in this architecture for saving states and for signal restoration [8].

Similar to FPGAs, the nanofabric is a regular 2D-mesh of interconnected fundamental units called nanoblocks, as shown in Fig. 1. A nanoblock can be programmed after fabrication to implement logic functions. The switchblock is the area where the input and output wires of nanoblocks overlap. It can be configured to route signals between nanoblocks [2].

A. Nanoblock

As shown in Fig. 2, a nanoblock consists of three parts: (1) the molecular logic array (MLA), which implements the functionality of the block, (2) the molecular latches, used for signal restoration and signal latching, and (3) the I/O area, used to connect the nanoblock to its neighbors [2].

The MLA is composed of two orthogonal sets of wires. At each intersection of two wires lies a configurable molecular

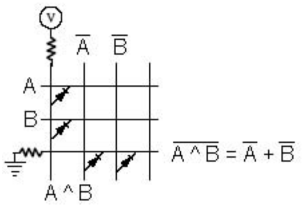


Fig. 3. An AND gate.

switch. The switches, when configured to be "on", act as diodes [2]. The direction of the current flowing through a "on" molecular switch is determined during fabrication and is non-reconfigurable. Fig. 3 shows the implementation of an AND gate. If either A or B is at logic "0", the coresponding diode is forward-biased and turned on. The resistors are manufactured appropriately, i.e., resistors attached to V_{DD} have smaller impedances than those attached to Gnd, such that the output vertical wire is pulled down to logic "0" [2]. Note that the resistance of nanowires and molecular switches are very low. Fig. 3 also shows how an OR gate can be implemented. This diode-resistor logic is unable to perform the inversion operation, therefore complemented inputs are required and the complement of each logic function also needs to be implemented.

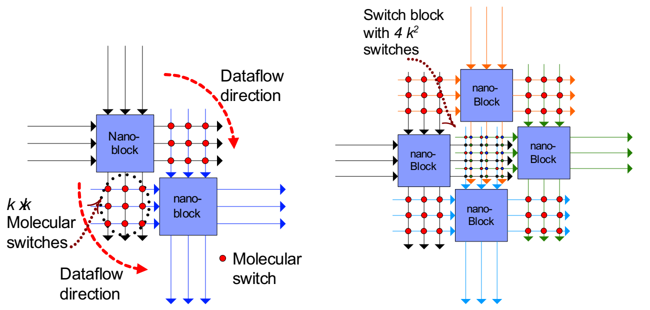
If the MLA portion of a nanoblock has k horizontal wires and k vertical wires, then the size of the nanoblock is referred to as $k \times k$. We only consider nanoblocks that have equal numbers of horizontal wires and vertical wires.

The above nanoblock design is dictated by fabrication constraints. Each side of the block can have either inputs or outputs, but not both. All nanoscale wire-to-wire connections are made between two orthogonal wires; precise end-to-end alignment is not possible. The outputs of the blocks are either facing south and east (SE) or north and west (NW), as shown in Fig. 2 [2]. Without loss of generality, we assume that a nanofabric consists of only SE nanoblocks whose outputs are facing south and east.

The MLA implements boolean functions using diode-resistor logic. The drawback of this logic style is that a signal is degraded whenever it passes a molecular switch. The molecular latch, constructed entirely from molecular-scale devices, is used to perform signal restoration using power from the clock to provide gain. The molecular latch also provides the properties of I/O isolation and noise immunity [8].

B. Switchblock

A switchblock is similar to the MLA portion of a nanoblock, with the difference that it does not have inline NDR latches, I/O ports and connections to V_{DD} and Gnd. As shown in Fig. 4, a switchblock is formed by 4 nanoblocks; crossing horizontal wires and vertical wires from the surrounding nanoblocks are connected by configurable molecular switches. If the size of the nanoblocks is $k \times k$, then there are $2k$ vertical wires and $2k$ horizontal wires inside a switchblock, and $4k^2$ cross-points can be formed. For SE nanoblocks, a switchblock is capable of providing four directions of data flow: west to south (WS), west to east (WE), north to east (NE), and north to south



(a) Half of a switchblock (b) A switchblock with four surrounding nanoblocks

Fig. 4. Nanoblock connectivity. [2]

(NS). WS and NE data flows can co-exist with each other in a same switchblock. On the other hand, WE and NS data flows cannot co-exist with any other data flows because vertical (horizontal) wires cannot be directly connected to vertical (horizontal) wires. Therefore if two vertical (horizontal) wires are to be connected, a horizontal (vertical) wire in the same switchblock must be used, which means that this horizontal (vertical) wire cannot be used by its own nanoblock in order to avoid conflicts.

