SCT: An Approach For Testing and Configuring Nanoscale Devices

Reza M.P. Rad and Mohammad Tehranipoor
Department of CSEE
University of Maryland Baltimore County
{reza2,tehrani}@umbc.edu

ABSTRACT

Molecular electronics-based devices are assumed to include at least 10^{10} gate-equivalents/cm^2 and defect densities as high as 10%; novel test strategies are necessary to efficiently test and diagnose these nanoscale devices. Configuration time, test time and defect map size are among the major challenges for these new devices. In this paper, we propose a new approach that simultaneously configures and tests nano devices. A new built-in self-test (BIST) scheme for testing and defect tolerance of nanoscale devices is proposed. The proposed procedure is based on testing reconfigurable nanoblocks at the time of implementing a function of a desired application on that block. This simultaneous configuration and test (SCT) procedure considerably reduces the test and configuration time. It also eliminates the need for storing the location of the defects in the defect map on/off-chip. The presented probabilistic analyses results show the effectiveness of this process in terms of test and configuration time for architectures with rich interconnect resources.

I. INTRODUCTION

Emerging technologies in developing nanoscale devices show horizons of a new era in digital circuit design. 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 can provide diode-logic circuits [1]. Many of these molecular switches are programmable and can save their state of connection or disconnection, hence it is possible to create configurable circuits using these switches. Carbon nanotubes (NTs) are synthesized with few nanometers in diameter and micrometers in length [2]. Also nanowires (NWs) with diameters as small as 3 nanometer and lengths of few hundred micrometers are reported in [3] and [4]. Several experiments to arrange nanowires in array structures are reported in [5] and [6]. Directed self-assembly and self-alignment techniques are mainly used to create these structures. *Nano-Imprint* lithography [7] is progressing to enable us with more precise designs and more flexibility in designing non-regular structures in nanoscale regime.

Different architectures are suggested in literature based on available nanoscale device and assembly technologies. Goldstein *et. al.* in [8] suggested an array architecture for nanoscale circuits. This design is an island-style architecture in which clusters are interconnected in an array structure. The island style architecture provides rich interconnect between clusters that improves flexibility of the fabric. DeHon and Wilson in [9] presented a detailed PLA-based FPGA architecture and used a technique, called modulated doping of nanowires, to create an interface for accessing nanowires through CMOS wires. Strukov and Likharev in [10] suggested a new 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.

Testability, defect tolerance, efficient access mechanism from CMOS support circuits to easily provide configuration and input/output signals, and also rich interconnect are the main challenges in design and test of nanoscale devices. Because of the high defect densities (up to 10%) in assembly and device technology, approximately all fabricated circuits using these technologies will have a handful of defects. Therefore, it is necessary that these circuits must be highly testable and an efficient defect tolerance scheme must be designed within them. The reconfigurability which is usually inherited to these circuits from their basic components, i.e. reconfigurable molecular switches, can provide a static fault tolerance for these devices. It has been shown in *Teramac* custom computer that reconfigurable architectures can tolerate high defect densities [11].

As commonly proposed in literature, a defect detection process must be performed and location of all defects in the device must be detected and stored as a defect map and in configuration phase, defective components should be avoided [12][13]. Several techniques have been proposed to extract the location of faulty blocks in nanoscale architectures [14][15][22].

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, especially in devices with high densities like nanoscale circuits. Even if effective methods could be found to determine the exact location of all defects in such architectures, storing all these information will need very large memories. Note that we cannot use the on-chip nanoscale resources as memory to store the defect map since they are unreliable. On the other hand, using on-chip CMOS scale memory for defect map will result in considerable area overhead. Therefore, it will not be practical to store this large defect map on-chip. It will not also be practical to ship the defect map of each chip along with it to the customer.

One possible solution is that customer's programming tool be able to perform both configuration and test. That means the tool should first apply the tests and extract the defect map, then configure the device for a specific application. Implementing an on-chip BIST circuit or using an on-chip microprocessor will significantly speed up this process but it will still be a very time consuming process to find the location of all defects in a chip with extremely large number of blocks. This process should probably be repeated every time a chip is reconfigured because of the existence of aging defects in nanoscale devices and also due to very high memory requirements for storing the defect map for all chips (different chips will have different defect maps). As above 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 [16] and for nano devices because the defect rate of nano devices is significantly higher than that of CMOS FPGAs. 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 implement the application completely and then apply the test, because the probability of having a fault-free configured application in this case will be very low.

