# SCT: A Novel Approach for Testing and Configuring Nanoscale Devices

REZA RAD
University of Maryland
and
MOHAMMAD TEHRANIPOOR
University of Connecticut

Novel strategies are necessary to efficiently test and configure emerging reconfigurable nanoscale devices, in addition to providing defect tolerance. This is mainly due to the high defect densities that are expected for these devices. Among different approaches, reconfiguration-based defect avoidance has proven to be a practical solution. However, configuration time, test time, and defect-map size remain among the major challenges for these new devices. In this article, we propose a new approach (called SCT) that simultaneously performs test and configuration. The proposed method uses a built-in self-test (BIST) scheme for test and defect tolerance. The method is based on testing reconfigurable nanoblocks at the time of implementing a function of a desired application on that block. The SCT method considerably reduces the total test and configuration time. It also eliminates the need for storing the location of defects in a defect map on- or off-chip. The presented probabilistic analysis results show the effectiveness of this method in terms of test and configuration time for architectures with rich interconnect resources. Also, a Verilog simulation model is developed for crossbar-based nano-architectures. This model is used to implement several MCNC benchmarks based on the proposed SCT method. The simulation results demonstrate efficiency of the method in terms of test time and yield under different defect rates.

Categories and Subject Descriptors: B.8 [**Performance and Reliability**]

General Terms: Reliability, Design

Additional Key Words and Phrases: Configuration and testing, reconfigurable nanoscale devices, fault tolerance, nanowire, crossbar

**ACM Reference Format:**

Rad, R. and Tehranipoor, M. 2008. SCT: A novel approach for testing and configuring nanoscale devices. ACM J. Emerg. Technol. Comput. Syst. 4, 3, Article 14 (August 2008), 24 pages. DOI = 10.1145/1389089.1389094 http://doi.acm.org/10.1145/1389089.1389094

Authors' addresses: R. Rad, University of Maryland, Baltimore, MD 20742; M. Tehranipoor, Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269; email: tehrani@engr.uconn.edu.

## 1. INTRODUCTION

As CMOS scaling becomes more and more difficult and expensive, research investments are growing to allow exploring a wide range of emerging devices and technologies. Developments in nanoscale technologies provide the hope that the current trend of integration of electronic devices can be continued. These technologies are becoming mature enough to implement devices with switching or transistor properties in few nanometer dimensions. Different molecules with switching and rectifying properties have been reported that can provide diode-logic circuits [Chen et al. 2003a]. Many of these molecular switches are programmable and can save their state of connection or disconnection; hence, it is possible to create configurable structures using these switches. Carbon nanotubes (CNTs) have been synthesized with one nanometer in diameter and micrometers in length [Dekker 1999] and several FET implementations are reported in the literature based on carbon nanotubes [Javey et al. 2003; Wind et al. 2002]. Logic circuits implemented based on these nanotube FETs have also been reported in Bachtold et al. [2001]. Also, nanowires (NWs) with diameter as small as 3 nanometer and lengths of few hundred micrometers are reported in Cui et al. [2001] and Morales and Lieber [2002]. Several experiments to arrange nanowires in array structures are reported in Huang et al. [2001] and Whang et al. [2003]. Directed self-assembly and self-alignment techniques are mainly used to create these structures. *Nano-imprint* lithography [Chou et al. 1997] is also used to implement nanowire array structures.

Various architectures are suggested in the literature based on available nanoscale device and assembly technologies. An island-style architecture for nanoscale circuits is proposed in Goldstein and Budiu [2001] with clusters interconnected in an array structure. The island-style architecture provides rich interconnect resources between clusters. A PLA-based FPGA architecture is presented in Dehon and Wilson [2004]. Their architecture is based on using a technique, called modulated doping of nanowires, to create an interface for accessing nanowires through CMOS wires. A cell-based architecture and an interface scheme, using special metal pins implemented on surface of substrate to provide the contacts with nanowires laying on the substrate, are proposed in Strukov and Likharev [2005].

Due to high defect densities (up to 10%) in assembly and device technology, approximately all fabricated devices using these technologies will have a handful of defects. Therefore, these circuits must be designed with high testability and efficient defect-tolerance schemes. The reconfigurability usually inherited by these circuits from their basic components, namely reconfigurable molecular switches, can provide static fault-tolerance for these devices. It has been shown in the *Teramac* custom computer that reconfigurable architectures can tolerate high defect densities [Culbertson et al. 1997].

A commonly proposed reconfiguration-based defect-tolerance method is that of requiring a defect detection process and storing the locations of all such defects in the device in the form of a defect map. In the configuration phase, defective components must be avoided [Mishra and Goldstein 2003; Dehon et al. 2003]. Several techniques based on using built-in self-test (BIST) have

been proposed to extract the location of faulty blocks in nanoscale architectures [Brown and Blanton 2004; Tehranipoor 2003; Wang and Chakrabarty 2005].

There are several difficulties with these approaches in dealing with the high defect densities of nanoscale devices. Locating all defects in a reconfigurable architecture with high defect density is a very challenging and time-consuming task. Even if effective methods could be found to determine the exact location of all defects in such architectures, storing this information will require very large memories. It will not be practical to use on-chip nanoscale resources as memory to store the defect map due to its unreliability. On the other hand, using on-chip CMOS memory for the defect map will result in considerable area overhead, making it prohibitively expensive and therefore impractical to store on-chip. Also, it will not be feasible to ship along with chip its own defect map.

One possible solution to this problem is to add the ability to perform both configuration and test to customers' programming tool. This means the tool should first apply the tests and extract the defect map, and then configure the device for a specific application. Implementing an on-chip BIST circuit or using an on-chip microprocessor to perform test and configuration will significantly speed-up this process, but it will still be very time consuming to find the location of all defects in a chip with an extremely large number of blocks. This process should probably be repeated every time a chip is reconfigured because of the time-dependent defects in nanoscale devices, and also due to the very large memory requirements for storing the defect map for all chips (different chips will have different defect maps). As this discussion shows, manufacturing test of nanoscale devices will face challenges in terms of creating, shipping, and storing the defect map.

Application-dependent testing of nanodevices can be considered as another possible solution. However, there are fundamental differences between using these techniques for FPGAs [Tahoori 2004] and for nanodevices, due mainly to the high defect rate of the latter. In application-dependent testing of FPGAs, an application is first completely configured on the FPGA and then it is tested. However, due to high defect densities in nanoscale devices, it is not possible to first complete the configuration of the application and then apply the test. By pursuing this approach, the probability of having a fault-free configured application will be very low due to the defects found in those blocks used for configuring the application.

### 1.1 Contribution and Article Organization

In this article we propose a technique that performs the configuration and testing of nanodevices simultaneously, called SCT. The application is first divided into smaller functions implementable on nanoblocks ($f_i \in \{f_1, f_2, ..., f_T\}$.) Then the function $f_i$ is configured into a nanoblock and tested using a BIST procedure. If configuration of $f_i$ on the assigned block is fault free, the technique proceeds to another block, configures it with next function, and applies the test again. If $f_i$ shows faulty behavior, it will be configured into another block and tested again. The proposed approach is very efficient in terms of the overall time to configure and test the circuit and also in tolerating defects. Also, the

requirement of storing the defect map on/off-chip is eliminated because there will be no need to find the exact location of defects. In this technique a block may be defective, but the function $f_i$ configured into it can nonetheless have a fault-free operation. This method is architecture independent and can be applied to all nanoscale architectures with high defect-rate conditions having rich interconnect resources.

The rest of the article is organized as follows. Section 2 presents an overview of prior related work. Nanowire-based crossbars are briefly described and their application as programmable logic blocks is discussed. Recently proposed test and defect-tolerance methods for architectures developed based on the crossbar are also reviewed. Section 3 presents the main idea of the proposed SCT technique. The steps required for testing blocks in an architecture are described. Also, the interconnect requirements of the proposed method are discussed and a method is suggested for testing interconnects between the functions of the configured application. Section 4 presents a probabilistic analysis to estimate the average amount of time required to configure and test a circuit on a nanoscale architecture using the proposed SCT method. Results of the time analysis are compared with the approximations made for a previously proposed test method [Wang and Chakrabarty 2005]. In Section 5, a Verilog simulation model for these devices is presented and simulation results for implementing a few MCNC benchmark circuits on a crossbar-based device under different defect rates are shown. The results suggest that SCT can provide high yields under high defect rates and moreover demonstrate that the timing requirements for performing SCT will remain quite low. Section 6 briefly discusses a few important issues, including a functional testing approach used in SCT, circuit performance considerations in the proposed defect avoidance methods, and application-dependent versus manufacturing testing for reconfigurable nanoscale devices. The work ends with conclusions in Section 7.