C. Defect Tolerance

The nanofabric has a much higher defect density than standard CMOS chips due to the imprecise and nondeterministic manufacturing process. Wires will rarely all be equidistant from each other. Wires that should be parallel may be askew or they may intersect. The connections between wires may be open or wires may be shorted [5].

The nanofabric has a built-in capability for defect tolerance due to its reconfigurability. An effective testing procedure should lead to a *defect map*, which provides the locations of the defective nanoblocks and switchblocks. The defect map can then be used by software tools to avoid faulty resources during system reconfiguration.

One metric of defect map quality is *recovery*, defined as the percentage of defect-free nanoblocks and switchblocks that are correctly diagnosed [3]. This metric indicates the diagnostic accuracy of a testing procedure. It should also be ensured that no faulty blocks are diagnosed as defect-free. An ideal recovery of 100% implies that every defect-free block in the nanofabric is correctly dignosed. However, this metric is useful only for simulation; in practice, it is difficult to assertain the actual number of defect-free blocks after fabrication. An effective testing procedure should correctly identify a large fraction of these defect-free blocks, thereby minimizing wastage.

III. Related Prior Work

Testing of nanofabrics was first addressed in [3], [4]. The none-some-many algorithm presented in [3] creates LFSR-based signature generators from a random selection of nanoblocks. This approach however makes the unrealistic assumption that unlimited interconnect resources are availabe to create signature generators from randomly-selected nanoblocks. Moreover, since this approach uses a large number of nanoblocks to build LFSR-based signature generators, it is coarse-grained and it can only provide limited recovery.

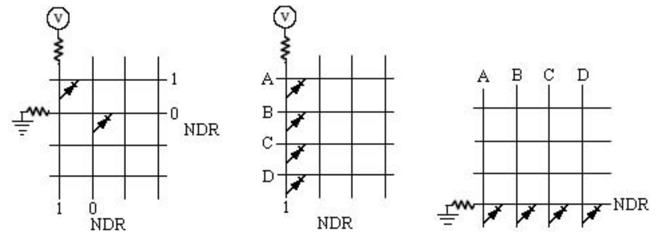
The CAEN-BIST approach presented in [4] is a fine-grained test method. It configures a nanoblock as a tester to test its neighboring nanoblocks. Test patterns are fed to both the tester and the nanoblock under test (BUT) from an external source. A defect-free BUT generates output patterns that are identical to the input patterns. The tester compares the input test patterns and the output patterns from the BUT to see if the BUT is defective. The average recovery is reported to be almost 100% for defect densities up to 20%. However, this approach makes two strong assumptions: (1) a k-bit comparator can be implemented using a nanoblock, and (2) defect-free data paths from external test circuits to testers and BUTs can be dynamically identified during the test procedure. Since the nanoblocks can only implement simple logic functions, it is not clear how they can be configured to implement k-bit comparators. Moveover, [4] does not consider the limitations in dataflow imposed by the biasing of the molecular switches. In addition, because the input patterns are provided by external circuits instead of internal nanoblocks, CAEN-BIST can only be performed in a wave-like manner in which a set of nanoblocks in the same diagonal tests another set of nanoblocks until the entire nanofabric has been tested. Therefore, the complexity of CAEN-BIST depends on the size of the nanofabric under test.

IV. NanoFabric BIST Approach

We now present a BIST approach for nanofabric testing that exploits the reconfigurability of nanoblocks and switchblocks. The nanoblocks of the nanofabric are configured as either TPGs, BUTs, or ORAs. Three nanoblocks (i.e., one TPG, BUT and ORA) along with the switchblocks between them form a test group (TG) in which the TPG applies input signals to the BUT, and the ORA examines the output responses from the BUT to determine if there is a defect in the group. The whole fabric is partitioned into a set of TGs such that all BUTs inside them can be tested in parallel.

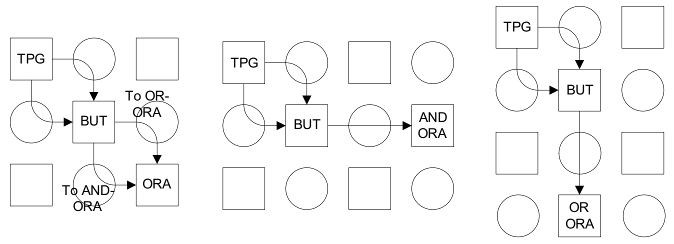
Our strategy relies on a set of fault detection configurations (FDCs) where different faults of the BUT can be tested. The proposed configurations can provide 100% fault coverage for any stuck-at, stuck-open, bridging, and connection faults in the nanoblocks. The details of these configurations are discussed in the next section. A BUT is deemed to be defect-free only if it operates correctly in all configurations.

