Fall 2011

Optimized testing and logic mapping methodology for CAEN-based nano-circuits

Sambhav Kundaikar

Department:

OPTIMIZED TESTING AND LOGIC MAPPING METHODOLOGY FOR CAEN-BASED NANO-CIRCUITS

By

SAMBHAV DILIP KUNDAIKAR

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER ENGINEERING

2011

Approved by:

Maciej Zawodniok, Advisor
Minsu Choi
Daryl Beetner

PUBLICATION THESIS OPTION

This thesis is composed of the following two papers which were reformatted in the style used by the university.

The first paper presented in pages 06-50 titled "OPTIMIZED TESTING TECHNIQUE FOR DEFECT TOLERANCE IN CAEN-BASED NANOFABRIC SYSTEMS" has been submitted to the IEEE TRANSACTIONS ON NANOTECHNOLOGY, 2011.

The second paper presented in pages 51-72 titled "INTRODUCTION TO A NOVEL DEFECT-AWARE LOGIC MAPPING APPROACH FOR CROSSBAR-BASED NANOFABRICS" is intended for submission to INTEGRATION, THE VLSI JOURNAL.

A condensed version of Paper I titled "OPTIMIZED BUILT-IN SELF-TEST TECHNIQUE FOR CAEN-BASED NANOFABRIC SYSTEMS" has been accepted at the IEEE NANO 2011, to be held in Portland-Oregon, August 15-19, 2011.

ABSTRACT

Nanotechnology has been shown to have the potential to replace the existing CMOS technology in the race to maintain the Moore's Law increases in IC complexity. This work considers the Chemically Assembled Electronic Nanotechnology (CAEN), which fabricates nanofabric using a low cost self-assembly and self-alignment chemical process and provides very high density. However, the main disadvantage of this technology is the inherently high defect rate that hinders the efforts to commercialize such systems. Existing testing and design techniques, for example for FPGAs, are ill suited since they typically assume vary low defect rates.

This thesis is comprised of two papers. In the paper I, a novel testing methodology is proposed along with a new set of test patterns and configurations which test the entire nanofabric for stuck-at, bridging and cross-point faults. An optimization technique is described to reduce the number of test configurations and testing time. A customization technique is also discussed to further increase the yield of the nanofabric when the desired functionality is known. Once the nanofabric has been tested and the faulty areas in the chip have been identified, the next step is to map the logic onto the nanofabric. The paper II discusses a new logic mapping approach for nanofabrics which have been tested to successfully implement AND/OR configurations of various logic functions. This approach uses the information provided by the testing technique from the paper I to simplify the logic mapping process. It used standard implementations of nanoblocks as compared to the existing techniques, which require customized solutions.

ACKNOWLEDGMENTS

It gives me great pleasure to thank all the people who have supported me and made this thesis possible. I would like to thank Dr. Maciej Zawodniok for being a great advisor throughout my Master's program and it has been a pleasure working with him. He has been a continuous source of motivation and has helped me develop my skills.

I would like to express my sincere gratitude to my thesis committee members, Dr. Minsu Choi and Dr. Daryl Beetner for their co-operation. I would also like to thank Dr. Waleed Al-Assadi, Dr. Minsu Choi, Dr. Daryl Beetner and Dr. Ali Hurson who have taught me excellent courses during my Master's program.

I would also like to thank Ritu Bhatia, Veeresh Hongal and many more friends who have helped me continuously throughout my Degree program and provided me with a refreshing environment. Most importantly, I would like to thank my parents, Dilip Kundaikar and Surat Kundaikar whose continuous support made this degree possible.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION

1. INTRODUCTION

1.1. OBJECTIVE

The main objective of this work is to design a new and improved approach for testing and design of logic functions using nanofabric arrays. The goal was to maximize the nanofabric yield while keeping the testing time to a minimum. The proposed testing process and logic mapping approach are design to work in concert to achieve these goals.

1.2. BACKGROUND

In the last 40 years there has been an exponential increase in the number of transistors per unit area. This increase has been in accordance with Moore's Law that predicted the number of transistors that could be placed on the chip would double every two years. However, there are some challenges like leakage currents, process variation, costs and reliability issues that may result to the end of scaling. This poses a threat to the continuation of Moore's Law. Therefore, a new technology will need to replace CMOS one in the near future.

One of the technologies under intense investigation as a possible alternative to CMOS is Chemically-Assembled Electronic Nanotechnology (CAEN) [1][2]. It has the potential to achieve high density while being fabricated using a low-cost chemical synthesis processes. The CAEN uses self-assembly and self-alignment to construct electronic circuits out of nanometer-scale devices. CAEN systems also referred to as nanofabrics, can achieve a density of as high as 10^8 to 10^{10} gates per cm^2 by using

interconnected 2D arrays of nanowires. The 2D arrays, referred to as nanoblocks are the fundamental units of a nanofabric.

However, the main drawback of the nanofabric system is its high defect rate which could be as high as 10%. Such a high defect rate leads to low yield thus making the manufacturing costs prohibitively high. Therefore it is not economically feasible to discard a nanofabric if it is found to have defects. Defect tolerance is needed to make such nanofabrics commercially viable. Defect tolerance refers to the ability to detect and locate faulty elements on a chip, and then avoid these faults through reconfiguration. Such approach has potential to increase yield and reduced manufacturing cost. Therefore, new testing approaches and methods are required to diagnose defective sections of the nanofabric and then use this information to effectively map logic onto the nanofabric.

1.3. PROPOSED APPROACHES

In this thesis, a new testing methodology is discussed which aims at maximizing the yield from a nanofabric while minimizing the testing time. In comparison to the existing testing methods for nanofabric testing, this method is much more efficient in terms of yield, total testing time and the number of test patterns and test configurations required. An optimization scheme is described which can be used to eliminate redundant test configurations to improve yield. It is also possible to customize the existing test configurations to meet the user requirements, which if used effectively, can further reduce the testing time and maximize yield of the nanofabric.

Once the nanofabric has been tested, a defect map is created which identifies the faulty blocks on the chip. This defect map can be used while mapping logic onto the

nanofabric so that the faulty blocks could be avoided. The logic mapping process takes the defect map and the function to be implemented as the inputs. The fault-free blocks are configured accordingly and connected with each other in order to obtain the final output.