## 2. OVERVIEW OF PRIOR WORK

### 2.1 Fabrication Feasibility

One of the main obstacles in the application of emerging molecular devices and nanowires, or of nanotubes in the current technology of integrated circuits, is the difference between fabrication methods of state-of-the-art CMOS devices and that of those new nanodevices. CMOS fabrication is a top-down technology based on photolithography and is not generally applicable to the emerging devices; most of the implementation methods proposed for new devices are bottom-up. In other words, low-level circuit components must be fabricated first, and then placed on the required positions to build a circuit.

Various techniques have been developed for integration of nanowires, nanotubes and molecular components. These include nanolithography [Whang et al. 2003], nano-imprint techniques [Chou et al. 1997], and atomic-force microscopy manipulation. Also, directed self-assembly through fluids is widely under investigation [Huang et al. 2001]. Although atomic manipulation techniques are capable of exact placement of nanoscale components, they cannot

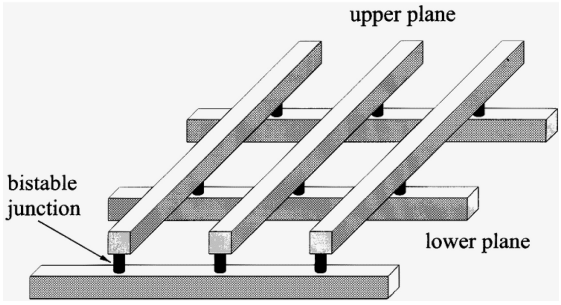


Fig. 1. Crossbar of nanowires with molecular switches in the crosspoints [Stan et al. 2003].

be used in mass production due to the time-consuming process of manipulating individual components. Nano-imprint and directed assembly techniques can be used to arrange multiple components in one step. However, these techniques cannot be used to create irregular structures from nanoscale components. Implementation of arrays (crossbars) of nanowires or nanotubes that is based on these methods has been reported in Huang et al. [2001].

### 2.2 Crossbars

Nanowire crossbars fabricated with a layer of molecular particles between the two sets of crossing wires are proposed as a potential alternative for implementing electronic circuits in Chen et al. [2003b]. Figure 1 shows a simple diagram of a crossbar. The molecular components are placed as a layer between the horizontal and vertical wires, and have special electrical properties. Their resistances can be tuned through applying appropriate voltages to them. The difference in resistance of the two states of these molecules is reported to be several orders of magnitude. Rectifying properties for these molecules are also reported in Heath and Ratner [2003].

This suggests that an array, comprised of nanowires and with molecular switches between them, can potentially be used as a programmable device. A crosspoint can be set into a diode connection or a disconnected state through applying programming voltages to the corresponding wires. Figure 2 shows a PLA built using a crossbar of wires. Diode connections between input lines and vertical lines provide the product terms and diodes configured between vertical lines and output lines create the sum of the products in the output. Figure 2 shows the implementation of functions $f = abc + \bar{a}\bar{b}c$ and $g = a\bar{b}c + \bar{a}bcd$ on the PLA.

### 2.3 Nanoscale Architectures

Several architectures have been proposed for reconfigurable crossbar-based electronic devices. An island-style structure is used in Goladstein and Budiu [2001] which is based on nanoblocks and switchblocks made of nanowire crossbars. This architecture assumes availability of long wires as interconnect resources between nanoblocks. It also suggests using NDR inline latches in the nanoblocks to make them capable of implementing sequential circuits.

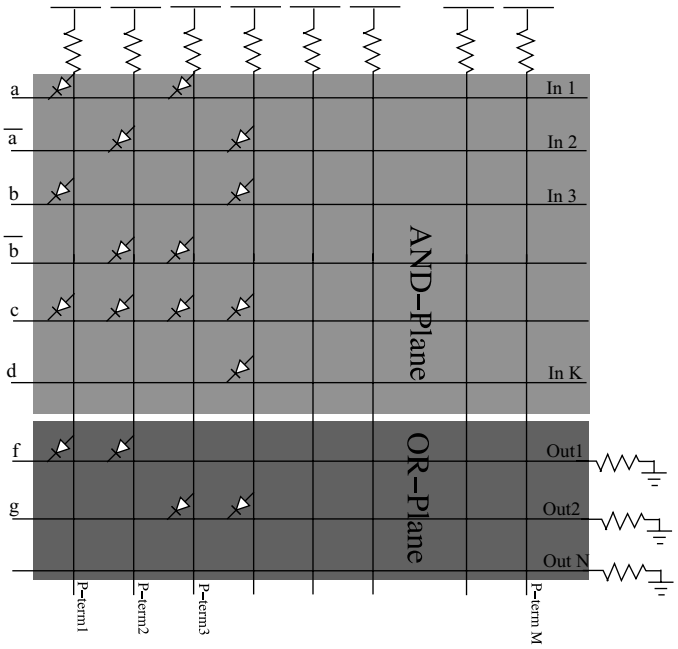


Fig. 2. PLA-based implementation. Functions $f = abc + \bar{a}\bar{b}c$ and $g = a\bar{b}c + \bar{a}bcd$ are implemented as two examples.

A programmable logic array (PLA)-based FPGA made of nanowire crossbars is presented in Dehon and Wilson [2004]. Nanowire doping techniques are proposed to provide an access mechanism to the crossbars, which is required to program the molecular switches of each crossbar PLA. The access mechanism is a critical part of all crossbar-based architectures. This is due to the fact that programming the crossbars requires applying voltages to the nanowires. Therefore, they should be accessible through programming circuitry that can be implemented in CMOS. Providing the required access to individual nanoscale wires is challenging and several techniques have been proposed to implement this interface [Dehon et al. 2005; Kuekes and Williams 2000].

In Rad and Tehranipoor [2006a] the authors proposed a crossbar-based implementation of logic-block clusters similar to those used in CMOS FPGAs. Also an interface scheme was proposed for accessing nanoscale crossbars through CMOS interconnect and programming resources. Simulation results for implementing MCNC benchmark circuits on the resulting FPGAs showed advantages in terms of area and performance compared to scaled CMOS FPGAs.

### 2.4 Test and Defect Tolerance

Test techniques have been proposed for extracting defect location information of crossbar-based devices. Techniques proposed in Brown and Blanton [2004], Tehranipoor [2005], and Wang and Chakrabarty [2005] are based on the architecture suggested in Goldstein and Budiu [2001]. In Wang and Chakrabarty [2005] it is suggested that the nanoblocks of the architecture can be configured

to comprise a test pattern generator (TPG), block under test (BUT), and output response analyzer (ORA). Several fault detection configurations are proposed to detect faults of different types in BUT. The array of nanoblocks and switchblocks is then partitioned into test groups containing a TPG, BUT, and ORA. These test groups can be configured to perform the test. The method proposed in Tehranipoor [2005] is conceptually similar to the one in Wang and Chakrabarty [2005]. The main difference is in the structure of test groups, which contains only a TPG and an ORA in this case. The simplified test groups result in reduced test and configuration time. Also additional test configurations are proposed to detect the faults in switchblocks. The technique proposed in Brown and Blanton [2004] performs testing of an array of nanoblocks in a wavelike scheme. Each nanoblock on a diagonal of the array tests its neighboring blocks and stores the status of these blocks. In other words, the defect map of the device will be stored on the device itself.

The structure of test groups in the proposed test techniques depends on the architecture of the device. Therefore, the test groups in these techniques can be modified to work for other architectures. Also, the fault model used in these methods is based on models defined for CMOS technology. However, due to fundamental differences between these technologies, a more appropriate fault model may be required for nanoscale crossbar-based devices.

## 3. SIMULTANEOUS CONFIGURATION AND TEST (SCT) METHOD

As discussed in Section 1, locating all defects in a nanodevice with very high density ($10^{10}$ gate-equivalent/cm$^2$) and high defect density (up to 10%) will be a challenging and time-consuming task. In this article a novel method is proposed for testing and configuring circuits while avoiding the time-consuming process of locating all defects. We assume that the architecture has rich interconnect resources and is able to provide efficient access to its logic blocks through its input/output interfaces. Architectures proposed in Dehon and Wilson [2004], Dehon et al. [2005], and Rad and Tehranipoor [2006b] all provide the required rich interconnect resources.

One of the methods suggested for FPGA BIST is to configure a number of blocks of an FPGA as test pattern generators (TPGs) and some other blocks as response analyzers (RAs) [Stroud et al. 1996]. The remaining configurable blocks can be selected as blocks under test (BUT) to receive the test patterns from TPGs and send the outputs to RAs.