The test procedure consists of a sequence of test phases, where each phase consists of the following steps: 1) partition the nanofabric into TGs, 2) configure the nanoblocks into one of the FDCs, 3) apply the test and read outputs of the ORAs, and 4) repeat from step 2) until all FDCs that are compatible with the current set of TGs are applied. Multiple test phases are needed to test each nanoblock in the fabric. When the test procedure is completed, a defect map can be constructed. The test procedure is discussed further in Section IV-B.



(a) A TPG. (b) An AND ORA. (c) An OR ORA.

Fig. 5. Nanoblocks configured as TPG and ORA.



(a) TG_SE with either AND or OR ORA (b) TG_E with AND ORA (c) TG_S with OR ORA

Fig. 6. Illustrating the need for different TGs.

A. BIST Architecture

Fig. 5(a) shows the structure of a TPG. In the proposed fault detection configurations, a BUT only needs all-"1" and all-"0" input signals, thus the TPG can be very simple. A TPG provides "1" and "0" on both output sides. Every output port is connected to a molecular latch, which provides pull-up and pull-down paths to the down-streaming block.

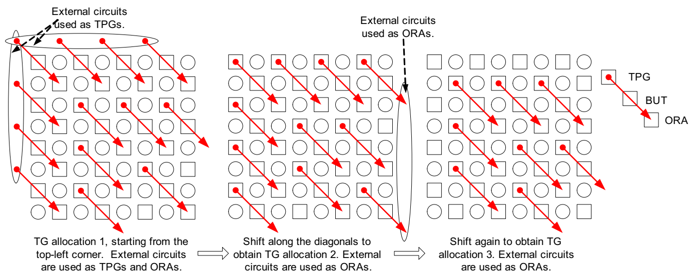
Fig. 5(b) and Fig. 5(c) show the structures of the ORAs. We carefully designed the BUTs such that the outputs of a defect-free BUT are identical, either all-"1" or all-"0". Therefore an ORA is simply a k-input AND gate or a k-input OR gate. Due to the restrictions of the CAEN architecture, an AND gate can only accept inputs on its east side for SE blocks (or west side for NW blocks), and an OR gate can only accept inputs on the north side for SE blocks. This restriction implies that three different types of TGs are needed such that all FDCs can be applied. As shown in Fig. 6(a), in TG_SE the ORA is to the SE of the BUT, and it can be either an AND or an OR gate, provided that the outputs come from the appropriate side of the BUT. Similarly, the ORA in TG_E (Fig. 6(b)) is an AND gate and the ORA in TG_S (Fig. 6(c)) is an OR gate.

We assume that the results of the ORAs can be read out using an access mechanism that is used for configuring the fabric. A similar assumption is made in recent work on nanofabric testing [3], [4].

Due to the connectivity restrictions shown in Fig. 4, none of the three TGs can be used for all FDCs. To achieve full fault coverage, we need a separate test procedure for each type of TG, which results in three partial defect maps. An overall defect map can then be derived from these partial defect maps. A nanoblock is considered to be defect-free only if it is defect-free in all the three partial defect maps; a switchblock, however, is deemed to be defect-free if it is defect-free in any of the three partial defect maps. An

```
BIST Procedure: input: type of test group (TG)
 1) while not all nanoblocks are tested
 2)   partition the fabric into TGs;
 3)   while not all compatible FDCs are applied
 4)     apply one FDC;
 5)     run the test;
 6)     read out ORA responses;
 7)   end while (FDC)
 8) end while (nanoblocks)
 9) generate the defect map.
```

Fig. 7. BIST procedure for any given TG



External circuits used as TPGs.
External circuits used as ORAs.
TPG
BUT
ORA
TG allocation 1, starting from the top-left corner. External circuits are used as TPGs and ORAs.
Shift along the diagonals to obtain TG allocation 2. External circuits are used as ORAs.
Shift again to obtain TG allocation 3. External circuits are used as ORAs.

Fig. 8. TG allocations for TG_SE

underlying assumption here is that the FDCs used in a test procedure can provide 100% fault coverage for switchblocks. Thus a defect switchblock fails to operate correctly in all the three test procedures and is marked as defective in all the three partial defect maps. Therefore, if a switchblock is deemed as defect-free in a partial defect map, it must be defect-free.