The traditional approach aims at designing an individually tailored realization of functions for each nanoblock. The programming typically differs due to the changing location of faults inside the nanoblock and particular function that is to be implemented. In contract, the proposed approach uses standardized gate configurations for AND/OR gates. This may result in small fraction of nanoblocks that will be considered faulty where a per-block customized function implementation might be able to recover it. However, such highly customized, existing approaches will lack flexibility during reconfiguration and incur a very high testing and design overhead since each nanoblock has to be analyzed and redesigned for all possible solutions. In contrast, the proposed approach utilizes standard predefined configurations, and hence this simplifies the mapping process. However, when accompanied by the proposed testing techniques will miss only small fraction of usable, partially defective nanoblock. Also, it does not require knowledge about the location of defects inside the nanoblock, nor has to consider it during logic design and mapping phase.

1.4. CONTRIBUTIONS

Paper I

- A novel testing technique for testing nanofabric arrays.
- An introduction of several new test configurations and patterns.

- Detailed analysis of fault coverage capabilities for the complete set of test configurations and their test patterns.
- An optimization technique for selecting the minimal test set and increasing utilization with reduced testing time, and a customization approach that further reduces testing time and increases nanofabric yield when the desired logical functionality is known.

Paper II

- A novel logic mapping approach for nanofabrics tested by the testing technique discussed in paper I.
- Study of the effects of various parameters such as the defect rate, location of defects, complexity of the function to be implemented, on the proposed mapping technique.

1.5. FUTURE WORK

The proposed testing methodology can be further extended to include extensive testing of switchblocks. The new proposed set of configurations and test patterns only test for nanoblocks faults, though switchblock faults can be detected in some cases. A new set of configurations could be developed such that the switchblocks are tested along with the nanoblocks. Also, a design of the comparator blocks is an important aspect of testing, and yet little literature is available on the topic. Paper I also discusses the recovery procedure, which can be further refined to allow arbitrary implementation of one or few gates within a nanoblock. The logic mapping technique could be modified/extended to incorporate these blocks since this would increase the overall utilization of the nanofabric. Moreover,

the current assumption was that only one pair of AND and OR gates with (k-1) inputs is implemented in each block. However, it is possible to implement several smaller gates within one nanoblock by making use of the unutilized cross-points.

PAPER

I. OPTIMIZED TESTING TECHNIQUE FOR DEFECT TOLERANCE IN CAEN-BASED NANOFABRIC SYSTEMS

Sambhav Kundaikar, Maciej Zawodniok

Electrical and Computer Engineering

Missouri University of Science and Technology

Rolla, MO 65401, USA

Email: sdk8v5@mail.mst.edu, mjzx9c@mst.edu

Abstract

Nanotechnology enables future advancements in integrated circuitry's miniaturization, energy and cost efficiency, and capabilities. However, a popular chemically assembled electronic nanotechnology (CAEN) has a high rate of defects that negates these benefits of the nanofabric. In order to address this challenge, a testing technique is proposed that maximizes the yield from a nanofabric while minimizing testing overhead. Moreover, traditional testing techniques, for example the ones employed in FPGA applications, assume low defect rate and fails to achieve high effectiveness when applied to testing nanofabrics. In this paper, a novel approach to testing nanofabric is proposed that includes new testing configurations, test-set optimization methodology, and design of customized configuration, which provide reduction in testing time while enhancing the utilization of the nanofabric. Part of the proposed scheme is a recovery procedure that further increases the utilization of nanoblocks at the expense of testing time. The proposed procedure tests all the components in parallel and identifies the defective

nanoblocks in a nanofabric. A defect map is generated to aid logic function implementation in a nanofabric. The proposed technique results in less number of test configurations compared to other proposed methods and a significant reduction in the test time.

1. INTRODUCTION

Moore's Law predicted that the number of transistors that could be placed on the chip would double every two years. The CMOS technology has been able to keep up with Moore's law as a result of scaling. However, CMOS technology today faces a number of challenges such as leakage currents, process variation, costs and reliability issues which may put an end to scaling. Therefore, new technologies are needed to replace CMOS in the future to continue the Moore's Law advancement.

Chemically-Assembled Electronic Nanotechnology (CAEN) has shown promising potential for the future of nano-scale design [1][2]. A basic building block of a nanofabric is nanoblock, which is an interconnected 2D array of nano-scale wires that can be electronically configured as logic networks, memory units, and signal-routing cells [3]. These nanofabric architectures are reconfigurable in nature and can achieve a density of 10^{10} to 10^{12} gates per cm^2, which is significantly higher than CMOS-based devices. However, due to the high defect rate in these nanofabrics, new testing strategies need to be devised to effectively test and diagnose the nanofabric within a reasonable time.

However, due to the low cost manufacturing process of self-assembly and self-alignment, the defect rate could be as high as 10%. Such high defect rate leads to low yield, which in turn results in high manufacturing costs. It is impractical to throw-away a fabricated chip once it is diagnosed to be defective. Thus defect tolerance is needed. The ability to identify and diagnose the faulty sites on a chip and develop techniques to avoid these faulty areas is known as Defect tolerance [3]-[9]. Also, the testing techniques needed for nanofabrics are complicated due to the large density, high defect rate and

large size of the nanoblocks. Hence traditional FPGA testing approaches do not apply to the nanofabric systems. A more distinct approach is required to address these challenges.

In this paper, a new testing methodology is discussed which configures the components as Block Under Tests (BUT) and Comparators (C). This method can handle high defect densities and large size nanofabrics. Our method uses a set of Test Architectures and BUT configurations to test the nanofabric. The design of each of the BUT configurations for the targeted faults is also explained. An external tester is used to apply test patterns to the BUTs. The nanoblocks are tested in parallel and hence the testing time does not depend on the size of the nanofabric. Further, an optimization technique is introduced which can increase both the testing speed and the utilization (yield) of the nanoblock.

The main contributions of this paper are: (a) a novel testing technique for testing nanofabric arrays, (b) an introduction of several new test configurations and patterns, (c) detailed analysis of fault coverage capabilities for the complete set of test configurations and their test patterns, (d) an optimization technique for selecting the minimal test set and increasing utilization with reduced testing time, and (e) customization approach that further reduces testing time and increases nanofabric yield when the desired logical functionality is known.

2. NANOFABRIC ARCHITECTURE

A nanofabric system has some similarities with a field-programmable gate-array (FPGA) [8], a regular 2D-array architecture and re-configurability. Reconfigurable devices are fault tolerant since faults can be identified and the functionality redesigned to avoid the corresponding defect map. Traditionally, the faulty blocks are masked during the configuration of the device and implementation. The remaining blocks are used, thus enabling to utilize partially defective chips. The same general methodology can be employed for nanofabric systems, though specific approach and testing scheme has to be designed to address high defect rate of the nanoblocks.

CAEN systems are fabricated using bottom-up manufacturing, where basic components such as nanowires are first obtained through chemical self-assembly, and then aligned to form a two-dimensional grid with configurable molecular switches at the junctions [1]. These molecular switches are configured to create useful circuits out of these grids [1].