The proposed method in this work is conceptually similar to the one proposed for FPGAs in Stroud et al. [1996]. However, we consider the TPG and RA to be components of the BIST circuit and implemented in CMOS scale to provide test patterns and analyze the responses. The detailed architecture of the proposed BIST circuit for testing configured blocks is described in this section and interconnect testing issues will be discussed in the next. The main difference between our method and the FPGA BIST methods is that in ours the goal of testing is not to confirm the correct functionality of a BUT for all functions. Here, the goal is to ensure correct operation of each function ($f_i$) of an application that is configured into one of the blocks in the device.

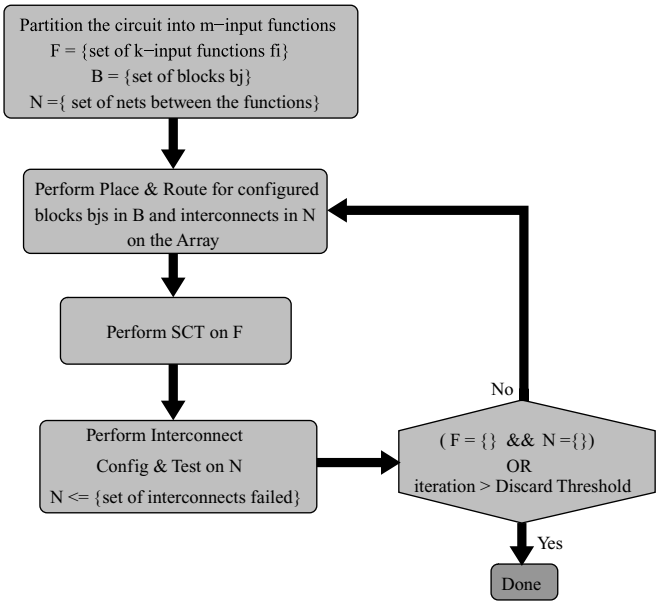


Fig. 3. The steps required to configure and test a device based on the SCT method.

In the SCT procedure, testing is not performed to locate all defects in all resources of a reconfigurable architecture. Instead, each block $b_j$ of the architecture is tested for a specific function of the application, that is, $f_i$, after $f_i$ is configured into the block $b_j$. The applied test here just checks the correct operability of the configured function $f_i$, rather than diagnosing all defects of the block $b_j$. So, there might be defects in block $b_j$ (defective molecular switches or nanowires), but, so long as these defects do not cause any malfunctioning, the function $f_i$ is identified as fault free. In other words, creating the function $f_i$ on a block $b_j$ requires just a subset of all nanowires and switches available in the block. So, if the defective components of the block are not used in configuration of $f_i$, then the function can operate without failure. Therefore defects in the block will be tolerated.

Figure 3 shows various steps taken to configure and test a device through the proposed method. First, the desired circuit/application must be partitioned into $m$-input functions. These functions are implementable both on the LUT of the BIST circuit and on crossbar blocks of the device. Next, a placement-and-routing (PNR) tool will assign a function (i.e., $f_i$) to block $b_j$ of the array and find the routing paths between functions. After assigning $f_i$ to $b_j$, SCT can configure function $f_i$ on its assigned block $b_j$ and test it. Details of this step will be discussed in the following. The set of functions that fail the test will be stored for future reference. In the next step, the interconnects between functions will be tested and the set of interconnects that fail the test is stored as well.

In the next iteration of the procedure, placement and routing of the blocks and interconnects will be performed again for the set of functions and interconnects that failed in the previous iteration. In these steps, a PNR algorithm uses

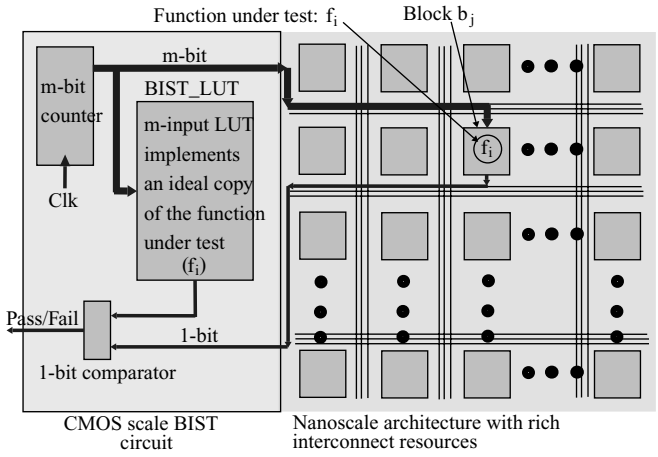


Fig. 4. The proposed CMOS BIST circuit used to test nanodevices.

redundant blocks and interconnect resources of the array to assign remaining functions and interconnects. If there are no longer any redundant resources available, those blocks that failed in implementing a function in one phase may be used to implement another function in another phase.

Once all the functions and interconnects are configured and tested on the device, the procedure will end successfully. However, if the number of iterations for implementing the circuit on the device exceeds a specified limit *(Discard-Threshold)*, then the chip will be discarded.

In each step of SCT, one of the logic functions of the circuit ($f_i$) is configured on its assigned block ($b_j$) and paths from the block to the BIST circuit will also be configured. The function $f_i$ is also configured into the LUT of the BIST circuit (see Figure 4). The BIST circuit can simply apply an exhaustive set of test patterns ($2^m$) to the function and test its functionality. If the implemented function passes the test, then it can be reliably used in the circuit. This process of selecting a function, configuring it onto a block of the device, creating connections between the configured block and the BIST circuit, and testing the function will be repeated for all functions of the application. If a function fails the test then the PNR will assign another block to the function in the next phase and SCT will be repeated for the function on the new block.

Figure 4 shows the proposed BIST circuit. It is composed of an $m$-bit counter, an $m$-input LUT, and a comparator, resulting in low BIST-area overhead. The BIST circuit is assumed implemented in reliable CMOS scale.

### 3.1 Configuration and Test for Logic Functions

Figure 5 shows the SCT procedure for configuration and testing functions of a circuit configured into blocks of a nanodevice. Each function $f_i \in F$ is assigned and configured into a block of the device ($b_j \in B$)(**Config**($b_j$ with $f_i$)). It is also configured into the LUT of the BIST circuit (**Config**(BIST_LUT with $f_i$)).

```
\\ Define:
\\ Set of blocks: B={b_1, b_2, .., b_M} ;
\\ Set of functions: F={f_1, f_2, .., f_T} ;

Foreach (f_i ∈ F) {
  Config (BIST_LUT with f_i);
  Config (b_j with f_i);
  \\b_j is a block of the device that is assigned to f_i through PNR
  Config (interconnects from b_j to BIST circuit)
  RunCounter();
  \\This applies 2^m test patterns to exhaustively test f_i on b_j
  Compare (output of BIST_ LUT with results obtained from f_i implemented on b_j);
  if (f_i failed)
    Recored f_i to be re-placed on another block in the next step of procedure;
}
```

Fig. 5. The SCT procedure for configuration and testing functions of a circuit on blocks of a nanodevice.

Interconnect resources of the device are configured to provide temporary paths between block $b_j$ and the BIST circuit (**Config**(interconnects from $b_j$ to BIST circuit)). These paths will be used to apply test patterns to the block $b_j$ configured with $f_i$ and return the results from $b_j$ to the BIST circuit. After configuring the paths, the counter used in BIST provides $2^m$ test patterns, both to the LUT of the BIST circuit and to the block under test, namely $b_j$, (**Run Counter()**). The comparator used in the BIST circuit compares the results of applying every pattern to the BIST's LUT and the function implemented on block $b_j$. If function $f_i$ fails, then it will be recorded and replaced in the next PNR step (see Figure 3). Once testing of the block is finished, the paths from block $b_j$ to the BIST circuit will no longer be used. Therefore, they can be returned to the set of available interconnect resources in the device.

### 3.2 Configuration and Testing of Interconnects

To complete the configuration of the application, interconnects between the functions $f_i$s should be configured and tested. If the interconnect resources of the nano-architecture are fabricated in CMOS scale, a CMOS test strategy such as one of those proposed in Harris and Tessier [2002] can be used to ensure the fault-free status of the interconnects. In case the fault on where CMOS wires is detected, the chip can be discarded. However, this will not result in high yield reduction because of the low defect density of CMOS technology compared to that of nanowire crossbars. Interfaces between CMOS interconnects and nanowires are presented in the architectures proposed in Strukov and Likarev [2005], Rad and Tehranipoor [2006b], and Dehon and Naeimi [2005]. The reliability and low defect density of CMOS wires provide considerable simplicity to the test and configuration process of nanoscale architectures.

