
Chapter 2: Built-in Self-Test and Defect Tolerance in Molecular Electronics-Based Nanofabrics

Z. Wang and K. Chakrabarty

1 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. Deep sub-micron (DSM) technologies suffer from high leakage, and it is projected that stand-by power and active power for CMOS chips will soon become comparable [2]. 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 [1, 3–6]. 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 [6]. The 2D arrays, referred to as nanoblocks, are the fundamental units of the nanofabric. A prototype nanowire-based crossbar design with 6.8 Gbits cm^2 density is reported in [7], and a manufacturer is promising memories using carbon nanotubes [8].

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 easily exceed 10% [6]; 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.

Modern memory chips use built-in redundancy, in which spare rows and columns are used to replace defective rows and columns, respectively, to achieve defect tolerance [9]. However, it is unlikely that any row or column of nanoblocks will be defect-free. Moreover, the nanofabric is intended to implement complex logic functions, hence simple row/column replacement algorithms will not be efficient. Majority voting is another fault tolerance technique using redundant hardware resources to perform correct computation in the presence of defects. However, due to the high anticipated defect density of CAEN-based systems, the amount of additional physical resources required will not be affordable.

A nanofabric system is similar to a field-programmable gate-array (FPGA) because of its regular 2D-array architecture and reconfigurability. A number of testing methods have been proposed for various FPGA architectures, e.g., [10–15]. In FPGA-BIST presented in [10, 11], 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. Teramac [16, 17] is an FPGA-based custom computer system that is capable of running user designs even if up to 75% of its FPGAs contain defects. A testing phase is used after fabrication to identify defects in FPGAs. In the Teramac system, circuit components are configured as LFSRs that generate long psuedo-random bit streams and communicate them to primary outputs. If the output bit stream is correct, all components are assumed to be defect-free; if incorrect, these components are configured to create new LFSR signature generators. The resources at the intersection of defective LFSR configurations are marked as defective. 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 Sect. 2, 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. However, the method used to read out test results also affects the overall testing time. In the absence of manufactured nanofabric chips, it is not clear how much time will be needed for ORA access; the access time may depend on the size of the nanofabric. We consider the detectability of multiple faults in blocks within the nanofabric. We also present simple bounds on the recovery that can be achieved for a given defect density.

The configurations for fault detection are presented in detail in Sects. 5 and 6. Section 7 discusses the detection of multiple faults. Section 8 presents an adaptive recovery procedure to further improve the performance and Sect. 9 investigates the effectiveness of the whole procedure. We present our simulation results in Sect. 10 and conclude the chapter in Sect. 11.

2 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 [3]. 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 [3].

The nanofabric architecture has been proposed for a CAEN-based system in [1, 3, 6]. 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 [18].

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 [3].

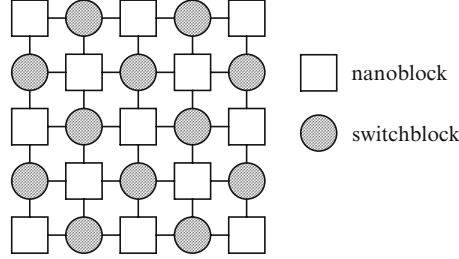


Fig. 1. The nanofabric architecture

2.1 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 [3].

The MLA is composed of two orthogonal sets of wires. At each intersection of two wires lies a configurable molecular switch. The switches, when configured to be “on”, act as diodes [3]. The direction of the current flowing through a “on” molecular switch is determined during fabrication and is non-reconfigurable. Figure 3 shows the implementation of an AND gate. If either A or B is at logic “0”, the corresponding diode is forward-biased and turned on. The resistors are manufactured appropriately, i.e., resistors attached to Gnd have smaller impedances than those attached to V_{DD} , such that the output vertical wire is pulled down to logic “0” [3]. Note that the resistance of nanowires and molecular switches are very low. Figure 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 [3]. 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 [18].

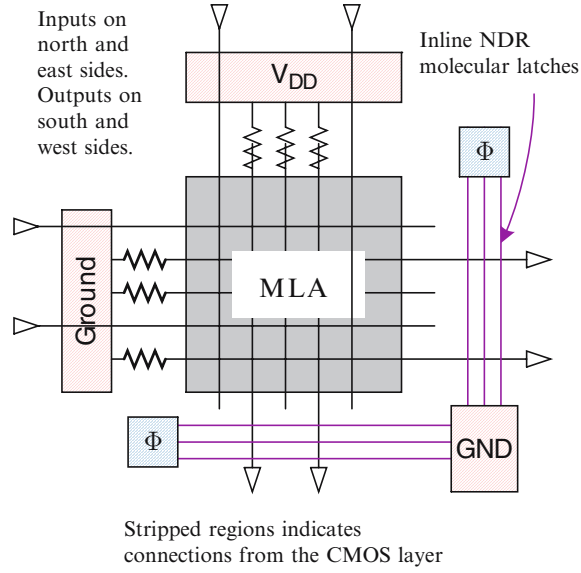


Fig. 2. Schematic of a nanoblock [3]

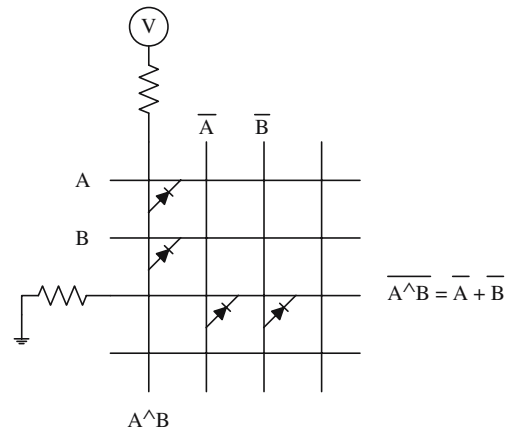


Fig. 3. An AND gate

2.2 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 four 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

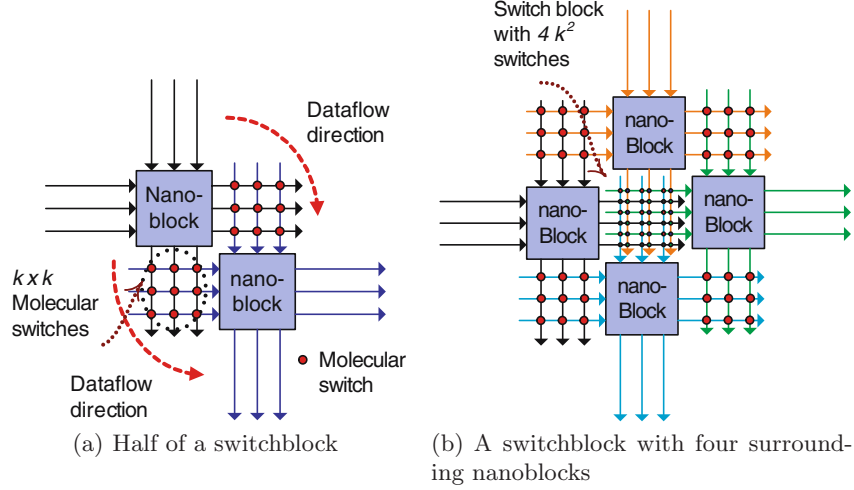


Fig. 4. Nanoblock connectivity [3]

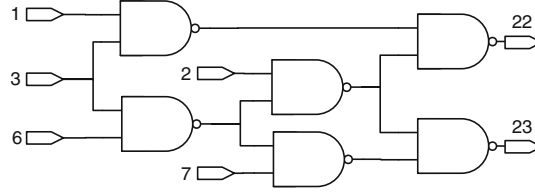
inside a switchblock, and $4k^2$ cross-points can be formed. For SE nanoblocks, a switchblock is capable of providing four directions of data flow: west to south (WS), west to east (WE), north to east (NE), and north to south (NS). WS and NE data flows can co-exist with each other in a same switchblock. 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.