One of the approaches suggested for FPGA BIST [18] is to configure a number of blocks of a FPGA as test pattern generators (TPGs) and some other blocks as response analyzers (RAs). The remaining configurable blocks can be selected as block under test (BUT) to receive the test patterns from TPGs and send the outputs to RAs. The BUT should be configured with a set of appropriate circuits and each time TPGs should provide tests for that specific function implemented on BUT and test it [18].

A. *Contribution and Paper Organization*

In this paper we propose a technique that simultaneously configures and tests a nano device. The application is first divided into smaller functions f_i, where $f_i \in \{f_1, f_2, ..., f_T\}$, implementable on nanoblocks. Then the function f_i is configured in a nanoblock and is tested using a BIST procedure. If configured f_i is fault-free, then the technique configures another block with function f_j. If it is faulty, f_i will be configured into another block and tested. The proposed approach is very efficient in terms of the overall time to configure and test the circuit and also 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 while the function f_i configured into it will operate without fault. This method is architecture independent and can be applied to all nanoscale architectures with high defect rate conditions having rich interconnect resources.

Section II presents our proposed SCT technique. In Section III, interconnect requirements of the proposed method are briefly discussed. Section IV presents a probabilistic analysis to calculate average amount of time required to configure and test a circuit on a nanoscale architecture using the proposed method. The results of time analysis are compared with the approximations made for a previously proposed test method [22]. The paper ends with conclusions in Section V.

II. Simultaneous Configuration and Test (SCT) Method

As mentioned above, locating all defects inside a nano device with very high density (10^{10} gate-equivalents/cm^2) and high defect density (up to 10%) will become a challenging and time consuming task. In this paper a method is proposed for testing and configuring circuits that can be used to avoid the time consuming process of locating all defects. We assume that the architecture have rich interconnect resources and is able to provide efficient access to its logic blocks through its input/output interfaces. Such architectures are presented in literature [9][13][17].

The proposed method, in this paper, is conceptually similar to those proposed for FPGAs [18], but we consider TPG and RA to be components of BIST circuit and to be implemented in CMOS scale to provide tests and analyze the responses. Detailed architecture of the proposed BIST circuit for testing configured blocks is described in this section and interconnect testing issues will be discussed in next section. The main difference between our method and the FPGA BIST method suggested in [18] is that in our method the goal of testing is not to confirm the correct functionality of a BUT for all functions. Here, the goal is to make sure that each function (f_i) of an application configured into a block is working correctly. So, the test patterns should be applied for testing that function only.

In SCT method instead of testing all resources of a reconfigurable architecture to locate all the defects in it, each block of the architecture can be tested for the specific function of a circuit, i.e. f_i, after f_i is configured into a block of the device. The applied test here just checks the correct functionality of the configured function (f_i), rather than diagnosing all defects of the block. So, there might be defects in molecular switches or nanowires of the block but as long as those 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 of that block. So, if the defective components of the block are not used during configuring f_i into that block, then the function can operate without fault. Therefore, the defects of the block are tolerated.

In SCT procedure, the application is divided into m-input functions and each time one of these functions (f_i) should be configured into a block of the device and the input and output lines from the BIST circuit to f_i must be configured. Also, the same function f_i should be configured into the LUT of the BIST circuit (see Figure 1). Next, the BIST circuit can simply apply 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. The process of selecting a function, mapping it to a block of the device and creating connections between that function and the BIST circuit and testing the function will be repeated for all functions of the application. If a function fails the test then we should configure another block with that function and the test process should be repeated.

Note that methods and tools for configuring nanoscale reconfigurable architectures will be similar to those used for FPGAs [19] but some modifications may be required due to architectural differences.

Figure 1 shows the proposed BIST scheme. The BIST circuit is composed of an m-bit counter, an m-input LUT and a comparator, resulting in low BIST area overhead. BIST circuit is assumed to be implemented in reliable CMOS scale.