If the interconnects used in the nanoscale architecture are made of high defect-rate components like nanowires, then a preprocessing test step is required to confirm their fault-free functionality. In this step, interconnects must be tested and fault-free connections should be determined. Since the structure of

```
\\ Define:
\\ Set of blocks: B={b1, b2, .., bM} ;
\\ Set of functions: F={f1, f2, .., fTf} ;
\\ Set of interconnects between functions: N={n1, n2, .., nTn} ;

\\Ik should connect blocks bx and by
\\function fx is configured and tested on bx and
\\fy configured and tested on by through previous step
\\output of fx will be connected to an input of fy through Ik
Foreach (Ik ∈ N) {
Config (interconnects from bx inputs to BIST circuit)
Config (interconnects from by inputs to BIST circuit (except the one that comes from fx))
Config (Ik from bx output to by input)
Config (BIST_LUT with fy);
Set (inputs of fx for having output value 1)
RunCounter(); \\ Apply inputs (test patterns) to fy
Compare (output of BIST_ LUT with results obtained from fy);
Set (inputs of fx for having output value 0)
RunCounter(); \\ Apply inputs (test patterns) to fy
Compare (output of BIST_ LUT with results obtained from fy);
if (Ik failed)
   recored Ik to be re-routed or fx and fy to be re-placed in the next step of procedure
}
```

Fig. 6. The proposed procedure for configuration and testing interconnects between functions of a circuit.

nanoscale architectures is predicted to be very regular, testing the interconnects can be done in a reasonable amount of time and the results can be passed to the programming device to be used during the configuration and test process described earlier.

Based on the SCT approach, configuration and testing of interconnects between functions of the circuit can be performed as described in Figure 6. For testing an interconnect $I_k$ between two functions $f_x$ and $f_y$ that have been configured and tested on blocks $b_x$ and $b_y$, respectively, the paths from these functions to the BIST circuit must be configured. Interconnect $I_k$ is configured from the output of $f_x$ to the input of $f_y$. The BIST will then apply an input to $f_x$ to force its output to 1. Then test patterns are applied to the other inputs of $f_y$ and its output is observed. The same steps will be repeated when the output of $f_x$ is set to 0. If the outputs of $f_y$ are correct, it can be concluded that $I_k$ is transferring correct values from $f_x$ to $f_y$. In case $I_k$ fails a test, it is recorded and through the next PNR step rerouted, or $f_x$ and $f_y$ can be placed on new blocks and $I_k$ rerouted. The test and configuration steps should be repeated for them. Figure 7 illustrates the aforesaid method.

### 3.3 Parallel Implementation of SCT

The BIST circuit, in general, is very small. The low area overhead of this BIST circuit provides the opportunity of parallel implementation of these testers on the chip, so that at any time more than one function can be implemented

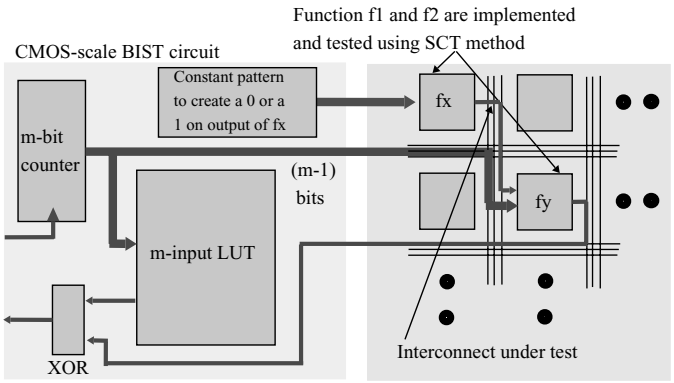


Fig. 7. Testing interconnects between the implemented functions $f_i$s with a modified BIST. The BIST circuitry includes a new reconfigurable block that generates input patterns to set any function's output to "1" or "0".

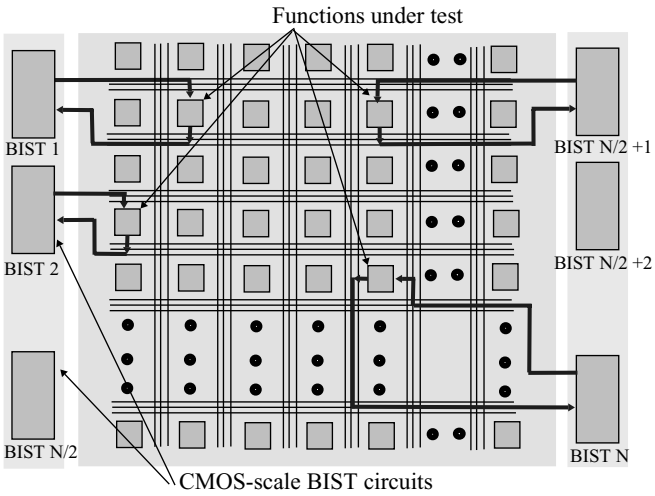


Fig. 8. Using multiple BIST circuits for testing nanodevices.

and tested on the device, as shown in Figure 8. In this case, more than one function and more than one block of the device will be selected at each step of block configuration and test. When multiple BIST circuits are implemented, test and configuration time will considerably reduce. Analysis presented in the next section shows the reduced time due to implementing parallel BIST circuits.

## 4. ANALYSIS OF THE SCT METHOD

In this section, an analysis is presented to show the efficiency of the proposed method. We assume that the interconnect structure of crossbar-based devices is similar to the nanowire crossbar blocks considered in the article. Therefore, a mechanism similar to SCT is expected to work for interconnects as well.

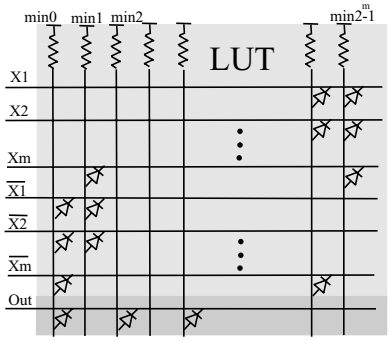


Fig. 9. Implementation of an $m$-input function on a $(2m+1) \times 2^m$ crossbar configured as a lookup table. Function $F(x_1, x_2, .., x_m) = \sum Minterms(0, 2, 4)$ is implemented as an example.

However, providing a detailed analysis for implementing SCT on interconnects requires detailed information about the interconnect architecture, switch boxes, and clusters. Therefore, we have limited our investigation to nanowire-based logic blocks.

In this analysis we first calculate the fault probability in function $f_i$ configured into a block of the device, namely block $b_j$. Using this probability we estimate the average number of blocks that must be configured to find a fault-free implementation for the function; hence, the average number of clock cycles required for configuring and testing function $f_i$ can be obtained. Finally, for an application with $T$ functions ($\{f_1, f_2, \ldots, f_T\}$), the average number of clock cycles required to configure and test all functions can be calculated. The results are compared with the number of cycles required to apply the BIST method presented in Wang and Chakrabarty [2005].

As Figure 9 shows, an $m$-input function can be implemented on a $(2m+1) \times 2^m$ crossbar of nanowires configured as an $m$-input lookup table (LUT). Since diode logic cannot provide signal inversion, the complement of the $m$ input signals should either be applied to the block or created inside the block. As seen in the figure, switches on the junctions of vertical nanowires with the first $2m$ horizontal nanowires are configured to provide the minterms. The switches on the junctions of vertical nanowires with the last horizontal nanowire ($Out$) are configured to provide the sum of minterms specified by the function.

The average probability of fault in a function configured on a LUT is calculated in the following. Let's assume that $P_o$ denotes the probability of a stuck-open fault in a molecular switch, $P_c$ shows the probability of a stuck-closed fault in a molecular switch, and $P_w$ is the probability of a defect in one of the nanowires.