2.3 Defect Tolerance

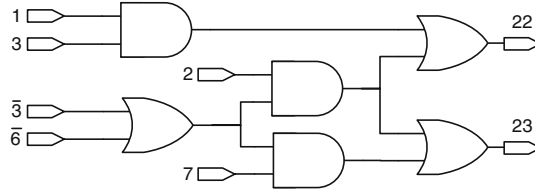
The nanofabric has a much higher defect density than standard CMOS chips due to the imprecise and nondeterministic manufacturing process. Wires will rarely all be equidistant from each other. Wires that should be parallel may be askew or they may intersect. The connections between wires may be open or wires may be shorted [6].

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 [1]. This



(a) Original schematic



(b) Equivalent schematic

Fig. 5. Schematics of c17

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 diagnosed. However, this metric is useful only for simulation; in practice, it is difficult to ascertain 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.

To illustrate how a nanofabric (with defects) can be used to implement a real circuit, we take c17, an ISCAS'85 benchmark circuit, as an example. Figure 5a shows the schematic of c17. To map it to a nanofabric, since nanofabrics cannot implement inversion logic, we first transform it into an equivalent circuit as shown in Fig. 5b. We assume that complemented input signals (e.g., $\bar{3}$ and $\bar{6}$) are available. Figure 6 shows how this equivalent circuit can be mapped to a nanofabric with defects. We assume that defective blocks (marked by crosses) are identified by a testing process. Note that even if a nanoblock/switchblock is defective, it may still be used to route signals if the exact nanowire in use is defect-free (e.g., the switchblock that routes output 22).

3 Related Prior Work

The testing of nanofabrics was first addressed in [1, 19]. The none-some-many algorithm presented in [1] creates LFSR-based signature generators from a random selection of nanoblocks. This approach however makes the unrealistic assumption that unlimited interconnect resources are available to create

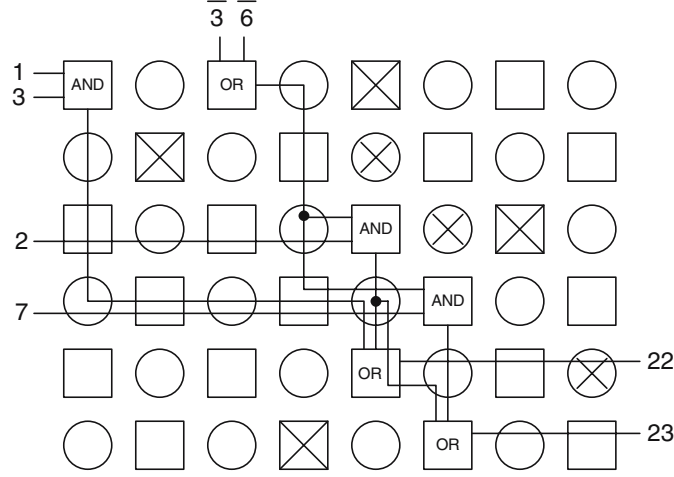


Fig. 6. The mapping of c17 to a defective nanofabric

signature generators from randomly-selected nanoblocks. Moreover, since this approach uses a large number of nanoblocks to implement LFSR-based signature generators, it is coarse-grained and it can only provide limited recovery.

The CAEN-BIST approach presented in [19] 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. Moreover, [19] 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.

Recently, a BIST method to test the nanoblocks was presented in [20]. However, this work does not address switchblocks; thus it can lead to defect-free switchblocks being deemed useless, and defective switchblocks being marked defect-free. Moreover, the recovery procedure in [20] is coarse-grained; it is limited by the fact that recovered blocks cannot be used to further increase the nanofabric recovery.

Another BIST method for nanofabrics was proposed recently in [21]; this method uses only TPGs and ORAs. The TPGs and ORAs are tested in parallel to decrease testing time and to improve recovery. However, this work does not address defects in switchblocks. In [22], an on-chip, application-specific BIST approach is described for nanoscale devices. This BIST method is performed whenever the chip is configured for a given application. It performs functional test for the blocks in the nanoarray and it eliminates the use of defect maps. However, because the BIST circuitry is implemented on-chip in the CMOS domain, the area overhead tends to be excessive compared with the size of the nano-domain circuitry. Moreover, during the BIST procedure, a large amount of memory is needed to save intermediate test results.

4 Nanofabric BIST Approach

We now present a BIST approach for nanofabric testing that exploits the re-configurability of nanoblocks and switchblocks. The nanoblocks 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. 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. In the nanofabric architecture described in [3], there are “long wires” that are used for interconnection purposes. We assume that such long wires can be used by external circuits to access blocks along the edges. Since the number of long wires is limited, multiple configurations are needed to access each of these blocks.

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, referred to FDC-1, are discussed in Sect. 5. A BUT is deemed to be defect-free only if it operates correctly in all FDC-1 configurations. In addition to FDC-1 configurations, another set of configurations is proposed to target faults in switchblocks (referred to as FDC-2). Switchblock testing is discussed in Sect. 6.

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 TGs 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 and switchblock in the fabric. Each configuration uses only one test pattern. When the test procedure is completed, a defect map can be constructed. The test procedure is discussed further in Sect. 4.2.

The method used to access ORAs affects the overall testing time. If long wires, as described in [1], are used to read the outputs of ORAs, due to restricted interconnect resources, it may not be possible to access all the ORAs simultaneously. This will make the access time dependent on the size of the nanofabric. However, in the absence of manufactured nanofabric chips, it is not clear how much time will be needed to access ORAs. Contactless measurement and testing techniques [23], e.g., electron beam testing [24] and massive observability [25], may be used to reduce the ORA access time. Embedded CMOS optical sensors may also facilitate ORA readout [26].

4.1 BIST Architecture

Figure 7a 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.