B. The BIST Procedure

The BIST procedure is shown in Fig. 7. In Step 2, TGs are allocated so that the BUTs can be tested in parallel. A *TG allocation* is a high level configuration for the entire nanofabric defining how TGs are created. We assume that nanoblocks along the edges can be accessed by external circuits, which in turn can serve as TPGs or ORAs for those blocks on the edges. The number of TG allocations is independent of the size of the fabric. For TG_SE, three allocations are needed, as shown in Fig. 8. A TG is represented by an arrow that starts from the TPG and ends at the ORA. Arrows whose start or end is outside the fabric represent those TGs that use external circuits as TPGs or ORAs. We simply shift all the arrows along the diagonals to obtain the three allocations. Similarly, for TG_S and TG_E, four allocations are needed.

In Step 9, initially all nanoblocks and switchblocks are deemed to be defective. If a BUT produces the correct outputs for all the applied FDCs, it is deemed to be defect-free. Since the TPG and the ORA are very simple, we assume it is unlikely that a faulty BUT will operate correctly in a TG whose TPG or ORA is also faulty. The switchblocks that connect the BUT to its ORA are also marked as defect free.

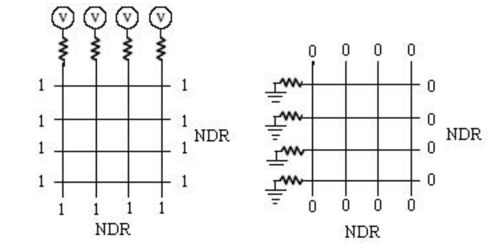
The overall BIST algorithm is shown in Fig. 9. In Step 4, a nanoblock is marked as defect free if and only if it is defect-free in all the three partial defect maps. A switchblock, however, is regarded as defect-free if it is defect-free in any of the defect maps.

V. Fault Detection Configurations

In each test phase, a set of configurations are needed to configure the BUTs into different circuits to provide 100%

```
Overall BIST Algorithm:
  1) Run the BIST procedure using TG_SE;
  2) Run the BIST procedure using TG_E;
  3) Run the BIST procedure using TG_S;
  4) Generate an overall defect map.
```

Fig. 9. Overall BIST algorithm.



(a) Category 1(a): stuck-at-0 and stuck-open faults. (b) Category 1(b): stuck-at-1 faults.

Fig. 10. BUTs for Category 1.

fault coverage for stuck-at, stuck-open, bridging, and connection faults in the nanoblocks. These configurations, called fault detection configurations (FDCs), are classified into five categories.

In these configurations, we make the following assumptions based on published papers on nanofabrics: (1) Each output port has an associated inline NDR latch, whose state can be reset to "0" [8], and (2) Resistors attached to Gnd have much smaller resistance that those attached to Gnd [2].

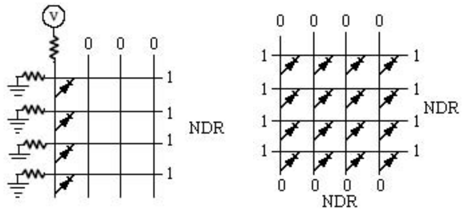
A. Category 1: FDCs for stuck-at and stuck-open faults

In Category 1(a) illustrated in Fig. 10(a), all inputs are connected to V$_{DD}$ or "1", therefore all outputs should be high for a defect-free BUT. The inline NDR molecular latches are initially set to "0" by adjusting V_{ref} [2]. For a defect-free BUT, since each output port is connected to an inline NDR, the latches are driven to "1". However, a line with a stuck-at-0 and/or stuck-open fault will fail to drive its associated NDR to the "1" state. By using a k-input AND gate as an ORA, any stuck-at-0 and stuck-open fault on a line can be detected. Category 1(a) includes two configurations, one corresponding to TG_SE and the other corresponding to TG_E. The TG, BUT and ORA for TG_SE are shown in Fig. 6.

Similarly, in Category 1(b) shown in Fig. 10(b), all inputs are connected to Gnd or "0". With a k-input OR gate, any stuck-at-1 fault can be detected. Category 1(b) also has two configurations, out of which one configuration is compatible with TG_SE and the other is compatible with TG_S.