The probability of an implemented minterm being faulty is given by

$$\begin{aligned} P_{(faulty\ \ minterm)} &= 1 - P_{(fault\text{-}free\ \ minterm)} \\ &= 1 - (1-p_c)^m (1-p_o)^{(m+1)} (1-p_w)^{(2m+2)}. \end{aligned}$$

The probability of an $m$-input function with $x$ minterms being faulty is obtained from

$$\begin{aligned} P_{(faulty\ \ function)} &= 1 - P_{(fault\text{-}free\ \ function)} \\ &= 1 - [(1-p_c)^m(1-p_o)^{(m+1)}(1-p_w)]^x(1-p_w)^{(2m+1)}. \end{aligned}$$

The average probability of having a fault in a function ($P_{ff}$) and the probability of successful implementation ($P_{si}$) of a function on a block after a number of repeated configurations ($N_r$) will be

$$P_{ff} = \frac{\sum_{x=1}^{2^m} P_{(faulty\ function)}}{2^m}$$

$$P_{si} = (1-P_{ff})P_{ff}^{(N_r-1)} \ \Rightarrow\ N_r = 1 + \frac{\log \frac{P_{si}}{1-P_{ff}}}{\log P_{ff}}.$$

Moreover, $N_r$ can also be defined as the average number of blocks that must be configured to find a fault-free implementation of a function $f_i$ on the device with a probability higher than $P_{si}$. The average number of switches to be configured for a function can be calculated as the average number of minterms in a function multiplied by the number of switches for each minterm. As seen in Figure 9, there are $(m+1)$ switches for each minterm to be configured, so the average number of switches to be configured for a function (denoted as $N_{sf}$) is

$$N_{sf} = \frac{m+1}{2^m}\sum_{x=1}^{2^m} x = \frac{(m+1)(2^m+1)}{2}.$$

We assume that the access and configuration structure of the architecture is capable of configuring $N_{cs}$ switches in each cycle. It should also be noted that $2^m$ tests must be applied to each of the configured functions so as to test it exhaustively. Therefore, the average number of cycles required to configure and test an $m$-input function with a success probability higher than $P_{si}$ would be

$$\#\ Config\ \&\ Test\ Cycles\ (prob \geq P_{si}) = N_r \cdot 2^m + \frac{N_r \cdot N_{sf}}{N_{cs}}.$$

The first term in the preceeding equation represents the number of cycles required for testing the configured functions ($2^m$ test patterns are applied every time a function is configured). The second term shows the number of cycles required for configuring the function. For a circuit with $T$ $m$-input functions, the time for performing the SCT procedure, assuming $N$ parallel BIST circuits (as shown in Figure 8), can be calculated as

$$\#\ Cycles\ (Parallel - SCT) = N_r \times \left(\frac{T}{N}\right) \times \left(2^m + \frac{N_{sf} \cdot N}{N_{cs}}\right). \tag{1}$$

In this equation, the first term of the sum is the number of cycles required to apply $2^m$ test patterns through $N$ parallel BIST circuits to $N$ configured functions, and the second term of the sum is the number of cycles required

to configure the functions into the blocks through configuration circuitry ($N_{cs}$ switches in each cycle).

To compare this with the number of cycles required to test a nanoscale architecture using the previously proposed test methods, we calculate the estimated cycles for testing a circuit with the same specifications mentioned earlier ($T$ $m$-input functions) based on the BIST method proposed in Wang and Chakrabarty [2005]. As reported in Wang and Chakrabarty [2005], the number of configurations required for testing a $K \times K$ nanoblock of a nanofabric is equal to $4K + 6$ and the number of configurations required for testing a $K \times K$ switchblock of a nanofabric is $4K^2$. If we implement an $m$-input function as shown in Figure 9, then the number of switches in a $K \times K$ block should be approximately equal to that used in the LUT structure shown in Figure 9. Therefore, $K$ can be related to $m$ calculated as $K = \sqrt{(2m+1)2^m}$.

To make a correct comparison, here we use the same assumptions and parameters used in the analysis of our SCT method. We assumed use of CMOS-scale interconnects for the architecture; hence, the required number of cycles for test and configuration of the interconnects was omitted from the calculations. Therefore, to keep the same conditions as Wang and Chakrabarty [2005], we assume the switchblocks of the architecture to be fault free. So, we omit the $4K^2$ configurations required for testing each switchblock in the architecture. We also assume the same target circuit with $T$ $m$-input functions, and we assume that the same $N_{cs}$ number of switches can be configured in parallel. The total number of configurations for testing a block, based on method presented in Wang and Chakrabarty [2005], will be

$$\textit{Cycles of BIST process for a block} = \frac{(4K+6)}{N_{cs}}.$$

Also assuming the same $p_w$, $p_c$ and $p_o$ parameters for the fault probabilities of nanowires and switches, the probability of a $K \times K$ nanoblock being fault free can be calculated as (there are $2K$ nanowires and $K^2$ switches in each block)

$$P_{(\textit{fault-free block})} = (1-p_w)^{2K}(1-p_c)^{K^2}(1-p_o)^{K^2}.$$

The average number of blocks that should be tested to find $T$ fault-free blocks will be

$$\textit{Total number of blocks to be tested} = \frac{T}{P_{(\textit{fault-free block})}}.$$

Hence, based on our analysis, the number of cycles required for the BIST method proposed in Wang and Chakrabarty [2005] will be calculated as

$$\begin{aligned} &\#\textit{Cycles}\,(\textit{BIST}\ [\text{Wang and Chakrabarty 2005}]) \\ &= \frac{T}{(1-p_w)^{2K}(1-p_c)^{K^2}(1-p_o)^{K^2}} \times \frac{4K+6}{N_{cs}}, \end{aligned} \tag{2}$$

where $K = \sqrt{(2m+1)2^m}$.

Once the defect map is created and stored, it can be used to configure the functions of the design on the device. Therefore, additional cycles will be required to implement the functions of the circuit based on the calculated defect

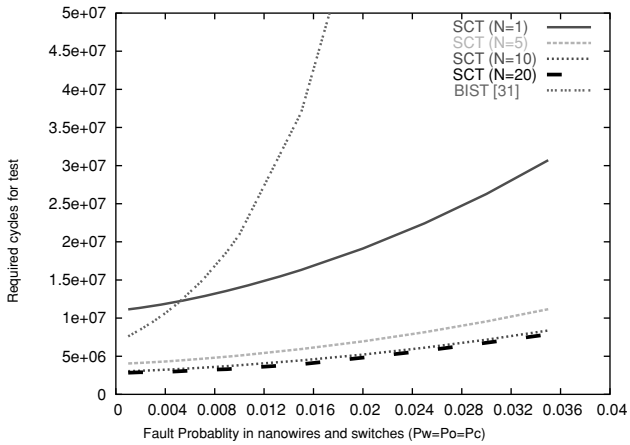


Fig. 10. Number of cycles required for testing a nanodevice using the SCT method and the BIST method proposed in Wang and Chakrabarty [2005] (shown as BIST[31]) when $m = 3, T = 1000000, P_{si} = 0.999, N_{cs} = 5$.

map. Also, as discussed in Wang and Chakrabarty [2005], applying the test patterns in this BIST process (as well as in almost all other proposed methods) will not result in finding the exact location of the faulty block. Hence, there will be some fault-free blocks that are wrongly determined as faulty during these test methods. This means that the recovery, that is, the ratio of detected fault-free blocks to all fault-free blocks, in these test methods is lower than 100%. To achieve high recovery it is suggested that a recovery procedure should be applied to exactly locate the faulty blocks. This diagnostic procedure will require a considerable number of reconfigurations that will significantly increase the test time. In the analysis presented earlier, we assumed that the test process can exactly locate the faulty blocks (in other words, we assumed 100% recovery). This assumption causes the BIST formula to show test cycles lower than the actual number of cycles required by the BIST method presented in Wang and Chakrabarty [2005].

The number of cycles for the SCT procedure and for the BIST process proposed in Wang and Chakrabarty [2005] are compared for different defect probabilities and design parameters, and the results are presented in Figures 10, 11, and 12. In each of these figures, constant values are assigned to $m$, $T$, $N_{cs}$, and $P_{si}$. Different values are assigned to $N$, that is, to the number of parallel working BIST circuits, and to the defect probabilities of the molecular switches and nanowires. As the results demonstrate, in most cases the required cycles for our SCT procedure are considerably fewer than those of the BIST method in Wang and Chakrabarty [2005].

As Figure 13 shows, increasing $N$, namely the number of concurrent BIST circuits, will result in decreasing the overall time of the SCT procedure and increasing the BIST area overhead. However, increasing $N$ will not decrease the overall SCT-procedure time linearly. If the configuration circuitry is capable of configuring a constant number of switches in each cycle ($N_{cs}$), then as $N$ increases the configuration time will become the dominant part of the SCT

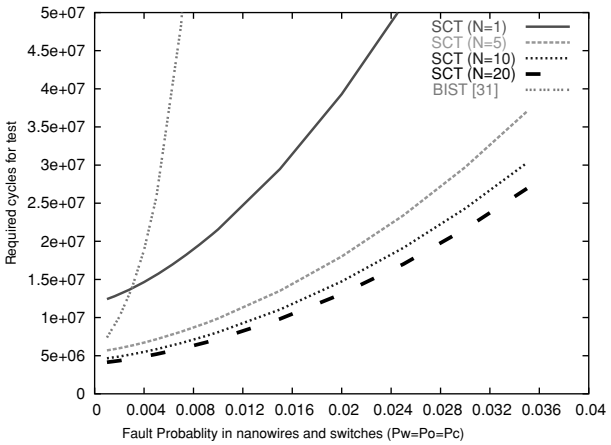


Fig. 11. Number of cycles required for testing a nanodevice using the SCT method and the BIST method proposed in Wang and Chakrabarty [2005] (shown as BIST[31]) when $m = 4, T = 500000, P_{si} = 0.999, N_{cs} = 5$.