Figure 2 shows the SCT procedure. In this procedure, a function f_i of the desired application is selected (line #10) from the

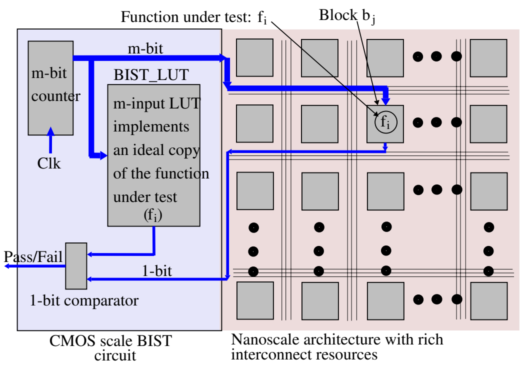
Function under test: f_i
Block b_j
m-bit
m-bit counter
BIST_LUT
m-input LUT implements an ideal copy of the function under test (f_i)
Clk
f_i
Pass/Fail
1-bit
1-bit comparator
CMOS scale BIST circuit
Nanoscale architecture with rich interconnect resources

Fig. 1. The proposed CMOS BIST circuit used to test nano devices.

```
01: \\ Define:
02: \\ Set of available blocks: B={$b_1$, $b_2$, .., $b_M$} ;
03: \\ Set of functions of the application: F={$f_1$, $f_2$, .., $f_T$} ;
04: \\ Set of (defective block, # of defects) :
05: \\ DB={($b_1$,$n_1$), ($b_2$,$n_2$), .., ($b_x$,$n_x$)} ;
06: \\ Set of discarded blocks: DiB ;
07: \\ # iterations a block should be detected as
08: \\ defective before it is discarded: Discard_Threshold ;

09: while ($F \neq \{\}$) {
10: Choose ($f_i$ from F) ; F = F - {$f_i$} ;
    Config (BIST_LUT with $f_i$);
11: if ($B \neq \{\}$) {
12:    while ($B \neq \{\}$) {
13:          Choose ($b_j$ from B); B = B - {$b_j$} ;
14:          Config ($b_j$ with $f_i$) ; ConfigRoutes ($b_j$ to BIST);
15:          if (BIST ($f_i$,$b_j$) = fail )
16:          DB = DB + {($b_j$,1)} ;
17:          else Break ; }
18:          }
19: else if ($B = \{\}$) {
20:       while ($DB \neq \{\}$) {
21:       Choose (($b_k$,$n_k$) from DB) ;
22:       Config ($b_k$ with $f_i$) ; ConfigRoutes ($b_k$ to BIST);
23:       if ( BIST ($f_i$,$b_k$)=fail ) {
24:          increment ($n_k$) ;
25:          if ($n_k \geq$ Discard_Threshold)
26:          DB = DB - {($b_k$,$n_k$)} ; DiB = DiB + $b_k$ ;
27:          }
28:       else Break ; }
29:          }
30: if ($B = \{\}$ and $DB = \{\}$) {
31:       Application is not implementable on the device.}
32:    }
```

Fig. 2. The SCT procedure.

list of functions (F) and configured on the LUT of the BIST circuit (BIST_LUT, line #10). If there are available blocks in the block list (B), one of them, e.g. b_j, will be selected and removed from the list of available blocks (line #13). Then function f_i will be mapped into this block (b_j) and the wires between this block and BIST circuit are configured (line #14). If the test fails (line #15), the block will be sent to the set of defective blocks, i.e. DB (line #16). This will be repeated until either all functions are configured into the blocks of the device ($F = \{\}$) and the procedure is successfully finished or there are still some functions of the application left unmapped and the set of available blocks (B) becomes empty. In this case all the blocks of the device have been tried once, either they have been successfully used to implement one of the functions or they have shown defective behavior and were sent to the set of defective blocks

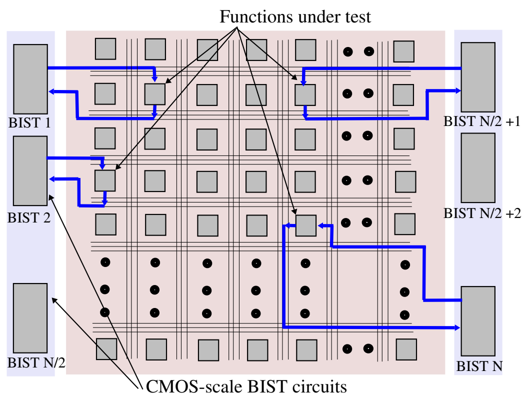

Fig. 3. Parallel use of multiple BIST circuits for testing nano devices.