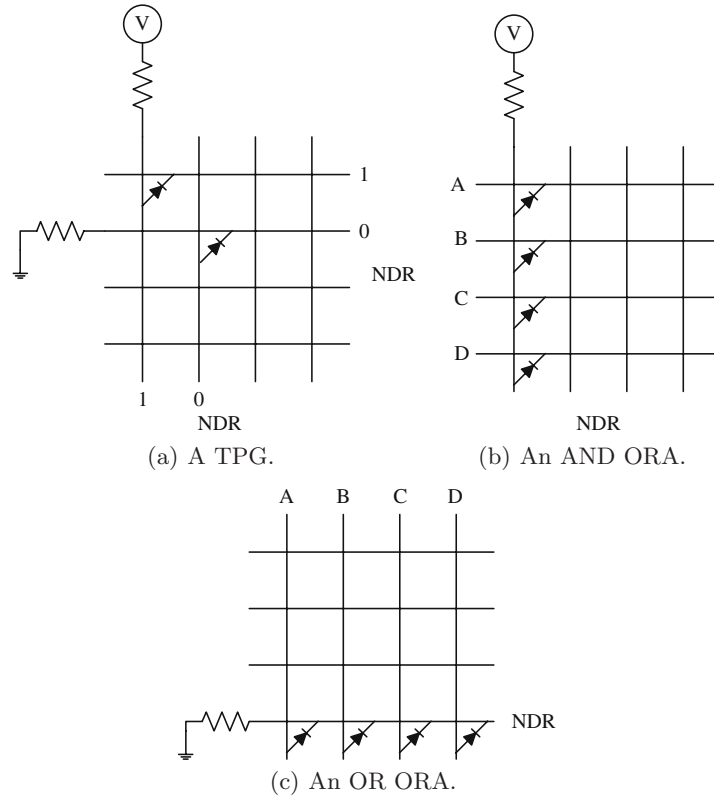


Fig. 7. Nanoblocks configured as TPG and ORA

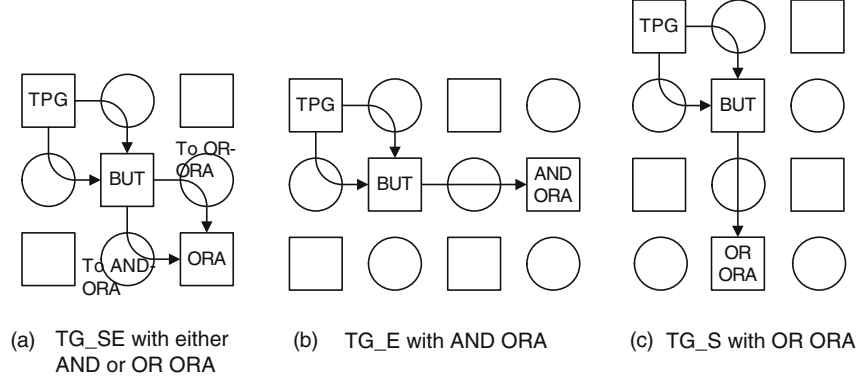


Fig. 8. Illustrating the need for different TGs

Figure 7b, c shows 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. 8a, 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. 8b) is an AND gate and the ORA in TG_S (Fig. 8c) is an OR gate, which will be discussed in Sect. 6. 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 [1, 19].

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 *fully-recovered* or defect-free only if it is defect-free in all the three partial defect maps; otherwise a nanoblock is considered as *partially-recovered* if it is defect-free in any of the partial defect maps. A partially-recovered nanoblock can be used in some specific applications but its complete functionality cannot be guaranteed.

4.2 The BIST Procedure

The BIST procedure is shown in Fig. 9. 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. The number of TG allocations is independent of the size of the fabric. For TG_SE, three

BIST Procedure: input: type of test group (TG)

1. while not all nanoblocks and switchblocks 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. 9. BIST procedure for any given TG

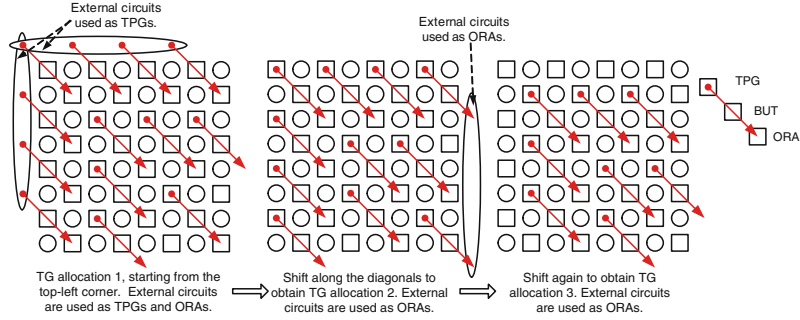


Fig. 10. TG allocations for TG_SE

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. Run the BIST procedure using TGs only for switchblock testing;
5. Generate an overall defect map;
6. Run the adaptive recovery procedure to further improve recovery.

Fig. 11. Overall BIST algorithm

allocations are needed, as shown in Fig. 10. 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.

A nanoblock cannot be recovered if any nanoblock or switchblock in the same TG is defective. However, we can use recovered nanoblocks and/or external circuits that are not adjacent to, but are reachable from the candidate defective block, as its TPG and ORA. This procedure, referred to as *adaptive recovery*, can greatly improve recovery and it is discussed in detail in Sect. 8. The overall BIST algorithm is shown in Fig. 11.

5 FDC-1: Fault Detection Configurations for Nanoblocks

FDC-1 configurations configure the TGs into different circuits to provide 100% fault coverage for stuck-at, stuck-open, bridging, and connection faults in nanoblocks. These configurations are classified into five categories.

In the FDC-1 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” [18], and (2) Resistors attached to Gnd have much smaller resistance than those attached to V_{DD} [3].

5.1 Category 1: FDCs for Stuck-At and Stuck-Open Faults

In Category 1(a) illustrated in Fig. 12a, 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} [3]. 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. 8.

Similarly, in Category 1(b) shown in Fig. 12b, all inputs are connected to Gnd or “0”. With a k -input OR gate, any stuck-at-1 fault can be detected.

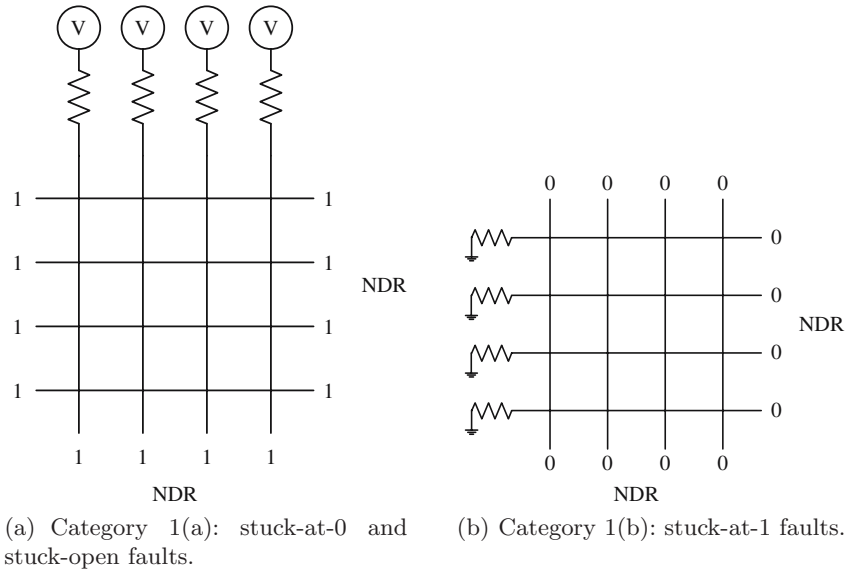


Fig. 12. BUTs for Category 1

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.

5.2 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.

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.

5.3 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” on the east side and “0” on the south side. 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.

5.4 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

Table 1. Summary of proposed FDC-1 configurations

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

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 AND 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.

5.5 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 1 lists the faults indicated with x entries that each category can detect.

6 FDC-2: Fault Detection Configurations for Switchblocks

FDC-1 configurations only address faults in nanoblocks. As shown in Fig. 4, switchblocks and nanoblocks are closely coupled with each other. Hence, if a test group generates an error for a given configuration, it is difficult to ascertain whether switchblocks or nanoblocks are faulty. On the other hand, if a test group is error-free for a given set of configurations, and if all the switchblocks and nanoblocks in the test group are exercised adequately by these configurations, we can conclude that these blocks are all defect-free.

In this section, we describe an additional set of configurations, referred to as FDC-2, to provide 100% coverage for switchblock faults.

As discussed in Sect. 2.2, a switchblock contains $4k^2$ cross-points and provides four data flow directions. For SE blocks, directions WS and NE each use k^2 cross-points, while directions NS and WE each use $2k^2$ cross-points. All the four directions need to be thoroughly tested.

Similar to [19], we use a walking binary sequence to test switchblocks. As shown in Fig. 15, to test the k^2 connections used by the WS direction, a sequence of test stimuli, 1000, 0100, 0010 and 0001, is applied to the inputs of the 4×4 switchblock. If the switchblock is defect-free, the same sequence should appear in the outputs. After the test sequence is applied, the configurations are shifted, as shown in Fig. 16. The walking sequence is repeated k times so that all connections are tested. Therefore a total of k^2 configurations are needed. This test set ensures that every connection in the switchblock is tested in both the “on” and “off” configurations. It provides 100% fault coverage for single stuck-line, bridging, and connection faults [19].