Similar to FPGAs, the nanofabric is a regular 2D-mesh of interconnected fundamental units called nanoblocks, as shown in Fig. 1. A nanoblock can be programmed after fabrication to implement a certain logic function, while the switchblock can be configured to route signals between nanoblocks.

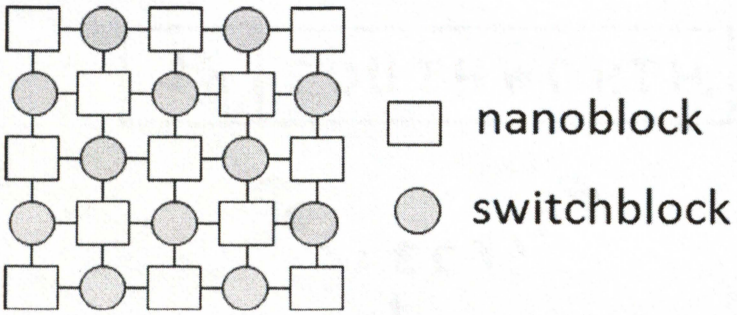


Fig. 1 Nanofabric architecture

2.1. NANOBLOCK

Figure 2 shows the schematic of a nanoblock. It is composed of three sections: (1) the molecular logic array, where the functionality of the block is located, (2) the latches, used for signal restoration and signal latching for sequential circuit implementation, and (3) the I/O area, used to connect the nanoblock to its neighbors through the switch block. The molecular logic array (MLA) portion of a nanoblock is composed of two orthogonal sets of wires. At each intersection of two wires lies a configurable molecular switch or a cross-point or a junction. The cross-points can be configured as "on" or "off" by applying a voltage potential across it. When configured to be "on", the cross-points act as diodes [1].

Figure 3 shows the implementation of an AND gate. If either A or B is at logic "0", the corresponding diode is forward-biased and turned on. The resistors are manufactured appropriately, i.e., resistors attached to VDD have smaller impedances than those attached to Gnd, such that the output vertical wire is pulled down to logic "0" [1].

Note that the resistance of nanowires and molecular switches is very low. Figure 3 also shows how an OR gate can be implemented for inputs X and Y.

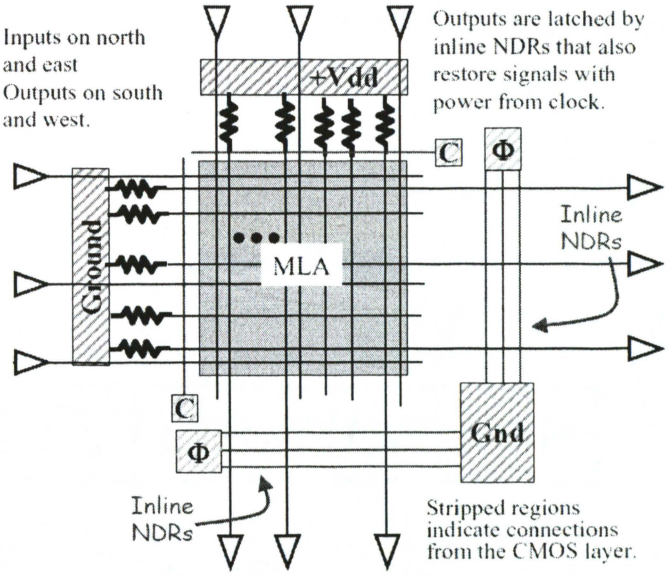


Fig. 2 Schematic of a nanoblock [1]

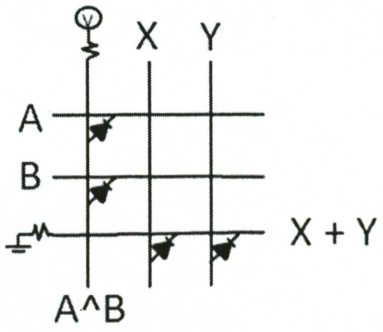


Fig. 3 AND/OR gate implementation

If the MLA portion of a nanoblock has k horizontal wires and k vertical wires, then the size of the nanoblock is referred to as kxk. The MLA implements Boolean functions using diode-resistor logic. The drawback of this logic style is that a signal is degraded whenever it passes a molecular switch. The molecular latch, constructed entirely from molecular-scale devices, is used to perform signal restoration using power from the clock to provide gain. The molecular latch also provides the properties of I/O isolation and noise immunity [14].

2.2. SWITCHBLOCK

A switchblock is shown in Fig. 4. It is similar to the MLA portion of a nanoblock, with the difference that it does not have inline NDR latches, I/O ports and connections to VDD and Gnd. A switchblock is formed by 4 nanoblocks. Crossing horizontal wires and vertical wires from the surrounding nanoblocks are connected by configurable molecular switches. If the size of the nanoblocks is k×k, then there are 2k vertical wires and 2k horizontal wires inside a switchblock and 4k2 cross-points can be formed [1].

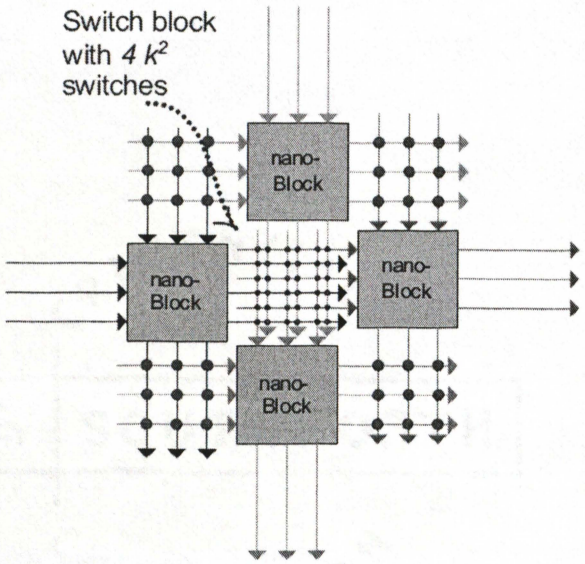


Fig. 4 A switchblock with four surrounding nanoblocks [1]

3. RELATED PREVIOUS WORKS

Defect tolerance methodology is presented in [2] such that components of a nanofabric are configured to test circuits to infer the defect status of individual components. Only stuck-at and stuck-open faults are targeted during test and the proposed technique does not provide high recovery. Application-dependent testing of FPGA has also been proposed in [9], [10]. A defective FPGA may pass the test for a specific application. This is a one-time configuration; in other words the FPGA will no longer be reconfigurable. This reduces yield loss and yields to manufacturing cost savings.