(DB). If a block shows faulty behavior for one function, it does not mean that it will necessarily be faulty for every function because different functions will use different subsets of switches and nanowires of a block. Therefore, if there are some functions of the application left, defective blocks of the device can be tried (line #21). However, if a block is tried for a number of different functions and for all of them showed a faulty behavior, i.e. $n_k \geq Discard\_Threshold$ (where n_k is the number of times block b_k was used to implement a function and the result was faulty), then the block should be discarded and sent to discarded block list, i.e. DiB (line #26). After trying all the defective blocks of the device if still there are some functions of the application left, then the application is identified to be not implementable on the device.

As seen in the figure, two selections should be made during these steps. First, one of the functions of the circuit should be selected from the set of all functions, i.e. F (line #10). Then, an appropriate block of the device should be selected and configured with the selected function (line #13 or #21). Block selection is a decision that programming device should make based on the available blocks in the architecture, communications between the function being implemented (f_i) and other functions of the application and timing requirements of the circuit.

Low area overhead of this BIST structure provides the opportunity of parallel implementation of these testers on the chip so that at any time more than one function can be implemented and tested on the device as shown in Figure 3. When multiple BIST circuits are implemented, test time will be reduced. In this case, more than one function and more than one block of the device should be selected at any time. Proper methods based on heuristics can be used for these selections.

The interconnections between functions of the circuit also need to be configured and tested. This will be further discussed in the next section.

III. Interconnects and Their Test Process

A. CMOS Scale Interconnects

To complete the configuration of the application, interconnections between the functions should be configured and tested. The SCT procedure explained earlier does not consider testing the interconnections between the functions. It uses interconnect resources of the device to provide an access for BIST to send test patterns and receive test responses to and from a function.

If the interconnects of the nano architecture are fabricated in CMOS scale, the problem of testing them can be removed (due to their reliability) or a CMOS test strategy such as those proposed in [20] can be used to ensure the fault-free status of the interconnects. In case of detecting a fault on CMOS wires, we can discard the chip and this will not be very costly because of the low defect density of CMOS technology. Since the interconnects are assumed to be in CMOS scale and defect-free, the connections between functions of the circuit can be made after the SCT procedure to complete the configuration of the circuit.

CMOS interconnections in nanoscale architectures are used in the architectures proposed in [10], [17] and [21]. The reliability and low defect density of CMOS wires provides considerable simplicity in test and configuration process of nanoscale architectures. Such nanoscale architectures will not be as area efficient as those architectures with nanowires used as interconnects. As discussed in [10], [17] and [21], the accessibility, testability and configurability provided by reliable CMOS scale interconnects are necessary for nanoscale architectures to be practical.

The SCT procedure proposed in previous section and also time analyses presented in the next section are based on the assumption of a nanoscale architecture with CMOS interconnections between the blocks.

B. Nanoscale Interconnects

If the interconnect system used in the nanoscale architecture is based on nanowires, then it has high defect rates. One possible solution to make the results of the SCT procedure efficient in this situation is to perform a pre-processing step to test the interconnects. In this pre-processing step, interconnects of the architecture must be tested and fault-free connections should be determined. Since the structure of nanoscale architectures is predicted to be very regular, testing the interconnects can be accomplished 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.

Another suggestion is that, when an implemented function on a block of a device fails during the test process, there is a possibility of fault both in the function and in the interconnects used for communication between the function and the BIST circuit. The programming tool should decide whether to change the block which was used for implementing the function or just change the interconnects used between the function and the BIST circuit and test the same block again (using the new interconnects). This decision should be made based on the configuration time requirements and defect probability of blocks and interconnects. Based on these, appropriate heuristic methods can be developed to find a fault-free implementation of the function in an optimum amount of time. Also, methods need to be proposed to test nanoscale interconnects created between the functions. When compared to using CMOS scale interconnects, the SCT procedure will require more time in case of using nanoscale interconnects.

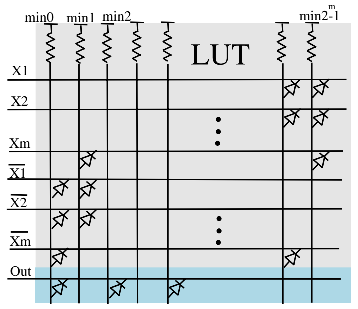

Fig. 4. Implementation of an m-input function on a $(2m+1) \times 2^m$ crossbar configured as a lookup table. Function $F(x_1, x_2, .., x_m) = \sum Minterms(0,2,4)$ is implemented as an example.