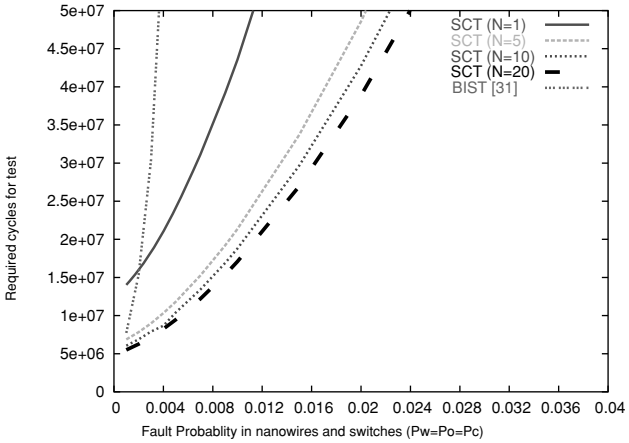


Fig. 12. Number of cycles required for testing a nanodevice using the SCT method and the BIST method proposed in Wang and Chakrabarty [2005] (shown as BIST[31]). Parameters: $m = 5, T = 250000, P_{si} = 0.999, N_{cs} = 5$.

procedure; and this part cannot be reduced through adding the number of parallel BIST circuits ($N$). Adding more parallel configuration circuitry, that is, increasing $N_{cs}$ from 5 to 20 as shown in the figure, will reduce the configuration time. This in turn reduces the SCT time, but specifically the time of the procedure and not the configuration time. The area overhead resulting from adding the parallel BIST circuits will be still small compared to the area required to store the defect map as in previously proposed BIST methods [Brown and Blanton 2004; Tehranipoor 2005; Wang and Chakrabarty 2005].

It can be argued that application of test methods that try to localize defects on the crossbar array is a one-time process, while SCT is a process that must be performed every time an application is being programmed on the device. The

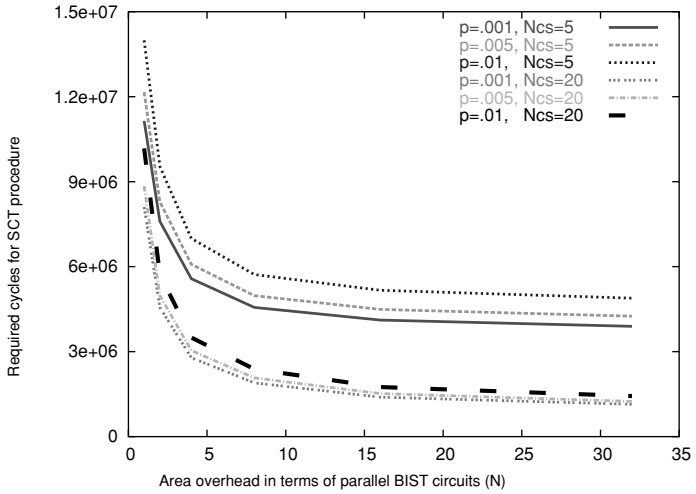


Fig. 13. SCT procedure time versus $N$, namely the number of parallel BIST circuits, which is a measure of BIST area overhead ($m = 3, T = 1000000, P_{si} = 0.999$).

assumption of a one-time process for generating the defect map may have to be dropped if the rate of time-dependent defects is considerable in nanodevices. It is also worth mentioning that the time required for testing each block based on the proposed SCT method is rather small, especially for small blocks. The major part of the processing time will be spent on finding new blocks and remapping the failing blocks on them. This time will be comparable to the time that a defect-map-based approach will require to find the correct mapping through referring to a defect map.

## 5. SIMULATION RESULTS

A set of simulations are performed to evaluate the proposed SCT method in terms of test time and capability in mapping benchmarks to defective crossbar-based devices. A Verilog model is developed for crossbars made of nanowires and molecular switches. In this model, each component of the crossbar, that is, the product term, input nanowire, output nanowire, and molecular switch, is defined as a module and connected to other components (i.e., modules) to form the crossbar. A defect flag is associated with each of these components and a randomly generated defect file will apply random defects to different components of the crossbar model. In this model we have considered closed and open defects for molecular switches and line defects for nanowires used as input, output, and product terms. Therefore, operation of the crossbar depends on the applied inputs and random defects created for the components. This crossbar model can be used both as a PLA and as a LUT by tuning the number of its inputs and product terms (or minterms).

A Perl program is written to use the Verilog model and implement five MCNC benchmarks, {i1, i2, i3, i4, i5}, using the SCT approach. The SIS package is used to map the benchmarks into $m$-input functions. Then the Perl program creates the required configuration, input, and defect files to implement each function of

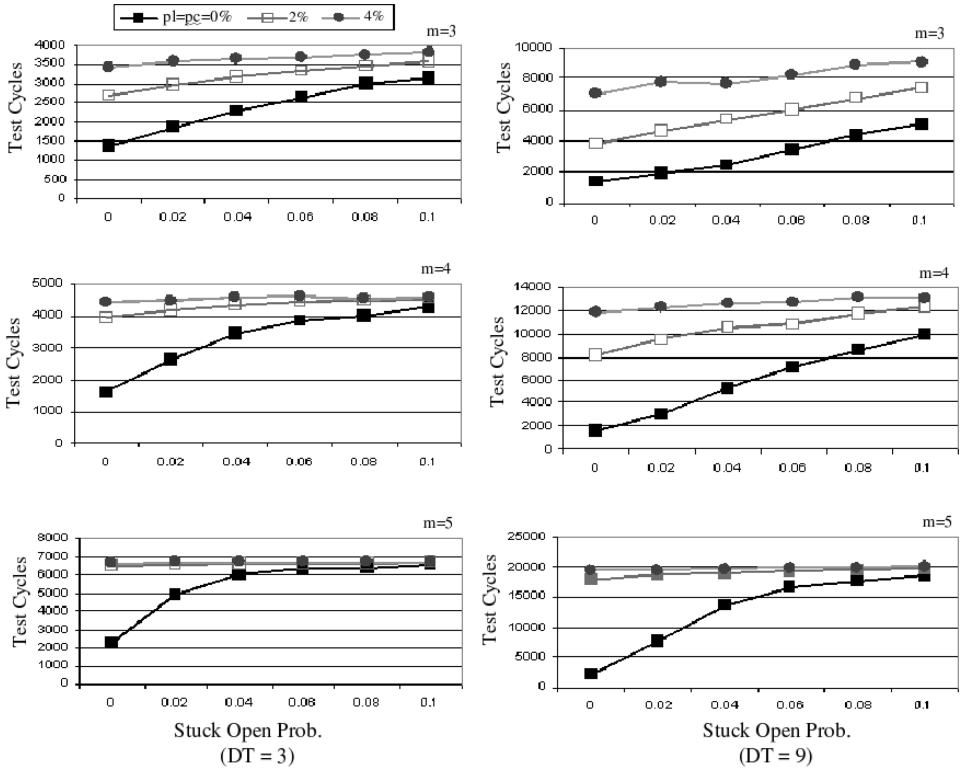


Fig. 14. Average number of test cycles for implementation of five MCNC benchmarks through the SCT method under different defect rates, LUT sizes, and *Discard–Thresholds*.

a benchmark on the Verilog model of crossbar. By running the VCS simulator, test patterns will be applied to the model and test results will be stored in a file. The Perl program compares the obtained results with those expected from the function. If the function passes the test, the next function of the benchmark will be configured; otherwise the implementation of the function on a new crossbar will be examined. This will be repeated until either the function passes the test or the number of iterations becomes more than *Discard–Threshold*, set by the user. The latter case will be counted as yield loss and the device will be discarded. These steps will be repeated for all functions of the benchmarks, and the average values of the test cycles and resulting yield are obtained.

The simulation is repeated for various defect probabilities and for different LUT sizes. Figure 14 shows the average test cycles resulting from simulations. The graphs consist of simulation results for different LUT sizes and values of *Discard–Threshold*.