In [3], testing is performed by configuring the blocks and switches as linear feedback shift registers (LFSR). If the final bit stream generated by LFSR is correct, all the components are assumed to be defect-free. Otherwise, there is at least one faulty component in LFSR. To diagnose, the components in LFSR and other components are used to configure a new LFSR. If the new LFSR is faulty, the component at the intersection of faulty LFSRs is considered defective. A defect database is created after completing the test and diagnosis. However only stuck-at and stuck-open faults are targeted during test and the proposed technique does not provide high recovery.

The CAEN-BIST approach presented in [7] configures a nanoblock as a tester to test its neighboring nanoblocks. Test patterns are fed to both the tester and the nanoblock under test (BUT) from an external source. A defect-free BUT generates output patterns that are identical to the input patterns. The tester compares the input test patterns and the output patterns from the BUT to see if the BUT is defective. However, CAEN-BIST is performed in a wave-like manner in which a set of nanoblocks in the same diagonal tests

another set of nanoblocks until the entire nanofabric has been tested. Therefore, the complexity and testing time depends on the size of the nanofabric under test.

Another BIST approach was proposed in [15] where the NanoBlocks can be configured as Test Pattern Generators (TPGs), Block Under Test (BUTs) or Output Response Analyzers (ORAs). These blocks, along with the corresponding Switchblocks, comprise a TG (Test Group). In a TG, the TPG generates the testing patterns for a BUT and ORAs examine the BUT output response. A total of 4k +6 configurations are needed to test for the stuck-at, stuck-open, bridging and defective cross-points.

In the Built-in Self-Test procedure discussed in [16] each NanoBlock is configured either as a Pattern Generator (PG) or a Response Generator (RG). A Test Group is created using a set of PGs, RGs and switchblock(s) between the two. The NanoBlock configured as a PG tests itself and generates the test pattern for RG. An external device is needed to program the NanoBlocks and read the RGs' responses. In the test configurations, stuck-at, stuck-open, forward biased and reverse biased diode and AND & OR bridging faults are targeted. If the size of the nanoblock is kxk, it is estimated that 8K + 5 configurations are needed to provide 100% fault coverage. This number of configurations is still large for a large value of k.

4. TESTING APPROACH

4.1. OVERVIEW

In this section, a testing approach is presented that efficiently and cost effectively identifies the faulty components. The entire nanofabric is setup based on test architectures to perform a self-test among the nanoblocks. In concert with carefully selected configurations for the nanoblocks under test, the proposed approach minimizes testing time while ensuring all fault testability. The analysis of the tests results in a defect map that indicates faulty nanoblocks.

Moreover, two methods are proposed that increase nanofabric yield by identifying partially defective nanoblocks that still can correctly implement the AND/OR logical functions. The proposed optimization approach selects the minimal subset from the entire, proposed list of configurations to test for the desired functionality only. Additionally, this test set optimization reduces the overall testing time. The second, customization method redesigns the test configurations such that only the required cross-points are tested. In contrast, the optimization technique only reduces the number of unnecessary configurations while still testing unnecessary cross-points. The drawback of the customization method is the need for creating a new set of test configurations for each desired logical function. The increased overhead may result in longer and more complex testing process.

First, a set of test architectures is presented in Section 4.2. Next, the nanoblocks configurations are introduced and discussed in Section 4.3. The optimization and customization techniques are proposed in next Section 5.

4.2. TEST ARCHITECTURES AND PROCEDURE

A Test Architecture (TA) defines the manner in which the nanoblocks and switchblocks are configured and connected. Each of the nanoblocks is configured either as a Block Under Test (BUT) or a comparator (C) [17] and the switch blocks are used to connect the outputs of BUTs to comparators. A nanoblock configured as a BUT in one TA is configured as a comparator in the other TA. Hence, half of the total number of nanoblocks is tested in each TA. In contrast to the traditional approach utilized in FPGA type application, the proposed TAs are constrained by the fabric topology. In typical nanofabric, nanoblocks have outputs on two sides called east (right) and south (bottom). In order to test nanoblocks' functionality in both directions a total of four (4) TAs are required, as shown in Fig. 5. In the figure, B refers to a nanoblock which is configured as a BUT and C refers to a nanoblock configured as a comparator.

The proposed approach defines two pairs of complementary TAs: TA-1 for outputs on the east side, and TA-2 for outputs on the south side. TA-1b is the complement of TA-1a since a BUT in TA-1a is a comparator in TA-1b and vice versa. Similarly, TA-2b is the complement of TA-2a. The selection of the particular test sets (TA-1a and TA-1b) or (TA-2a and TA-2b) is dictated by the test configuration as discussed in the subsection 4.3.

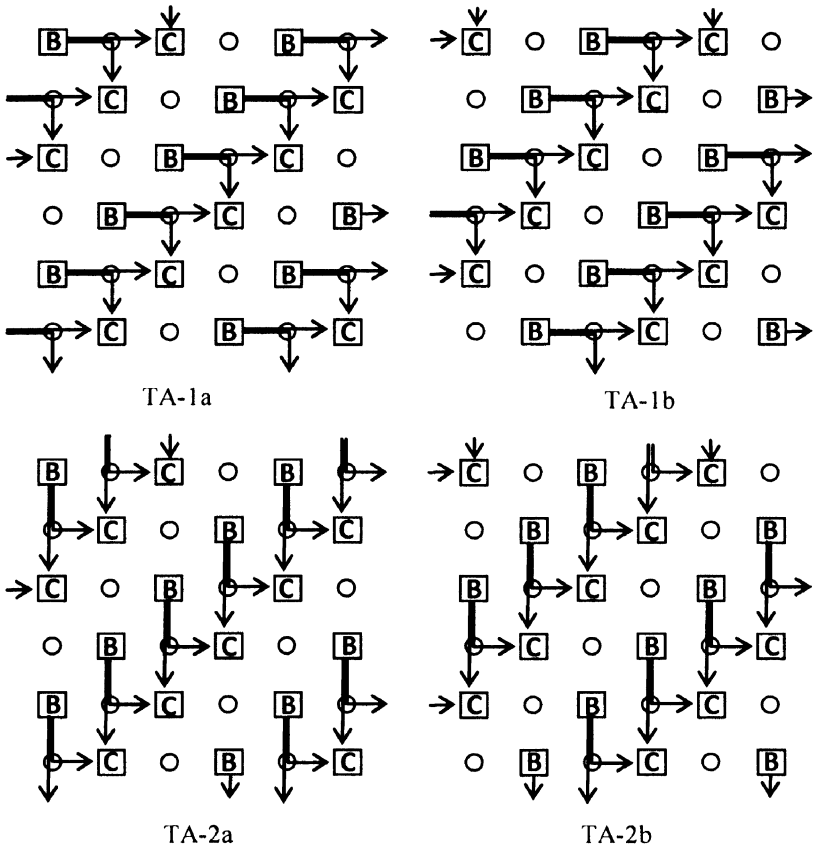


Fig. 5 Test architectures

The test patterns are applied simultaneously using an external tester. Each comparator compares outputs from two BUTs and each BUT output is connected to two comparators. The output of the comparator will be successful if the BUTs being compared and the comparator itself are defect free. If any of the BUTs or the comparator itself is faulty, it's an unsuccessful comparison. Since the defect rate is of the order of 10%-15%, it is assumed that the probability of two defective BUTs being compared by the same comparator is very low. It is assumed that the comparator generates a "0" for a successful comparison and a "1" for an unsuccessful comparison. This helps generate an intermediate defect map called the partial defect map and, in turn, the final defect map. A