IV. Analysis Of The SCT Method

Since no nanodevice is available to perform any real implementation, in this section, a probabilistic analysis is presented to show the timing requirements of the proposed process. We first calculate the probability of occurring a fault in a function f_i configured into a block of the device. Based on this probability we calculate the average number of blocks that must be configured to finally find a fault-free implementation for the function. Hence, we can calculate the required number of clock cycles for configuring and testing function f_i. Finally, for an application with T functions ($\{f_1, f_2, ..., f_T\}$ 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 [22].

To keep the analysis simple, CMOS scale interconnects are assumed for the architecture. We target an application that can be partitioned into T small functions each with m inputs, i.e. functions $\{f_1, f_2, ..., f_T\}$. As Figure 4 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). Note that since diode logic cannot provide signal inversion, 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 the vertical nanowires with first $2m$ horizontal nanowires are configured to provide the minterms. The switches on the junctions of vertical nanowires with last horizontal nanowire (*Out*) are configured to provide the sum of the minterms specified by the function. So, the first $2m$ horizontal wires create the *AND* terms (minterms) and the last horizontal wire create the *OR* term.

First, we calculate the average probability of having a fault in a function configured in a LUT. Let's assume that P_o is the probability of an open fault in a molecular switch, P_c shows the probability of a closed fault in a molecular switch and P_w denotes the probability of a fault caused by a defect in one of the nanowires. The probability of an implemented minterm to be faulty is given by:

$$P_{(faulty\ minterm)} = 1 - P_{(fault-free\ minterm)}$$
$$= 1 - (1-p_c)^m (1-p_o)^{(m+1)} (1-p_w)^{(2m+2)}$$

The probability of an m-input function with x minterms to be faulty is obtained from:

$$P_{(faulty\ function)} = 1 - P_{(fault-free\ function)}$$

$$= 1-[(1-p_c)^m(1-p_o)^{(m+1)}(1-p_w)]^x(1-p_w)^{(2m+1)}$$

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

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

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

N_r can also be defined as the average number of blocks that should be configured to finally 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 the Figure 4, there are $(m+1)$ switches for each minterm to be configured, so the average number of switches to be configured for a function (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 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 should be applied to each of the configured functions 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:

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

First term in the above equation is the number of cycles required for testing the configured functions (each time a function is configured, 2^m test patterns should be applied to it). Second term is the number of cycles required for configuring the function. For a circuit with T m-input functions, the time of performing SCT procedure assuming N parallel BIST circuits, as shown in Figure 3, can be calculated as:

$$\#\,Cycles\ (SCT) = N_r \times (\frac{T}{N}) \times (2^m + \frac{N_{sf}\cdot N}{N_{cs}}) \quad (1)$$

In this equation, first term of the sum is the 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 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 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 [22]. The number of configurations required for testing a $K \times K$ nanoblock of a nanofabric is calculated to be $4K+6$ and the number of configurations required for testing a $K \times K$ switchblock of a nanofabric is estimated to be $4K^2$ [22].

To make a fair comparison, here we use the same assumptions and parameters used in the analysis of our SCT method. We assumed using CMOS scale interconnects for the architecture hence, the required number of cycles for test and configuration of the interconnects were omitted from the calculations. Therefore, to keep the same conditions for [22], 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. Based on our analysis, the number of cycles required for BIST method proposed in [22] based on these assumptions can be calculated as:

$$\#\,Cycles\ (BIST\ [22]) =$$
$$\frac{T}{(1-p_w)^{2K}(1-p_c)^{K^2}(1-p_o)^{K^2}} \times \frac{4K+6}{N_{cs}} \quad (2)$$

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

Once 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 map. Also, as discussed in [22], applying the test patterns in that BIST process (and 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 determined as faulty during these test methods. That means recovery, i.e. 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 considerable number of reconfigurations that will significantly increase the test time. In analysis presented above 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 [22].

The number of cycles for SCT procedure and for BIST process proposed in [22] are compared for different defect probabilities and design parameters and the results are presented in Figures 5 and 6. In each of these figures, constant values are assigned to m, T, N_{cs} and P_{si}. Different values are assigned to N, i.e. the number of parallel working BIST circuits, and to the fault probabilities of the molecular switches and nanowires. As the results demonstrate, in most cases the required cycles for our SCT procedure is considerably less than cycles of BIST method of [22].