As expected, increasing the defect rate will increase the configuration and test time. Also, an increase in LUT size and *Discard–Threshold* will considerably increase the number of cycles. A stuck-open fault in one switch will not affect functionality of other components of the crossbar. Therefore, stuck-open faults are independent probabilistic events. On the other hand, when a stuck-closed or line defect occurs, not only the defective component but also other

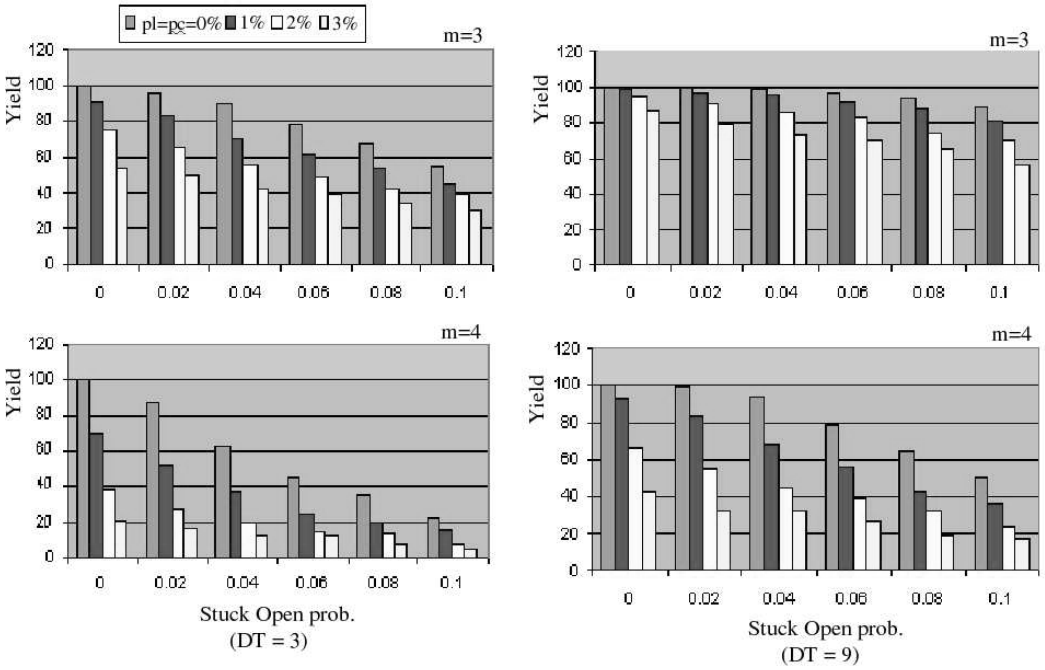


Fig. 15. Average yield of implementation of five MCNC benchmarks on LUT blocks using the SCT method under different defect rates, LUT sizes, and *Discard–Threshold*s (DT).

components connected to the defective part will not be able to function correctly. Therefore, an increase in probability of stuck-closed and line defects ($p_c$ and $p_l$) will considerably increase test cycles.

Figure 15 shows the simulation results for the achieved yield under different defect rates, LUT sizes, and *Discard–Thresholds*. As seen in the figure, yield decreases by an increase in defect rate. It also decreases for larger LUTs. This is due to the fact that in an $m$-input LUT, all possible $2^m$ minterms are configured on the block while only a few will be used to implement each function. Therefore, each LUT contains a considerable number of hardware components not used in the function implemented on them, but which may cause failures in operation of the function. The resulting yield of the SCT procedure can be significantly improved if PLA blocks are used instead of LUTs. This is because in $m$-input PLAs, the number of product terms is much less than $2^m$. Therefore, having less defect-prone hardware results in a greater chance of fault-free implementation of functions on PLAs. To explore the yield of SCT on PLA blocks, the same set of experiments were repeated on devices made of PLAs. Figure 16 shows the obtained yield using our simulation. Comparison of this figure with Figure 15 shows that the yield is considerably increased for PLAs.

Finally, Figure 17 demonstrates the variation in required test cycles and achieved yields for different discard thresholds under a 10% stuck-open probability and with stuck-closed and line-defect probabilities ranging from 0% to 3% each. As seen in the figure, increasing the discard threshold will improve the yield at a cost of greater numberes of test cycles.

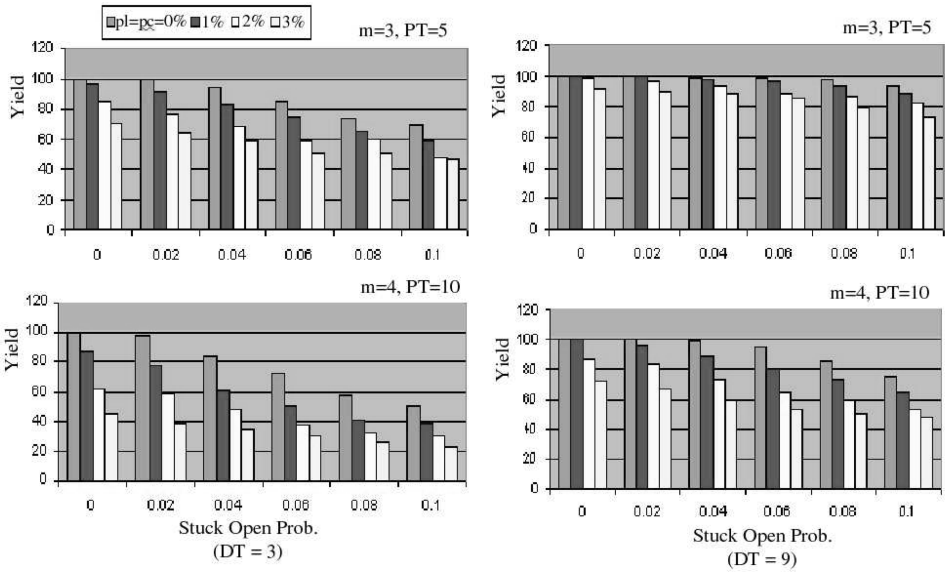


Fig. 16. Average yield for implementation of five MCNC benchmarks on PLA blocks using the SCT method under different defect rates, PLA sizes, and *Discard-Threshold*s. PT represents the number of product terms in a PLA compared to $2^m$ in a LUTs.

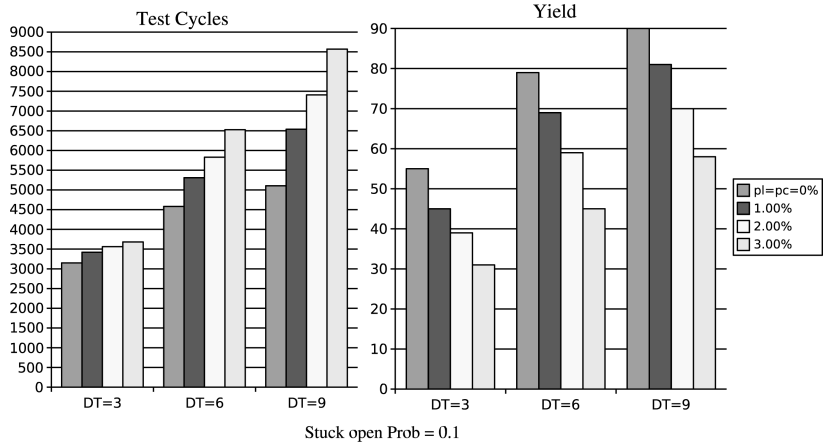


Fig. 17. Required test cycles and achieved yield for different values of *Discard–Threshold*, for block size $m = 3$.

## 6. DISCUSSION

### 6.1 Functional Testing of Nanoblocks

The method proposed here for testing blocks of a configurable crossbar-based device performs an exhaustive functional test for the function implemented on each block. Such an approach is selected to make the test independent of the fault model of nanoscale components. Fault models for crossbar-based

devices are proposed in the literature [Tehranipoor 2005; Wang and Chakrabarty 2005]. The proposed models are close to those used in CMOS-scale devices and include stuck-open and stuck-closed faults for molecular switches and bridging faults for nanowires. However, more accurate fault models may be necessary to cover all the physical phenomena encountered in the different defect types that occur in fabrication of crossbars. Fundamental differences in fabrication of these devices with the CMOS fabrication process will result in different types of defect. High defect rates raise the probability of having multiple faults on the same block. Therefore, appropriate fault models must be devised to target all these effects.

Due to lack of such accurate fault models, a pseudo-exhaustive functional testing approach is used in this article. This approach can handle multiple fault and also all different types of faults that may occur in crossbar-based blocks. This is due to the fact that a block passes the test only if it performs exactly the expected function. Even when multiple defects exist on the crossbar, if they do not cause any error in output of the implemented function, the block would pass the test.

### 6.2 Performance Criteria in Defect Avoidance Methods

As described in Section 1, reconfiguration-based defect avoidance has been suggested as a practical method to tolerate high defect rates in nanoscale devices. However, this approach faces two main difficulties in mass production of devices. The first problem, which is addressed in this work, is creating and storing the defect map. As discussed, obtaining accurate defect location information in a high-density device with high defect rates is a challenging task. Also, storing the defect map of each chip, either on- or off-chip, will be difficult. The method proposed in this article, removes the requirement of creating and storing a defect map through combining the configuration and test phases.