6.1 Testing the WS Dataflow Direction

In FDC-1 Category (2), the input to the BUT is simply a walking sequence of ones. If the TPG is configured to provide this sequence to the BUT, the switchblock that connects the TPG to the BUT, and provides the WS dataflow direction, can be tested; see Fig. 17. A total of k^2 additional configurations need to be added to Category (2). Only the TPG and the switchblock under test are changed in these configurations. If the test group operates correctly in this category, we can conclude that the k^2 connections of the switchblock that are used by dataflow direction WS are defect-free.

6.2 Testing the NE Dataflow Direction

In FDC-1 Category (4), the input to the BUT is also a walking sequence of ones. Therefore, we can add k^2 additional configurations to Category (4) to test the switchblock that connects the TPG and the BUT and provides the NE dataflow direction, as shown in Fig. 18.

6.3 Testing the NS and WE Dataflow Directions

To test the NS and WE directions, FDC-1 category (2) and (4) are used again. However, we need two new types of TGs, where the TPG is to the north and west of the TPG, respectively.

As shown in Fig. 19, a new type of TG, TG_SE_W, is used in order to test the WE direction. The BUT is configured as FDC-1 category (4). Although the WE direction uses $2k^2$ connections, we still need k^2 additional configurations. This is because among the $2k^2$ connections, the k^2 connections between the horizontal wires from the west nanoblock and the vertical wires from the south nanoblock are tested in the WS direction; only the other k^2 connections

between the horizontal wires from the west nanoblock and the vertical wires from the north nanoblock need to be tested (Fig. 20). Figure 21 illustrates how the NS direction is tested.

In summary, a total of $4k^2$ configurations are needed to test the four dataflow directions that a switchblock provides. Each direction requires k^2 configurations. Directions WS and NE are tested in TG_E and TG_SE respectively, and therefore can share TG allocations with FDC-1. Directions NS and WE, however, must be tested separately using TG_SE.W and TG_E.N.

7 The Detection of Multiple Faults

In this section, we discuss the effectiveness of the FDC-1 and FDC-2 configurations in detecting multiple faults in nanoblocks and switchblocks. If none of the proposed configurations can detect a specific combination of multiple faults within one test group, then there may be two cases: (1) more than one block within the test group are faulty and these faults mask each other; (2) faults inside one nanofabric or switchblock mask each other, such that the output response of the faulty block is the same as the error-free response.

While Case (1) cannot be ruled out, it is not likely in practice for the proposed BIST approach. Since the test groups are simple and each test group contains only a small number of blocks (one TPG, BUT, ORA and the associated switchblocks), any faulty block will cause the whole group to generate incorrect outputs. The probability for a faulty test group to pass a test is extremely low. Moreover, the nanoblocks in a test group are also used in other test groups in different roles, which facilitates fault detection. Therefore, in this section we will only consider Case (2).

Since FDC-2 is an exhaustive functional test for the switchblocks, in which each cross-point is tested for both on and off states, FDC-2 can detect any combinations of faults in a switchblock. If a switchblock passes all tests provided by FDC-2, then it is guaranteed to be fault-free for any number of connection faults.

Next, we consider whether multiple faults in a BUT can mask each other and become undetectable. Intuitively, since the BUT is intensively exercised by $4k+6$ FDC-1 configurations, the probability is extremely low that a specific combination of multiple faults inside the BUT can escape these tests.

7.1 Multiple Stuck-At Faults

In Category 1 of FDC-1, since the ORA directly observes the output responses of the nanowires, any number of stuck-at faults on horizontal or vertical nanowires can be detected. Therefore, FDC-1 can provide 100% coverage for multiple stuck-at faults in one nanoblock. Moreover, stuck-at faults on these wires cannot be masked by other faults, such as bridging faults and cross-point faults, in the same BUT because the responses of the nanowires

are directly observed. This implies that whenever a BUT contains stuck-at faults, it can always be identified as being faulty.

7.2 Multiple Bridging Faults

Multiple bridging faults inside a BUT, including AND- and OR-bridging faults among all nanowires, can also be detected by FDC-1. When detecting bridging faults using Category 2, 4 and 5, respectively, k configurations are applied to the BUT, and each nanowire is tested separately to see if it is bridged to any other nanowire. Therefore, if there are multiple pairs of nanowires that are bridged, they can be detected separately.

Bridging faults that involve more than two nanowires can also be detected. For example, as in Category 5 (Fig. 14b), only one nanowire is connected to 0 and all other nanowires are connected to 1. If the nanowire connected to 0 is bridged to multiple nanowires, as long as it is OR-bridged to one nanowire, it will be detected. This is also true for Category 2 and 4. Note that in this work, we only consider AND- and OR-bridging faults. We have not considered the physical phenomenon of resistive bridging faults.

A defective diode behaves either like a stuck-open switch or a stuck-on switch, depending on the bias voltage across it. If a defective diode is reverse-biased, then the two nanowires associated with this diode are either AND-bridged or OR-bridged with each other. Therefore, bridging faults can be detected even if there are defective reverse-biased diodes. Defective forward-biased diodes are simply stuck-open switches, whose existence will not affect the detection of any other fault. Hence, multiple bridging faults are detectable even when there are defective diodes in the same nanoblock.

In some cases, multiple bridging faults in a BUT may form feedback loops and turns the BUT into a sequential circuits. The detection of such faults is complex and we will not consider it in this work.

7.3 Multiple Cross-Point Faults

In Category 2, the diodes associated with the nanowire that is connected to V_{DD} are tested if they operate correctly when forward-biased. If other diodes in the same nanoblock are defective, they behave as either stuck-open or stuck-on switches. It can be seen from Fig. 13a that these defective diodes will not affect the test of the forward-biased diodes, even if they cause their associated nanowires to be AND-bridged or OR-bridged. In Category 3, all diodes are reverse-biased and tested in parallel. If there are multiple defective diodes in the nanoblock, then they may form feedback loops in the nanoblock. We do not consider such faults in this work.

In summary, FDC-1 can detect most combinations of multiple faults in a single nanoblock. FDC-1 can provide 100% coverage for multiple stuck-at faults, 100% non-feedback bridging faults among nanowires, and 100% coverage for multiple cross-point faults. Feedback bridging faults and feedback cross-point faults are beyond the scope of this work.

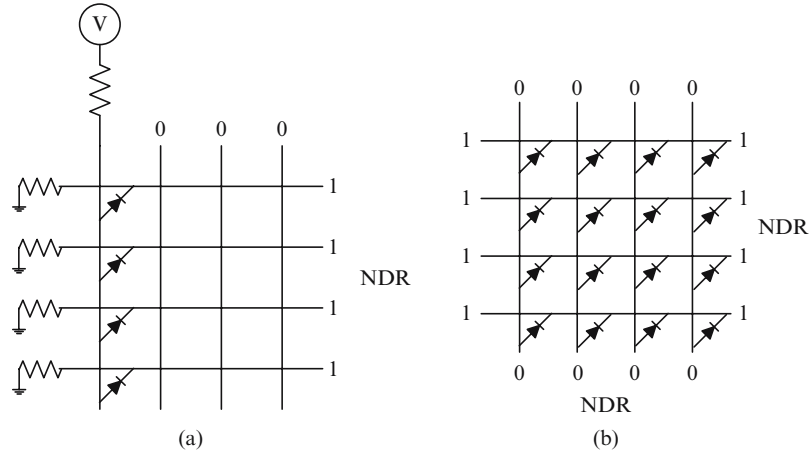


Fig. 13. (a) Category 2: forward-biased diodes faults and AND-bridging faults (v/v , v/h). (b) Category 3: reverse-biased diodes faults