test is run for each type of fault to be targeted to create Partial Defect Maps for corresponding faults. Combining all the partial defect maps gives the final defect map, which the compiler can use to configure the nanofabric by avoiding defective blocks.

4.2.1. Testing Procedure. A block is declared fault-free only if it does not manifest any of the faults targeted. Initially the Final Defect Map is set to NULL. Next, the BUT configuration is optimized/customized for the targeted fault. A raw defect map is a defect map for a particular fault. For each targeted fault, the raw defect map is initially set to null. The first test architecture is then selected and the corresponding BUT configuration and test patterns are applied. The comparator outputs are read out and possible faulty blocks are marked as suspects. The complementary test architecture is then applied and the procedure is repeated. The comparator outputs are read out and a decision is made whether the marked blocks are actually faulty or not. Figure 6 gives the flowchart for this testing procedure.

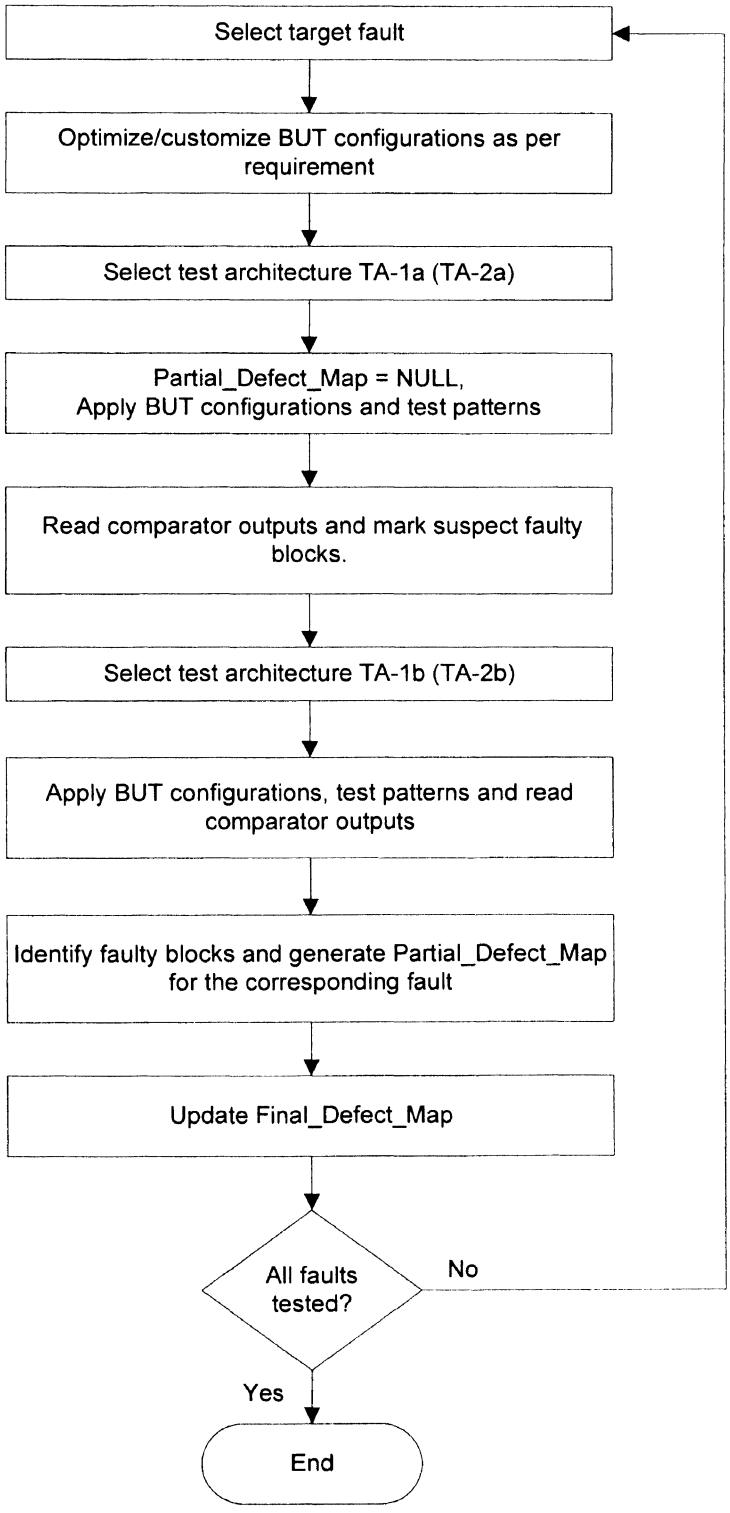


Fig. 6 Testing flow

4.2.2. Algorithm for Locating Defects. The comparator detects inconsistencies between the outputs of two BUTs, which indicate a presence of a fault. In order to uniquely identify the defective nanoblocks two things need to be analyzed - (1) outputs of adjacent comparators and (2) test results in the complementary architecture. Once the defect is located the Raw Defect Map is updated accordingly. The faulty block identification algorithm is presented in Table I. In general, the faulty block is identified when it is marked as faulty by two corresponding comparators. The complementary test architecture is used to identify faults in the comparators.

Table I. Pseudo-code of the faulty block detection algorithm

1. *Assign a score of zero (0) to all the nanoblocks.*
2. *For each TA:*
    a. *For each comparator:*
        i. *If the comparator gives false output, increment score "+1" for each of the corresponding BUTs*
    b. *For each BUT:*
        i. *If total score is equal to two (2) then block is marked as defective*
    c. *Create the raw defect map.*

4.2.3. Example of a Faulty Block Detection. Figure 7 illustrates a section of nanofabric for two test architectures with the blocks numbered 1 through 18. The two complementary architectures are needed to ensure that each of the blocks is tested.