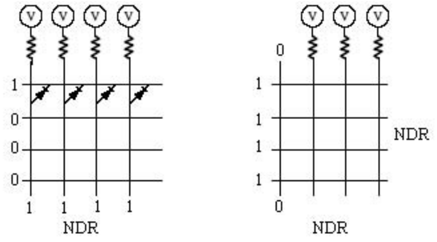
B. Category 2: FDCs for connections of forward-biased diodes and AND-bridging faults

Category 2 includes k configurations. In each configuration, one of the vertical wires is connected to V$_{DD}$ and the other vertical wires are connected to "0". All horizontal wires are connected to Gnd. The k junctions along the vertical wire that is connected to V$_{DD}$ are configured to "on" and are forward-biased. The outputs are "1" for a defect-free BUT. If a forward-biased diode is defective, the horizontal wire attached to it becomes "0". With a k-input AND gate ORA, we can test all the $k \times k$ cross-points.



(a) (b)

Fig. 11. (a) Category 2: forward-biased diodes faults and AND-bridging faults (v/v, v/h). (b) Category 3: reverse-biased diodes faults.



(a) (b)

Fig. 12. (a) Category 4: AND-bridging fault among horizontal wires (h/h). (b) Category 5: OR-bridging fault.

The above category can also detect AND-bridging faults among the vertical wires (v/v). For a given configuration, if the wire connected to V$_{DD}$ is shorted to any of the other vertical wires, the output will become "0" and can be detected by the ORA. All the k configurations together can test any AND-bridging faults between vertical wires.

AND-bridging faults involving a vertical wire and a horizontal wire (v/h) can also be detected using the FDC in Category 2. If any of the vertical wires connected to "0" is shorted to any of the horizontal wires, the output becomes "0" and the error can be detected. The k configurations for this FDC can together detect any AND-bridging faults of this type.

Configurations in Category 2 can only be used with TG_E because an AND ORA is used and the output responses come from the W-side of the BUT. Hence TG_S and TG_SE cannot be used for this category.

C. Category 3: reverse-biased diodes

In Category 3, all molecular switches are configured to be closed. Horizontal wires are connected to "1" and vertical wires to "0". Therefore all the cross-points are reverse-biased and the output should be "1". If any of the reverse-biased diodes is defective and has a small enough resistance to bridge the wires forming this cross-point, the output on the east side will be pulled down to "0" (AND-bridging), or the output on the south side will be pulled up to "1" (OR-bridging). By using an AND/OR gate we can detect defective reverse-biased diodes.

A total of two configurations are needed for this category, one using TG_E to observe W-side outputs and the other using TG_S to observe S-side outputs.

D. Category 4: AND-bridging fault among horizontal wires (h/h)

Category 4 is the same as Category 3, with the difference that the vertical wires are connected to V$_{DD}$. The correct output on a wire is "1". For a given configuration, if the horizontal wire connected to "1" is shorted to any of the other horizontal wires, assuming AND-bridging fault, the output will become "0" because there are pull-down resistors attached to V_{DD}. A k-input ORA can detect this fault. This category requires the use of TG_SE. The k configurations can together detect any AND-bridging faults in nanoblocks.

E. Category 5: OR-bridging fault

Category 5 is used to detect OR-bridging faults between any of the nano wires. Only one wire is connected to "0" and all other wires are connected to V_{DD} or "1". If the "0"-wire is bridged to any other wire, it will be pulled up to "1". We only need to monitor the voltage level of this single wire using a 1-input OR gate. Clearly, $2k$ configurations are needed to test all the possible OR-bridging faults. Note that TG_SE and TG_S are compatible with this category.

In summary, a total of $4k + 6$ configurations are needed to test for the stuck-at, stuck-open, bridging and defective cross-points. Table I lists the faults indicated with x entries that each category can detect.

Fault model	1a	1b	2	3	4	5
stuck-at-0	x					
stuck-at-1		x				
open-line	x					
AND-bridging (v/v, v/h)			x			
AND-bridging (h/h)					x	
OR-bridging						x
cross-points (forward)			x			
cross-points (reverse)				x		
# of configurations needed	2	2	k	2	k	$2k$
Type of TG	SE, E	SE, S	E	E, S	SE	SE, S