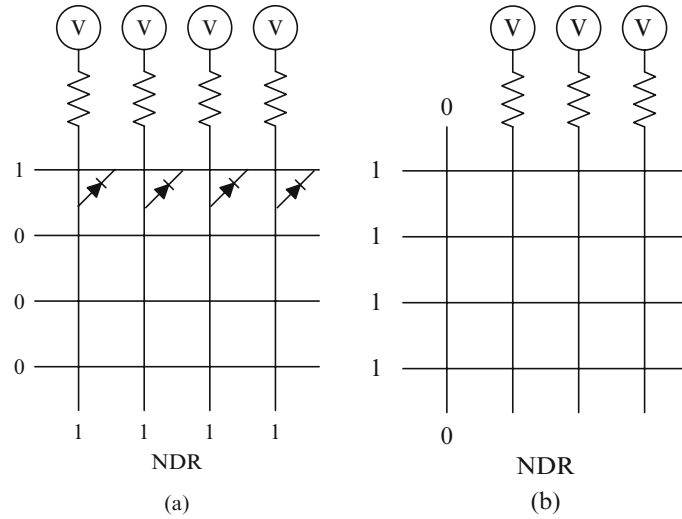


Fig. 14. (a) Category 4: AND-bridging fault among horizontal wires (h/h). (b) Category 5: OR-bridging fault

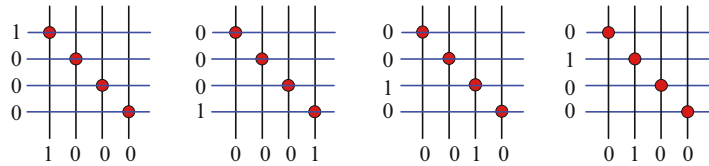


Fig. 15. A walking sequence of ones is applied to the inputs and appears in the outputs

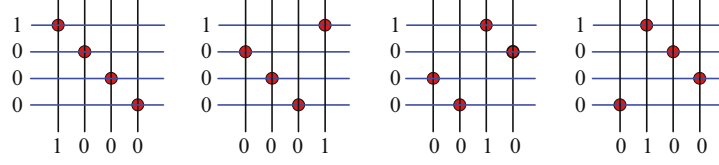


Fig. 16. The configuration of the switchblock is shifted after all test patterns have been applied and the process is repeated until all connections have been tested

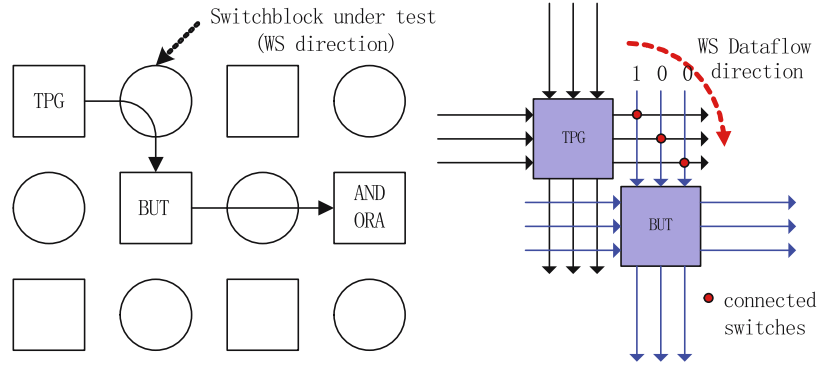


Fig. 17. Testing the WS direction of a switchblock: The BUT is configured as FDC-1 category (2), except that all inputs are from the switchblock under test. In the k^2 configurations, the TPG and the switchblock under test are changed so that the walking sequence are repeated k times and the k^2 connections are tested. FDC-1 category (2) only works in TG.E

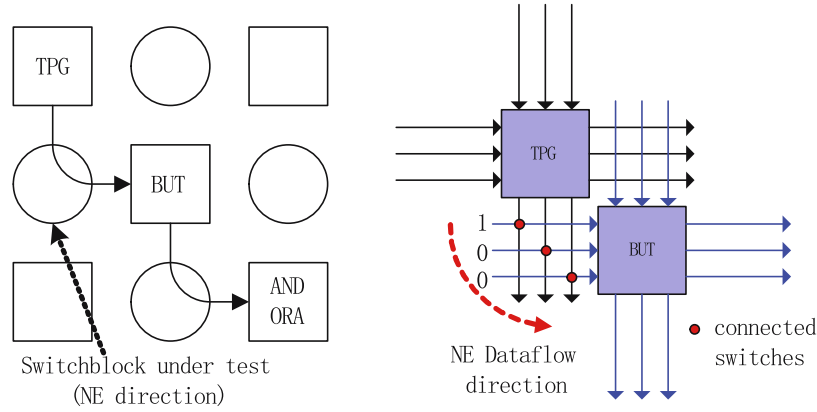


Fig. 18. Testing the NE direction of a switchblock: The BUT is configured as FDC-1 category (4), except that all inputs are from the switchblock under test. In the k^2 configurations, the TPG and the switchblock under test are changed so that the walking sequence are repeated k times and the k^2 connections are tested. FDC-1 category (4) only works in TG.SE

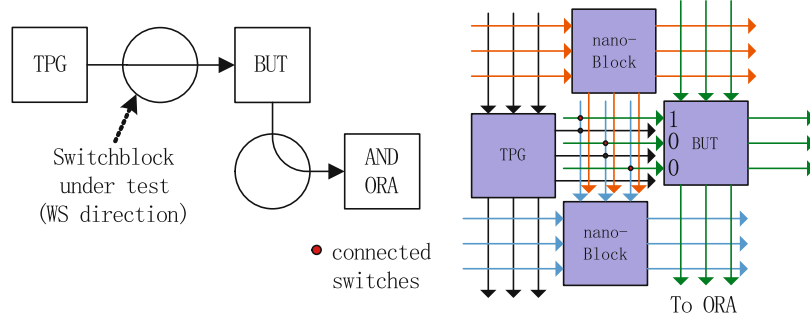


Fig. 19. Testing the WE direction of a switchblock: The BUT is configured as FDC-1 category (4). These configurations need a new TG type, i.e., TG_SE_W

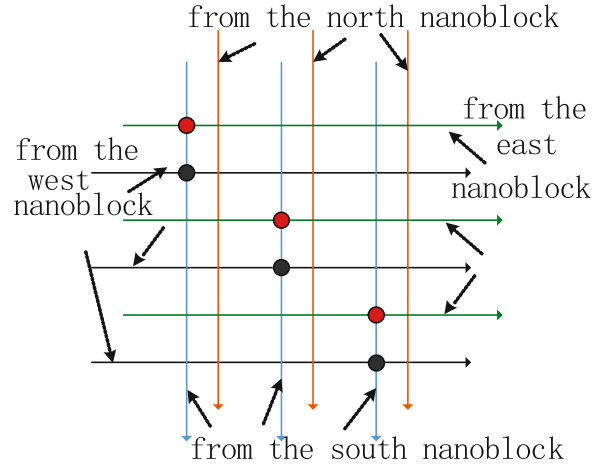


Fig. 20. Only connections between the vertical wires from the north nanoblock and the horizontal wires from the west nanoblock need to be tested in the WE direction

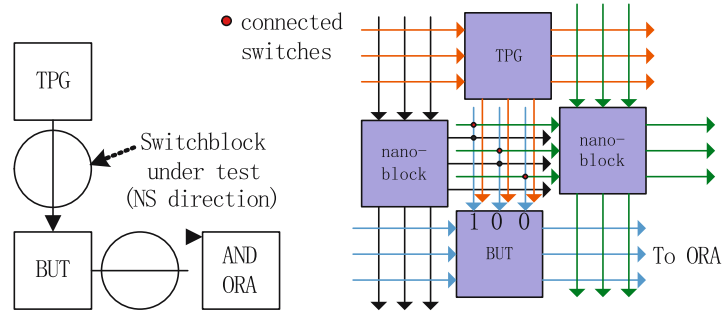


Fig. 21. Testing the NS direction of a switchblock: The BUT is configured as FDC category (2). These configurations need a new TG type: TG_E_N

8 Adaptive Recovery Procedure

As discussed in Sect. 4.2, a defect-free nanoblock or switchblock B cannot be recovered if there are faulty blocks in the same TG. However, if other blocks outside the TG can be used to form a new TG to test B , this block may be recovered. For example, as shown in Fig. 22, a new TG that is similar to TG_E but consists of non-adjacent nanoblocks is created to test the BUT whose west-side neighbor is defective. In this section, we propose an adaptive recovery procedure that dynamically generates configurations to recover blocks that are not recovered by Steps (1)–(5) in Fig. 11.