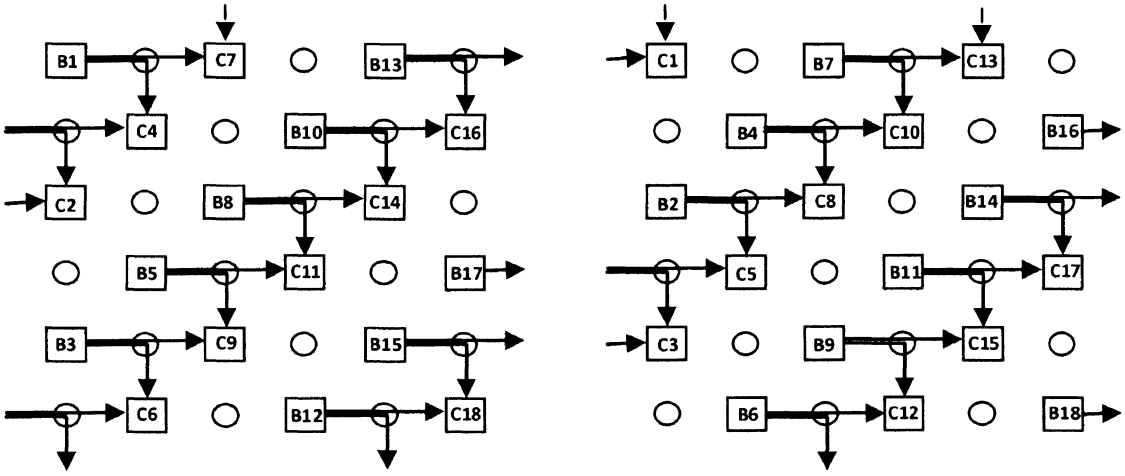


Fig. 7 Subset of nanofabric in TA-1a and TA-1b

Consider the following assumptions:

- The fault targeted is F1 and the test architectures used are TA-1a and TA-1b.
- The size of the nanofabric is 6x6.
- Blocks B4 and B5 have faults F1.

It can be observed that block B5 is tested in TA-1a and block B4 is tested in TA-1b. Consider the first test architecture TA-1a. The blocks C2, C4, C6, C7, C9, C11, C13, C16 and C18 are configured as comparators and compare the outputs of the BUTs. Initially all the nanoblocks are assigned a score of "0".

Due to the presence of the faulty block B5, the following comparators would generate a "1" at the output:

- C9: Blocks B3 or B5 have fault F1.

  Increment B3 and B5 by "1". B3=1, B5=1.

- C11: Blocks B5 or B8 have fault F1

  Increment B5 and B8 by "1". B5=2, B8=1.

The remaining comparisons are successful. Analyzing the above results, the following is obtained:

B3=1

B5=2

B8=1

Rest of the BUTs = 0.

Therefore, eliminating all the blocks with a score of "1" and "0" and only retaining blocks with a score of "2" as faulty blocks, block B5 is diagnosed as faulty and the defect map is updated accordingly. Similarly the faulty block B4 is identified using test architecture TA-1b. Since the nanoblock size is very large, efficient data structures based on Bloom filters have been proposed for storage of the defect map [19].

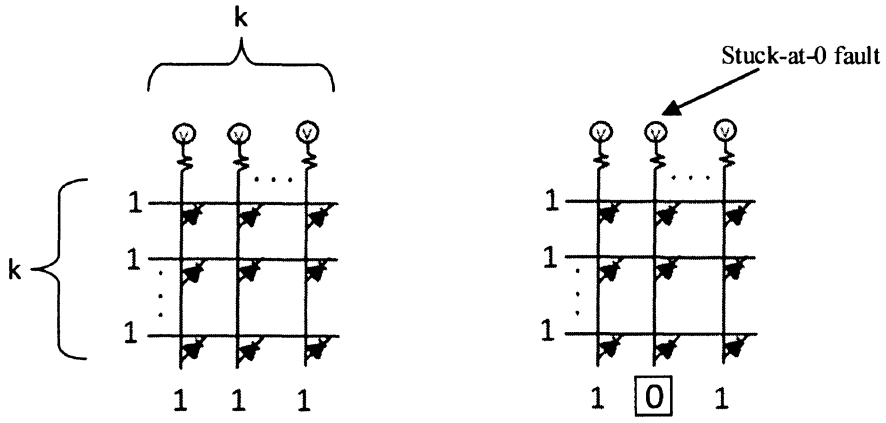
4.3. DESIGN OF NEW BUT CONFIGURATIONS, TEST PATTERNS AND FAULT COVERAGE ANALYSIS

In order to test the BUTs for defects, they need to be configured internally and test patterns need to be applied. Configuring a BUT refers to programming the various cross-points in the nanoblock and applying the desired voltages to the nanowires. The BUT configuration and the test pattern target specific fault types. In turn, the particular

configuration dictates which pair of complementary test architecture has to be used based on the direction of inputs and outputs. In each configuration, one or more faults can be targeted.

In general the faults can be divided into two categories: nanowire faults and the cross-point faults. The former affects the entire vertical or horizontal wire in a nanoblock, thus rendering it useless. The latter affect only individual cross points and can potentially have no effect on the correctness of a logic function implemented using the remaining good cross-points inside the particular block. The configurations are grouped in two sets: (a) Set I that targets the nanowire faults and includes configurations C1 through C4, and (b) Set II that targets the cross-point faults and includes configurations C5 and C6. The following faults are targeted in the next subsection: stuck-at and stuck-open faults, connection faults, and bridging faults using a set of test architectures, BUT configurations and test patterns.

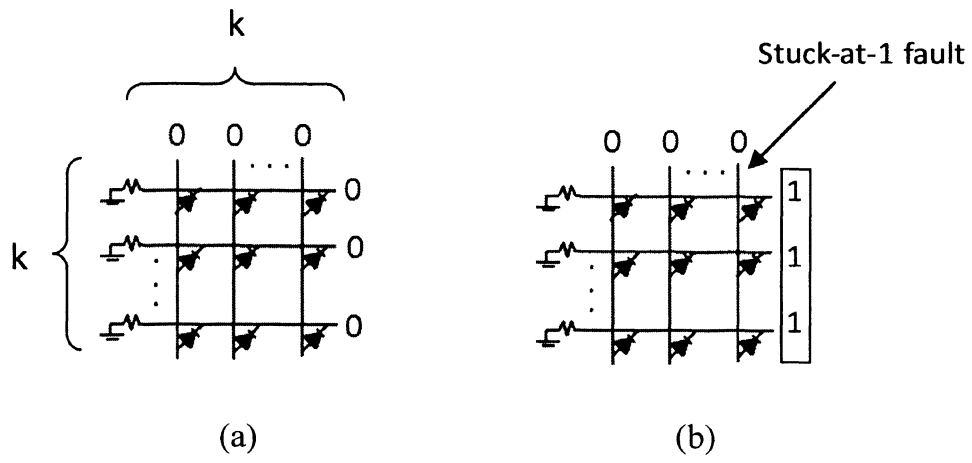
4.3.1. Configuration C1: Stuck-at-0 and Stuck-Open Faults. The BUT configuration C1 is shown in Fig. 8(a). Here, all the junctions are programmed as diodes and all inputs are connected to "1". Hence all outputs should be "1" for a fault free BUT. However if any of the lines has a stuck-at-0 and/or stuck-open fault, the corresponding output will be "0". Test architectures TA-1a and TA-1b will be used in conjunction with this BUT configuration since the inputs are on the west and outputs on the south of the nanoblock. Consider for instance that a vertical line has a stuck-at-0 fault as shown in Fig. 8(b) and hence is permanently tied to 0. As a result, the output for the vertical line 2 will be a "0" while for all other defect free lines the output will be a "1".