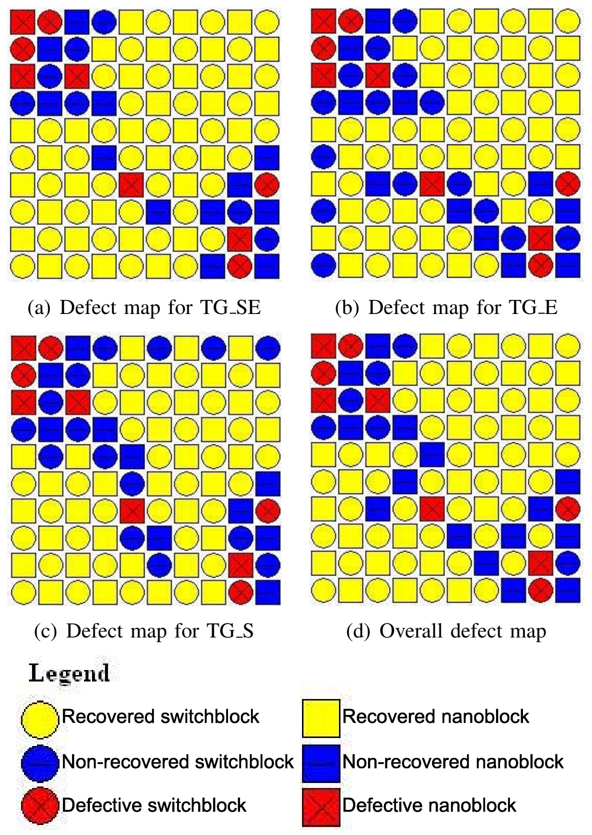
TABLE I
SUMMARY OF PROPOSED CONFIGURATIONS

VI. Simulation Results

Finally, we present simulation results for the proposed BIST approach. Fig. 13 shows the defect maps obtained from simulation on a 10×10 nanofabric with 10% defect density. The overall recovery, defined as the percentage of the number of defect-free nanoblocks and switchblocks that are diagnosed as defect-free, is 76.9%. It can be seen that a defective switchblock or nanoblock renders many of its neighboring blocks unusable because they can't be recovered.

VII. Conclusion

We have presented a new built-in self-test strategy for the CAEN-based nanofabric. The proposed BIST procedure configures the nanoblocks into test pattern generators (TPGs), blocks under test (BUTs) and output response analyzers (ORAs), which in turn form test groups where the BUTs are tested. A test group only contains three nanoblocks and the switchblocks between them, therefore the algorithm is able to handle nanofabric systems with a large number of devices and high defect densities. A set of configurations have been presented to detect various faults inside a nanoblock; these configurations provide 100% fault coverage for stuck-at, stuck-open, bridging, and connection faults. The complexity of this algorithm and its recovery capability are independent of the size of the nanofabric. Ongoing work is focused on the design of additional configurations to gurantee coverage for switch-block faults. We are also investigating multi-step adaptive diagnose methods to achieve higher recovery, especially for even higher defect densities.



(a) Defect map for TG_SE (b) Defect map for TG_E (c) Defect map for TG_S (d) Overall defect map

Fig. 13. Defect maps for a 10×10 fabric with 10% defect density.

References

[1] "Semiconductor industries association roadmap." [Online]. Available: http://public.itrs.net

[2] S. C. Goldstein and M. Budiu, "NanoFabrics: Spatial computing using molecular electronics," in *Proceedings of International Symposium on Computer Architecture*, 2001, pp. 178–189.

[3] M. Mishra and S. Goldstein, "Defect tolerance at the end of the roadmap," in *Proc. International Test Conference*, 2003, pp. 1201–1210.

[4] J. G. Brown and R. D. S. Blanton, "CAEN-BIST: Testing the nanofabric," in *Proc. International Test Conference*, 2004, pp. 462–471.

[5] M. R. Stan, *et al.*, "Molecular electronics: From devices and interconnect to circuits and architecture," *Proc. IEEE*, vol. 91, pp. 1940–1957, Nov. 2003.

[6] C. Stroud, S. Konala, P. Chen, and M. Abramovici, "Built-in self-test of logic blocks in FPGAs (finally, a free lunch: BIST without overhead!)," in *Proc. IEEE VLSI Test Symposium*, 1996, pp. 387–392.

[7] M. Abramovici, E. Lee, and C. Stroud, "BIST-based diagnostics for FPGA logic blocks," in *Proc. International Test Conference*, 1997, pp. 539–547.

[8] S. C. Goldstein and D. Rosewater, "What makes a good molecular-scale computer device?" School of Computer Science, Carneige Mellon University, Tech. Rep. CMU-CS-02-181, Sept. 2002.

[9] M. B. Tahoori, E. J. McCluskey, M. Renovell, and P. Faure, "A multi-configuration strategy for an application dependent testing of fpgas," in *Proc. VLSI Test Symposium*, 2004, pp. 154–159.