As Figure 7 shows, increasing N, i.e. 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 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 configuration time will become the dominant part of the SCT procedure and this part cannot be reduced through adding the number of parallel BIST circuits (N). Adding more parallel configuration circuitry, i.e. increasing N_{cs} from 5 to 20 as shown in the figure, will reduce the configuration time and this in turn reduces the SCT time. Area overhead resulted by

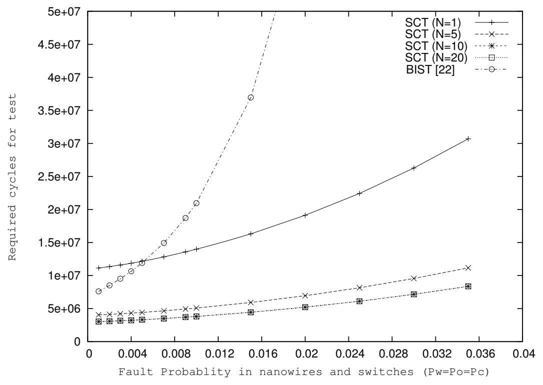

Fig. 5. Number of cycles required for testing a nano device using SCT method and BIST method proposed in [22] when $m = 3, T = 1000000, P_{si} = 0.999, N_{cs} = 5$.

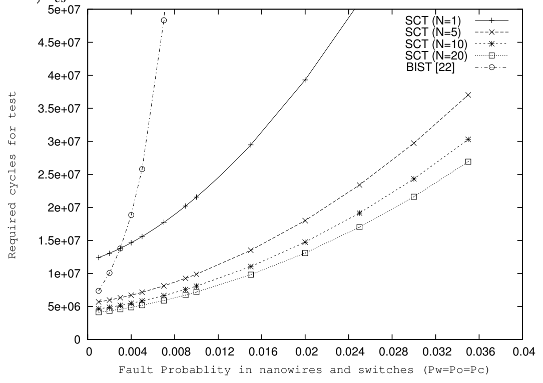

Fig. 6. Number of cycles required for testing a nano device using SCT method and BIST method proposed in [22] when $m = 4, T = 500000, P_{si} = 0.999, N_{cs} = 5$.

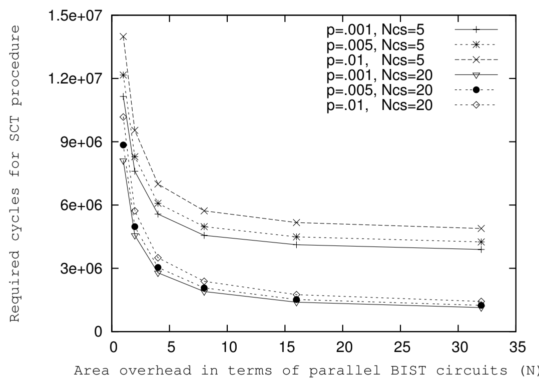

Fig. 7. SCT procedure time versus N, i.e. the number of parallel BIST circuits, which is a measure of BIST area overhead ($m = 3, T = 1000000, P_{si} = 0.999$).

adding the parallel BIST circuits will be still small comparing to the area required to store the defect map as previously proposed by BIST methods such as [14][15][22].

V. Conclusion

A new approach to testing nanoscale devices was proposed in this paper. This approach, called simultaneous configuration and test (SCT), is an architecture independent method. The method is based on exhaustive testing of the functions of an application while they are being configured into the nanoscale device. This method removes the requirement of storing the location of all defects in a large defect map. 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. The results confirm the efficiency of SCT for high density and high defect rate nanoscale devices compared to previously proposed test methods whose purpose is to locate all the defects and create a defect map.

References

[1] Y. Chen, D.A.A. Ohlberg, X. Li, D.R. Stewart, R.S. Williams, J.O. Jeppesen, K.A. Nielsen, J.F. Stoddart, D.L. Olynick and E. Anderson, "Nanoscale molecular switch devices fabricated by imprint lithography," *Appl. Phys. Lett.*,vol. 82, no. 10, pp. 1610-1612, 2003.

[2] C. Dekker, "Carbon nanotubes as molecular quantum wires," *Phys.Today*, pp. 22-28, May 1999.