(a) (b)

Fig. 8 (a) BUT configurations and test patterns for C1 (b) Example fault detection

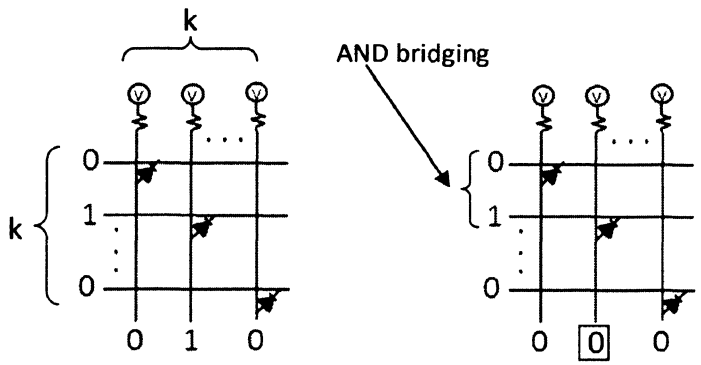
4.3.2. Configuration C2: Stuck-at-1 Faults. The BUT configuration C2 is shown in Fig. 9(a). Here all the junctions are programmed to behave as diodes and all inputs are connected to "0". Hence, for a defect free BUT all the outputs should be "0". However, if any of the lines has a stuck-at-1 fault, the corresponding output will be "1". Test architectures TA-2a and TA-2b will be used in conjunction with this BUT configuration since the inputs are on the north and outputs on the east of the nanoblock. Consider for instance that a vertical line has a stuck-at-1 fault as shown in Fig. 9(b) and hence is permanently tied to 1. As a result, the outputs will be "0".



(a) (b)

Fig. 9 (a) BUT configurations and test patterns for C2 (b) Example fault detection

4.3.3. Configuration C3: AND/OR Bridging Faults (H/H). The bridging faults between adjacent horizontal wires can be detected by using the configuration shown in Fig. 10(a). The BUT is configured in such a way that the programmed diodes are forward biased. Consider that there is AND-bridging between the second and third horizontal wire as shown in Fig. 10(b) and as a result the second horizontal wire is pulled down to a "0", the output will be inverted. Test architecture TA-1a and TA-1b will be used in conjunction with this BUT configuration.



(a) (b)

Fig. 10 (a) BUT configurations and test patterns for C3 (b) Example fault detection

4.3.4. Configuration C4: AND/OR Bridging Faults (V/H). This is similar to the previous configuration, and tests for bridging faults between the vertical wires. The configuration is as shown in Fig. 11(a). The inputs applied and the expected outputs are as shown. However if there is any AND/OR bridging between the vertical wires, the output changes. Consider for example that the first vertical wire changes to a "1" due to OR bridging with its neighboring wires. This is shown in Fig. 11(b). The diode on that wire will now be reverse-biased and hence act as open switch and the corresponding horizontal output will become a "1". Test architecture TA-2a and TA-2b will be used in conjunction with this BUT configuration.

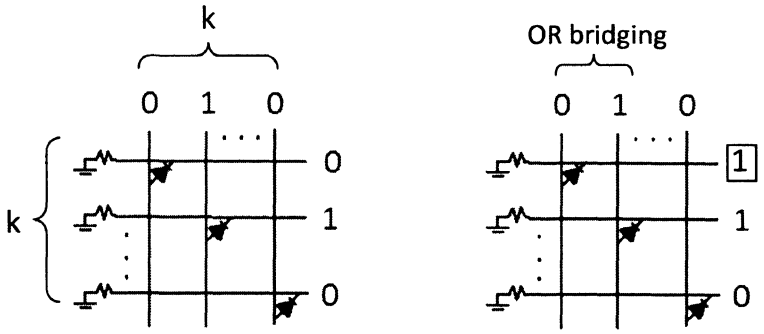
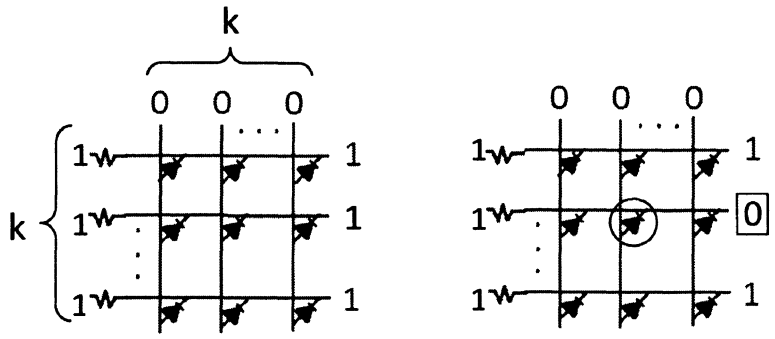


Fig. 11 (a) BUT configurations and test patterns for C4 (b) Example fault detection

4.3.5. Configuration C5: Reverse Biased Diode and V/H Bridging Faults. Here all the junctions are programmed to behave as diodes. Horizontal wires are connected to "1" while the vertical wires are connected to "0" as shown in Fig. 12(a). The output can be taken either from the east or from the south side. Hence either of the two test architecture sets can be used. Such a configuration reverse biases all the junction diodes and the outputs are all 1's if taken on the east side and all 0's if taken on the south side. In the figure shown, outputs are taken on the east. If any of the reversed biased diode is defective and has a small resistance, or there is bridging between the horizontal and vertical wires, the output on the east side will be pulled down to "0". This configuration thus detects defective reversed biased diodes and bridging faults between vertical and horizontal wires. Consider the example shown in Fig. 12(b). The circled diode is faulty and offers a small resistance in the reverse biased state. Hence the corresponding horizontal output changes to a 0.