The adaptive recovery procedure only tries to recover blocks that have a high probability to be defect-free. These blocks are referred to as *candidates*. A candidate nanoblock or switchblock must have at least one of its surrounding blocks (1) fully-recovered, or (2) partially-recovered and the failing tests do not involve the candidate block itself. This is because a defective block will cause all its surrounding blocks to fail the tests whose TGs contain it.

In the adaptive recovery procedure, first the partial and overall defect maps are searched and candidates are identified. Then a search procedure is invoked for each candidate. Paths that start from the current candidate block and lead to a recovered block, another candidate block, or an edge of the nanofabric are searched, such that new TGs can be formed. The same procedure as shown in Fig. 9 is then executed to test the current candidate. To avoid using known defective blocks, the paths only consist of fully or partially-recovered blocks and other candidate blocks, as shown in Fig. 22. In this work, a simple back-trace search algorithm has been implemented.

The nanofabric is reconfigured multiple times until either all candidates are tested or a satisfactory recovery level is achieved. In each configuration, several candidates can be tested in parallel. The adaptive recovery procedure is summarized in Fig. 23.

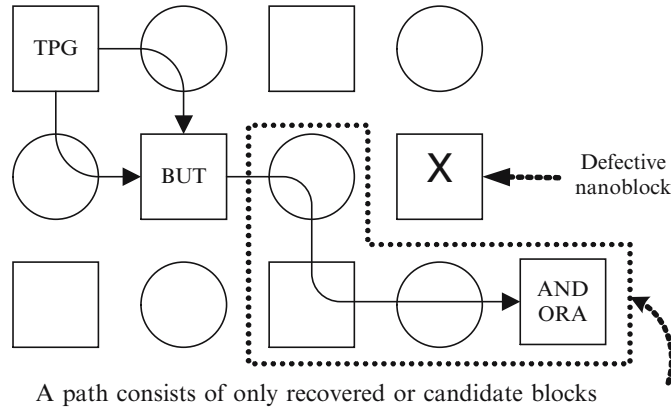


Fig. 22. A TG_E variant TG that uses non-adjacent nanoblocks

Adaptive Recovery Procedure:

1. identify candidates;
2. while (not all candidates are tested) and (satisfactory recovery not reached)
3. generate TGs for each remaining candidate;
4. while not all compatible FDCs are applied
5. apply one FDC;
6. run the test;
7. read out URA responses;
8. end while (FDC)
9. end while (candidates)

Fig. 23. The adaptive recovery procedure

9 Recovery Analysis

In order to better understand the effectiveness of the proposed BIST procedure, we derive simple upper and lower bounds of the recovery. As shown in Fig. 26, a defective switchblock causes two of its surrounding nanoblocks to become non-recoverable. Thus the minimum recovery is obtained when all defective blocks are switchblocks and each of them causes two nanoblocks to become unrecoverable. For a $n \times n$ nanofabric with defect density d , there are a total of $(1 - d)n^2$ defect-free blocks, among which $2dn^2$ defect-free nanoblocks are not recovered in the worst case. A lower bound LB on the recovery is therefore given by

$$LB = \frac{(1 - d)n^2 - 2dn^2}{(1 - d)n^2} \times 100\% = \frac{1 - 3d}{1 - d} \times 100\%$$

For $d = 0.1$ ($d = 0.2$), LB is 78% (50%). Figure 24 shows how LB varies with d .

The best recovery is achieved when all defective blocks are nanoblocks and they are not close to each other so that no switchblocks are rendered non-recoverable. Hence the upper bound is simply $UB = 100\%$. However, this upper bound is meaningful only when the defect density is below a maximum value d_{max} , such that each switchblock has at least three defect-free surrounding nanoblocks. If a switchblock has more than two defective surrounding nanoblocks, the probability that it is not recoverable is $1/3$ (i.e., the two defective nanoblocks are both at the inputs or outputs of the four dataflow directions). To derive d_{max} , consider a nanofabric consisting of m nanoblocks and m switchblocks with defect density d . Since each nanoblock is shared by four switchblocks, there should be at least $\frac{3m}{4}$ defect-free nanoblocks. Hence $m - d_{max} \times 2m = \frac{3m}{4}$ and $d_{max} = 12.5\%$. If $d > d_{max}$, the upper bound is below 100% and it depends on both the defect density and the distribution of the defective blocks; it is difficult to derive a closed-form expression for the upper bound.

In some special cases, even when d is higher than 12.5%, the recovery can still be 100%. Figure 25 illustrates a special case, in which all defective

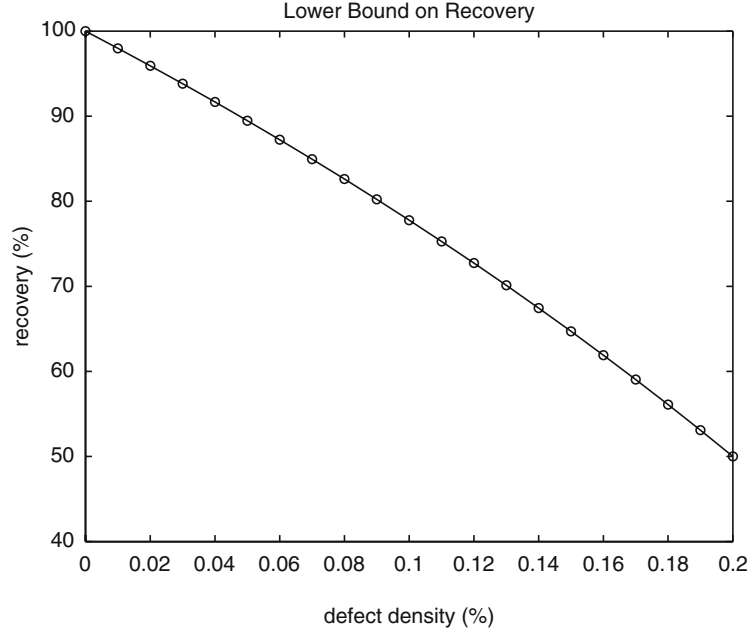


Fig. 24. Lower bound on the recovery

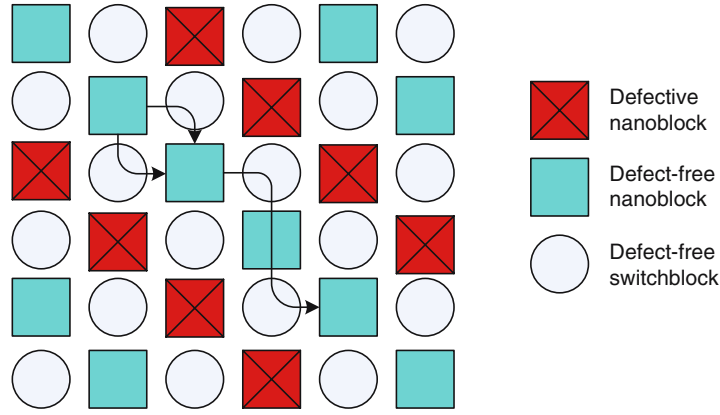


Fig. 25. Special case of 100% recovery

blocks are nanoblocks and they only appear along alternate major diagonals. In this case, the defect density $d = 25\%$. Using adaptive recovery as shown in Fig. 25, a recovery of 100% is achievable. Note that all switchblocks are partially-recovered.