The second problem related to defect avoidance approaches is that of performance. Since defect locations are random on each chip, in order to avoid defects, a placement-and-routing (PNR) tool should map the circuit into a chip while considering its specific defect map. This results in chips with different performances. Therefore, the PNR tool should confirm that all mapped circuits on different chips will satisfy all the performance criteria. If a PNR cannot meet the performance requirements, then either the chip should be discarded (yield loss) or it may be considered as a lower-performance product.

It can be expected that traditional PNR algorithms may face difficulties in meeting the performance requirements on a device if limitations on selecting blocks are to be considered based on the defect map. The PNR algorithms that can be used along with SCT must be able to allocate device blocks dynamically in each phase of the SCT. In each phase, once the allocation of blocks is done, the PNR algorithm must evaluate the achievable performance. This type of dynamic block assignment and performance evaluation can be effectively used to ensure the configuration of high-performance applications and to minimize the variation in performance of different copies of the same application.

### 6.3 Application-Dependent Testing and Manufacturing Test

The proposed method can be used both as a manufacturing and an application-dependent test. Test methods for ASIC devices are based on extracting a set of test patterns that can detect a high percentage of faults in a device under test. However, in the case of reconfigurable devices, it is very difficult to find a generic set of test patterns that provide acceptable coverage for all configurations of the device. Therefore, manufacturing test methods used for reconfigurable devices is mostly based on configuring the device with a known set of applications and applying test patterns for each. The same approach can be suggested for manufacturing test of nanoscale devices based on the method proposed in this article. Also, application-dependent test methods are proposed for reconfigurable devices. These techniques are based on testing a possibly defective FPGA for a specific application. The same approach is applicable to reconfigurable crossbar-based devices through the SCT method discussed here.

## 7. CONCLUSION

A new approach to testing nanoscale devices is proposed in this article. This approach, called simultaneous configuration and test (SCT), is based on exhaustive testing of the functions of an application while they are being configured into the nanoscale blocks. It eliminates the requirement of storing the locations of all defects in a large defect map. The proposed method is architecture independent and applicable to all nanodevice architectures having rich interconnect resources. Probabilistic analyses on the number of cycles required to accomplish the SCT procedure in comparison to another BIST process demonstrate the time effectiveness of the proposed method. Simulation results confirm the efficiency of SCT for high-density and high defect-rate nanoscale devices, as compared to previously proposed test methods whose purpose is to locate all the defects and create a defect map.

## REFERENCES


BACHTOLD, A., HADLEY, P., NAKANISHI, T., AND DEKKER, C. 2001. Logic circuits with carbon nanotube transistors. *Sci. 294*, 1317–1320.

BETZ, V., ROSE, J., AND MARQUARDT, A. 1999. *Architecture and CAD for Deep Submicron FPGAs*. Kluwer, Norwell, MA.

BROWN, J. G. AND BLANTON, R. D. 2004. CAEN-BIST: Testing the Nanofabrics. in Proc. *Int. Test Conf. (ITC'04)*, pp. 462–471.

CHEN, Y., OHLBERG, D. A. A., LI, X., STEWART, D. R., WILLIAMS, R. S., JEPPESEN, J. O., NIELSEN, K. A., STODDART, J. F., OLYNICK, D. L., AND ANDERSON, E. 2003a. Nanoscale molecular switch devices fabricated by imprint lithography. *Appl. Phys. Lett. 82*, 10, 1610–1612.

CHEN, Y., JUNG, G., OHLBERG, D., LI, X., STEWART, D., JEPPESEN, J., NIELSEN, K., STODDART, J., AND WILLIAMS, R. 2003b. Nanoscale molecular-switch crossbar circuits. *Nanotechnol. Inst. Phys. 14*, 462–468.

CHOU, S. Y., KRAUSS, P. R., ZHANG, W., GUO, L., AND ZHUANG, L. 1997. Sub-10 nm imprint lithography and applications. *J. Vac. Sci. Technol. B, Microelectron. Process. Phenom.*, *15*, 6, 2897–2904.

CUI, Y., LAUHON, L. J., GUDIKSEN, M. S., WANG, J., AND LIEBER, C. M. 2001. Diameter-Controlled synthesis of single crystal silicon nanowires. *Appl. Phys. Lett.*, *78*, 15, 2214–2216.

CULBERTSON, B., AMERSON, R., CARTER, R., KUEKES, P., AND SNIDER, G. 1997. Defect tolerance on the Teramac custom computer. In *Proceedings of the IEEE Symposium on FPGA's for Custom Computing Machines (FCCM)*, 116–123.

DeHon, A., Lincoln, P., and Savage, J. E. 2003. Stochastic assembly of sublithographic nanoscale interfaces. *IEEE Trans. Nanotechnol. 2*, 3, 165–174.

DeHon, A. and Naeimi, H. 2005. Seven strategies for tolerating highly defective fabrication. *IEEE Des. Test Comput. 22*, 4, 306–315.

DeHon, A., Goldstein, S. C., Kuekes, P. J., and Lincoln, P. 2005. Nonphotolithographic nanoscale memory density prospects. *IEEE Trans. Nanotechnol. 4*, 2.

DeHon, A. and Wilson, M. J. 2004. Nanowire-Based sublithographic programmable logic arrays. In *International Symposium on Field Programmable Gate Arrays (FPGA)*.

Dekker, C. 1999. Carbon nanotubes as molecular quantum wires. *Phys. Today*, 22–28.

Goldstein, S. C. and Budiu, M. 2001. NanoFabric: Spatial computing using molecular electronics. In *Proceedings of the International Symposium on Computer Architecture*, 178–189.

Harris, I. and Tessier, R. 2002. Testing and diagnosis of interconnect faults in cluster-based FPGA architecture. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst. 21*, 11, 1337–1343.

Heath, J. R. and Ratner, M. A. 2003. Molecular electronics. *Phys. Today*, 43–49.

Huang, Y., Duan, X., Wei, Q., and Lieber, C. M. 2001. Directed assembly of one-dimensional nanostructures into functional networks. *Sci. 291*, 630–633.

Javey, A., Guo, J., Wang, Q., Lundstrom, M., and Dai, H. 2003. Ballistic carbon nanotube field-effect transistors. *Nature 424*, 654–657.

Kuekes, P. J. and Williams, R. S. 2000. Demultiplexer for a molecular wire crossbar network. U.S. patent number 6256767.

Mishra, M. and Goldstein, S. C. 2003. Defect tolerance at the end of roadmap. In *Proceedings of the International Test Conference (ITC)*.

Morales, A. M. and Lieber, C. M. 1998. A laser ablation method for synthesis of crystalline semiconductor nanowires. *Sci. 279*, 208–211.

Rad, R. M. and Tehranipoor, M. 2006. A new hybrid FPGA with nanoscale clusters and CMOS routing. In *Proceedings of the Design Automation Conference (DAC)*. to appear.

Rad, R. M. and Tehranipoor, M. 2006b. Fine-Grained island style architecture for molecular electronic devices. submitted to *International Symposium on Field Programmable Gate Arrays (FPGA)*.

Stan, M. R., Franzon, P. D., Goldestein, S. C., Lach, J. C., and Ziegler, M. M. 2003. Molecular electronics: From devices and interconnect to circuit and architecture. *Proc. IEEE*, 1940–1957.

Stroud, C., Konala, C., Chen, P., and Abramovici, M. 1996. Built-In self-test of logic blocks in FPGAs (finally, a free lunch: BIST without overhead!). In *proceedings of the 14th VLSI Test Symposium*, 387–392.

Strukov, D. B. and Likharev, K. K. 2005. CMOL FPGA: A reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. *Nanotechnol. Inst. Phys. 16*, 888–900.

Tahoori, M. 2004. Application-Dependent diagnosis of FPGAs. In *proceedings of the International Test Conference (ITC)*, 645–654.

Tehranipoor, M. 2005. Defect tolerance for molecular electronics-based nanofabrics using built-in self-test procedure. *In Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, 305–313.

Wang, Z. and Chakrabarty, K. 2005. Using built-in self-test and adaptive recovery for defect tolerance in molecular electronics-based nanofabrics. In *Proceedings of the International Test Conference (ITC)*. to appear.

Whang, D., Jin, S., and Lieber, C. M. 2003. Nanolithography using hierarchically assembled nanowire masks. *Nanolett. 3*, 7, 951–954.

Wind, S.J., Appenzeller, J., Martel, R., Derycke, V., and Avouris, P.H. 2002. Vertical scaling of carbon nanotube field-effect transistors using top gate electrodes. *Appl. Phys. Lett. 80*, 20, 3817–3819.

Received September 2006; revised May 2007; accepted June 2007