[3] Y. Cui, L. J. Lauhon, M. S. Gudiksen, J. Wang, and C. M. Lieber, "Diameter-controlled synthesis of single crystal silicon nanowires," *Appl. Phys. Lett.*, vol. 78, no. 15, pp. 2214-2216, 2001.

[4] A. M. Morales and C. M. Lieber, "A laser ablation method for synthesis of crystalline semiconductor nanowires," *Science,*,vol. 279, pp. 208-211,1998.

[5] Y. Huang, X. Duan, Q. Wei, and C. M. Lieber, "Directed assembly of one-dimensional nanostructures into functional networks," *Science*, vol. 291, pp. 630-633, Jan, 2001.

[6] D. Whang, S. Jin, and C. M. Lieber, "Nanolithography using hierarchically assembled nanowire masks," *Nanoletters*, vol. 3, no. 7, pp.951-954, July 2003.

[7] S. Y. Chou, P. R. Krauss,W. Zhang, L. Guo, and L. Zhuang, "Sub-10 nm imprint lithography and applications," *J. Vac. Sci. Technol. B, Microelectron. Process. Phenom.*, vol. 15, no. 6, pp. 2897-2904, Nov-Dec, 1997.

[8] S. C. Goldstein and M. Budiu, "NanoFabric: Spatial Computing using Molecular Electronics," in Proc. *Int. Symp. on Computer Architecture*, pp. 178-189, 2001.

[9] A. DeHon, M. J. Wilson, "Nanowire-based Sublithographic Programmable Logic Arrays," *Int. Symp. on Field Programmable Gate Arrays (FPGA'04)*, 2004.

[10] Dmitri B Strukov and Konstantin K Likarev, "CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices," *Nanotechnology, Inst. of Phys.*, vol. 16, pp. 888-900, 2005.

[11] B. Culbertson, R. Amerson, R. Carter, P. Kuekes and G. Snider, "Defect Tolerance on the Teramac Custom Computer," in Proc. *IEEE Symp. on FPGA's for Custom Computing Machines (FCCM'97)*, pp. 116-123, 1997.

[12] M. Mishra and S.C. Goldstein, "Defect Tolerance at the End of Roadmap," in Proc. *Int. Test Conf. (ITC'03)*, 2003.

[13] A. DeHon and H. Naeimi, " Seven Strategies for Tolerating Highly Defective Fabrication," *IEEE Design & Test of Computers*, vol. 22, Issue 4, pp. 306-315, 2005.

[14] J. G. Brown and R. D. S. Blanton, "CAEN-BIST: Testing the Nanofabrics," in Proc. *Int. Test Conf. (ITC'04)*, pp. 462-471, 2004.

[15] M. Tehranipoor, " Defect Tolerance for Molecular Electronics-Based NanoFabrics Using Built-In Self-Test Procedure," in Proc. *Int. Symp. on Defect and Fault Tolerance in VLSI Systems (DFT'05)*, pp. 305-313, 2005.

[16] M. Tahoori, "Application-Dependent Diagnosis of FPGAs," in Proc. *Int. Test Conf. (ITC'04)*, pp. 645-654, 2004.

[17] R. M. Rad and M. Tehranipoor, "Fine-Grained Island Style Architecture for Molecular Electronic Devices," submitted to *Int. Symp. on Field Programmable Gate Arrays (FPGA'06)*, 2006.

[18] 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. *14th VLSI Test Symposium*, pp. 387-392, 1996.

[19] V. Betz, J. Rose and A. Marquardt, "Architecture and CAD for Deep Submicron FPGAs," Norwell, MA: Kluwer, 1999.

[20] I. Harris and R. Tessier, "Testing and Diagnosis of Interconnect Faults in Cluster-Based FPGA Architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 11, pp. 1337-1343, Nov. 2002.

[21] A. DeHon, S. C. Goldstein, P. J. Kuekes and P. Lincoln, "Nonphotolithographic Nanoscale Memory Density Prospects," *IEEE Transactions on Nanotechnology*, vol. 4, No. 2, March 2005.

[22] Z. Wang and K. Chakrabarty, "Using Built-In Self-Test and Adaptive Recovery for Defect Tolerance in Molecular Electronics-Based Nanofabrics," to appear in *Int. Test Conf. (ITC'05)*, 2005.