10 Simulation Results

In this section, we present simulation results for the proposed BIST approach. The BIST procedure was implemented in C++ and an array of bits is used to represent the nanofabric. Blocks are randomly selected to be faulty and a simple array look-up is performed to determine if a nanoblock or a switchblock is defective.

Figure 26 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 fully or partially-recovered blocks to the total number of defect-free blocks, is 92.1%. There are 11 defective blocks, 59 fully-recovered blocks, and 23 partially-recovered blocks. Among the 82 recovered blocks, 13 blocks are recovered by the adaptive recovery procedure.

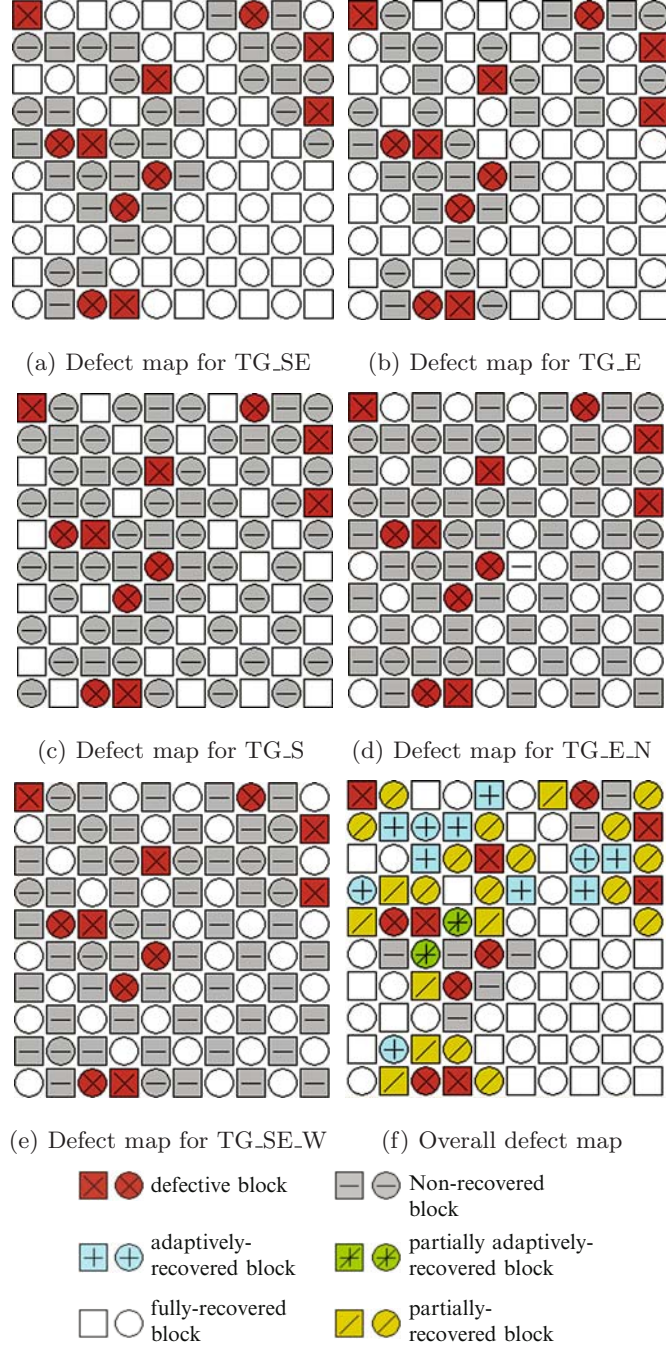
In TG_SE and TG_E, both nanoblocks and switchblocks are tested. Therefore the corresponding partial defect maps contain recovered nanoblocks and switchblocks. In other partial defect maps, there are only recovered nanoblocks or switchblocks. The adaptive recovery procedure is performed as the last step, so only the overall defect map shows blocks recovered by it.

As shown in Fig. 26, a defective switchblock consistently renders the nanoblocks to the east or south of it non-recoverable, because these nanoblocks, when configured as BUTs, cannot receive input signals from TPGs through the defective switchblock. A defective nanoblock, however, only causes its surrounding switchblocks to be partially-recovered. These switchblocks fail the tests involving the defective nanoblock, but pass other tests not involving it. Therefore, a defective switchblock has greater impact on recovery than a defective nanoblock.

Figure 27 shows how the recovery varies with defect density and the size of the nanofabric. For each value of defect density and nanofabric size, 100 simulations were performed and the average recovery as well as the maximum and minimum recovery values were determined. As shown in Fig. 27a, the average recovery is almost independent of the size of the nanofabric. This implies that the BIST procedure and the recovery algorithm scale well with the size of the nanofabric. Moreover, the average recovery is inversely proportional to the defect density. This is obvious because a defective block, especially a defective switchblock, affects the functionality of its neighbors.

Figure 27b, c show that the recoveries of small nanofabrics are more likely to be affected by the defect distribution. With the same defect density, if relatively more switchblocks than nanoblocks are defective, the recovery is likely to be lower. In large nanofabrics, the number of defective nanoblocks and the number of the defective switchblocks are more likely to be the same. The recovery values lie within the upper and lower bounds derived in Sect. 9. In particular, the recovery obtained is close to the lower bound. The minimum recoveries when $d = 0.1$ and $d = 0.2$ are about 78 and 56%, respectively.

Figure 28 shows how clustered defects affect recovery. We use two clustered defect models: (1) defective blocks are in the same row or column, and (2)

**Fig. 26.** Defect maps for a 10×10 fabric with 10% defect density

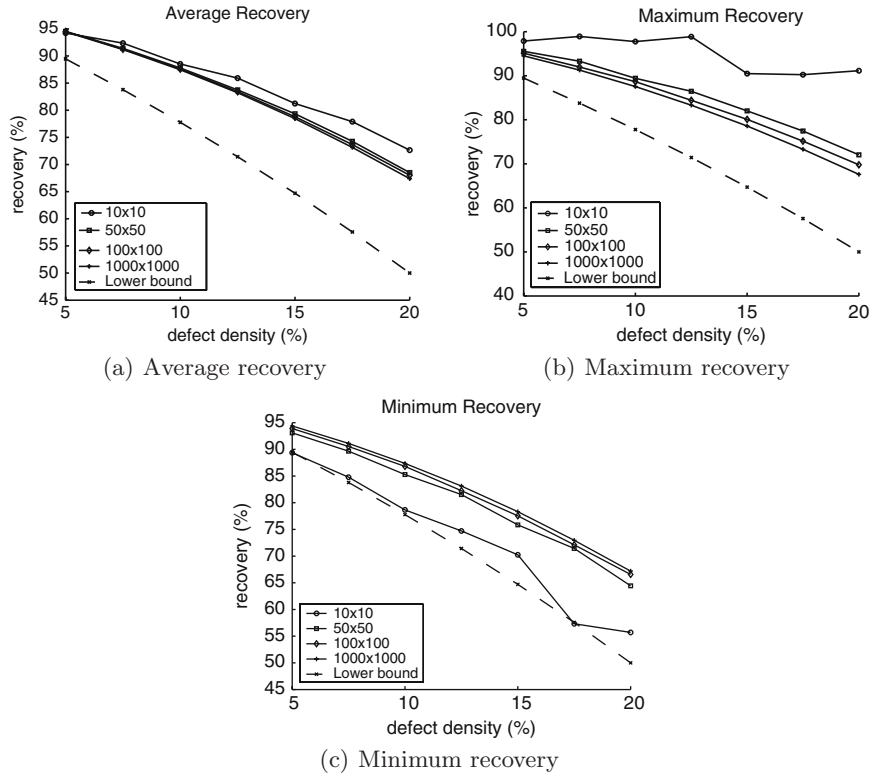


Fig. 27. Recovery results

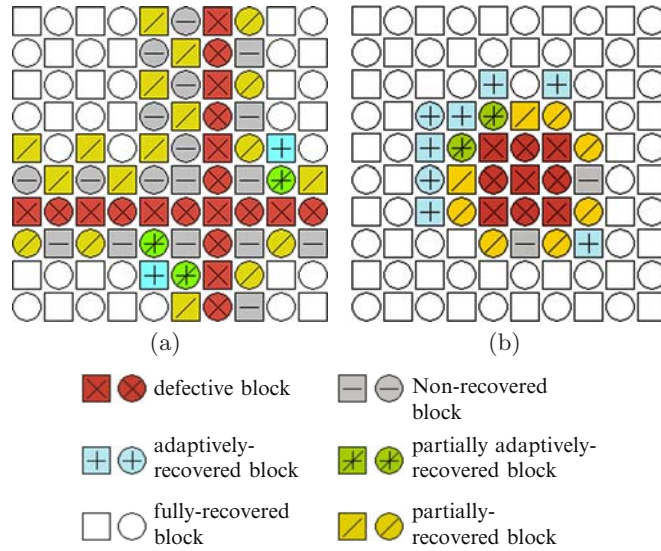


Fig. 28. Clustered defective blocks

defective blocks are inside a rectangle. The recoveries of Fig. 28a, b are 77.8 and 97.8% respectively, and the defect densities are 19 and 16% respectively. Only the nanoblocks and switchblocks adjacent to the edges of the cluster cannot be recovered. Clustered defects do not influence the effectiveness of the BIST approach.

11 Conclusions

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 and switchblocks 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 in both nanoblocks and switchblocks; these configurations provide 100% fault coverage for stuck-at, stuck-open, bridging, and connection faults. All multiple stuck-at faults, as well as non-feedback bridging and crosspoints faults can also be detected. The complexity of this algorithm and its recovery capability are independent of the size of the nanofabric (if the time required to read out output responses is $O(1)$). The recovery procedure can also utilize recovered defect-free blocks in an adaptive fashion.

References

1. M. Mishra and S. Goldstein, "Defect Tolerance at the End of the Roadmap," in *Proc. International Test Conference*, 2003, pp. 1201–1210.
2. E. J. Nowack, "Maintaining the Benefits of CMOS scaling when Scaling Bogs Down," *IBM Journal of Research and Development*, no. 2/3, Mar.-May 2002.
3. S. C. Goldstein and M. Budiu, "NanoFabrics: Spatial Computing Using Molecular Electronics," in *Proc. International Symposium on Computer Architecture*, 2001, pp. 178–189.
4. S. C. Goldstein and D. Rosewater, "Digital Logic Using Molecular Electronics," in *Proc. IEEE International Solid State Circuits Conference*, vol. 1, 2002, pp. 204–459.
5. M. Butts, A. DeHon, and S. C. Goldstein, "Molecular Electronics: Devices, Systems and Tools for Gigagate, Gigabit Chips," in *Proc. International Conference on Computer-Aided Design*, 2002, pp. 433–440.
6. M. R. Stan, P. D. Franzon, S. C. Goldstein, J. C. Lach, and M. M. Ziegler, "Molecular Electronics: From Devices and Interconnect to Circuits and Architecture," *Proc. IEEE*, vol. 91, Nov. 2003, pp. 1940–1957.
7. Y. Chen, G.-Y. Jung, D. A. A. Ohlberg, X. Li, D. R. Stewart, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, and R. StanleyWilliams, "Nanoscale molecular-switch crossbar circuits," *Nanotechnology*, vol. 14, Mar. 2003, pp. 462–468.

8. Nantero Inc., <http://www.nantero.com/>.
9. A. J. van de Goor, *Testing Semiconductor Memories: Theory and Practice*. ComTex Publishing, 1998.
10. 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.
11. M. Abramovici, E. Lee, and C. Stroud, "BIST-based Diagnostics for FPGA Logic Blocks," in *Proc. International Test Conference*, 1997, pp. 539–547.
12. C. Metra, G. Mojoli, S. Pastore, D. Salvi, and G. Sechi, "Novel Technique for Testing FPGAs," in *Proc. Design, Automation and Test in Europe*, 1998, pp. 89–94.
13. S. J. Wang and T. M. Tsai, "Test and Diagnosis of Fault Logic Blocks in FPGAs," in *IEE Proceedings: Computers and Digital Techniques*, vol. 146, 1999, pp. 100–106.
14. M. B. Tahoori, E. J. McCluskey, M. Renovell, and P. Faure, "A multi-configuration dependent testing of FPGAs," in *Proc. IEEE VLSI Test Symposium*, 2004, pp. 154–159.
15. M. Tahoori and S. Mitra, "Techniques and algorithms for fault grading of FPGA interconnect test configurations," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, Feb. 2004, pp. 261–272.
16. W. B. Culbertson, R. Amerson, R. J. Carter, P. Kuekes, and G. Snider, "Defect Tolerance on the Teramac Custom Computer," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, 1997, pp. 116–223.
17. J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams, "A Defect-Tolerant Computer Architecture: Opportunities for Nanotechnology," *Science*, vol. 280, Jun. 1998, pp. 1716–1721.
18. S. C. Goldstein and D. Rosewater, "What Makes a Good Molecular-Scale Computer Device?" School of Computer Science, Carnegie Mellon University, Tech. Rep. CMU-CS-02-181, Sep. 2002.
19. J. G. Brown and R. D. S. Blanton, "CAEN-BIST: Testing the NanoFabric," in *Proc. International Test Conference*, 2004, pp. 462–471.
20. Z. Wang and K. Chakrabarty, "Built-in Self-Test of Molecular Electronics-Based Nanofabrics," in *Proc. European Test Symposium*, 2005, pp. 168–173.
21. M. Tehranipoor, "Defect Tolerance for Molecular Electronics-Based NanoFabrics Using Built-In Self-Test Procedure," in *Proc. International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2005, pp. 305–313.
22. R. M. Rad and M. Tehranipoor, "SCT: An Approach for Testing and Configuring Nanoscale Devices," in *Proc. IEEE VLSI Test Symposium*, 2006 (to appear).
23. S. Sayil, D. V. Kerns, and S. E. Kerns, "A survey contactless measurement and testing techniques," *IEEE Potentials*, vol. 24, Feb.-Mar. 2005, pp. 25–28.
24. M. Vallet and P. Sardin, "Electrical testing for failure analysis: Ebeam testing," *Microelectronic Engineering*, vol. 49, 1999, pp. 157–167.
25. A. Mabrouk and A. Hubbard, "Design and implementation of an optical testing technique for VLSI chips using a potential-sensitive fluorescing dye," in *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 1997, pp. 568–572.
26. S. Sayil, "All-Silicon Optical Contactless Testing Of ICs," *International Journal of Electronics*, vol. 89, 2002, pp. 537–547.

Emerging Nanotechnologies
Test, Defect Tolerance, and Reliability
Tehranipoor, M. (Ed.)
2008, XII, 408 p. 200 illus., Hardcover
ISBN: 978-0-387-74746-0