(a) (b)

Fig. 12 (a) BUT configurations and test patterns for C5 (b) Example fault detection

4.3.6. Configuration C6: Forward Biased Diode Faults. For a kxk nanoblock, detection of forward biased diode faults requires a total of k configurations to test all the cross-points. Figure 13(a) shows these configurations. Here the vertical wires are connected to Vdd and only one junction is programmed as a diode on each of the horizontal and vertical wires in each configuration. Thus in the first configuration, the cross-points are programmed in a diagonal fashion. The different sub-configurations are obtained by shifting the programmed cross-points one position to the right. The outputs are "0" for a defect free BUT since the diodes are forward biased and transmit a "0" to the output. If any of the forward biased diode is defective as shown in Fig. 13(b), the corresponding vertical wire output will be pulled up to a "1". Test architecture TA-1a and TA-1b will be used in conjunction with this BUT configuration.

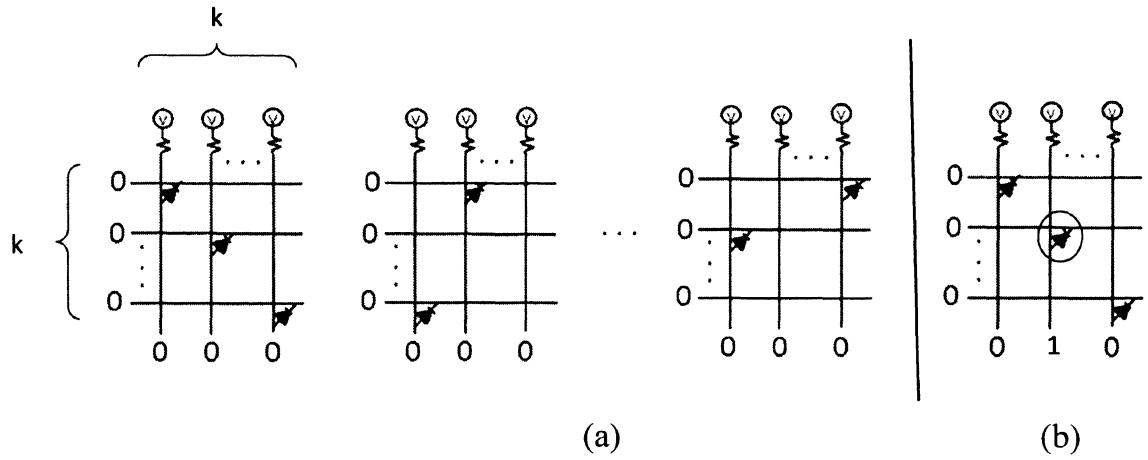


Fig. 13 (a) BUT configurations and test patterns for C6 (b) Example fault detection

4.4. FAULT COVERAGE AND NUMBER OF CONFIGURATIONS

Table II summarizes the various faults covered using these configurations, and the BUT configurations-test architectures to be used for each of the faults. A Total of 2k+10 configurations are needed to test the entire nanofabric for the given faults. This number is reasonably small compared to other BIST techniques and also provides sufficient fault coverage. Also, it is independent of the number of nanoblocks in a nanofabric. These configurations can be modified as per the optimization technique which will be described in Section 5. By optimization, the main aim is to reduce the number of configurations and/or the number of cross-points programmed thereby reducing the testing time.

Table II. Fault coverage and configurations

Fault Model	Configuration					
	Set I				Set II	
	C1	C2	C3	C4	C5	C6
Stuck-at-0	x					
Stuck-at-1		x				
open-line	x					
AND bridging (h/h)			x			
AND bridging (v/v)				x		
OR bridging (h/h)			x			
OR bridging (v/v)				x		
OR bridging (v/h)					x	
Cross-points (forward)						x
Cross-points (reverse)					x	
Configurations/test patterns	1	1	1	1	1	k
Test Architecture	TA-1	TA-2	TA-1	TA-2	TA-1 /TA-2	TA-1
Elimination of redundant configurations	N	N	N	N	N	Y
Customization for effective cross-point testing	N	N	N	N	Y	Y

5. OPTIMIZATION TECHNIQUE

The BUT configurations described above are used to test for one or more faults. Thus it gives the user a flexibility to choose the configurations depending upon which faults are targeted. Also, each fault detecting configuration will have its own cost associated with it i.e. testing time, complexity etc.

The proposed testing strategy can be optimized such that utilization of the nanoblocks is increased, the testing time reduced, and the number of required configurations decreased. Exhaustive test includes all configurations C1 through C6, where all cross-points and nanowires of each nanoblock are tested for all faults. However, it may not be necessary to test all the cross-points/NWs in a nanofabric since there are redundant cross-points. Consequently, the entire test process includes unnecessary checks and it is possible to test only a subset of cross-points and NWs, which are required to implement a logic function. For example, the optimization scheme would select a subset of tests that target only the needed cross-points out of all k^2 ones. First, the list of cross-points/NWs tested in each configuration is tabulated. Next, the columns that correspond to the cross-points not required by the targeted logic function are removed. Then, using Petrick's method of reduction, the minimal test set required for the targeted faults is obtained. In fact, the Petrick's method will list all possible combinations of the tests that are required. Consequently, it is straightforward to assign various weights to performance metrics such as testing time, configuration time thus enabling a more flexible optimization.

A typical logic implementation in nanoblocks realizes AND and OR gates only [13], since these are the basic gates employed when implementing any logic function.

Note, that nanoblocks require external component to implement NOT logic function thus complementing the minimal and complete set of gates. A k×k nanoblock can be used to realize an AND/OR gate with (k-1)-inputs.

Figure 3 shows an example of a 2-input AND/OR gate implemented using a 3x3 nanoblock. It can be observed that all the NWs need to be fault free. However, only 4 out of the total 9 cross-points are programmed and take part in implementing the logic function. These cross-points are referred to as effective cross-points. Similarly for a 4x4 nanoblock, in order to implement a 3-input AND/OR gate, there are 6 effective cross-points. In general for a k×k nanoblock, in order to implement a (k-1) input AND/OR gate, only 2(k-1) effective cross points are needed. However, all the NWs need to be tested. The BUT configurations and test patterns described in Section 4.3 have been split into two sets – set I which tests the nanowire faults and set II which tests the cross-point faults. Thus it is possible to optimize the BUT configurations C5 and C6 which belong to set II, in such a way so as to test only the effective cross-points.

The performance will be evaluated using utilization metric, which is a probability that a nanoblock is successfully used to implement a logic function. Considering the effective cross-points, the utilization of nanoblocks can be improved since faults in unused cross points will neither be tested nor interfere with correct function implementation. The increase in utilization is due to the fact that it is possible to use a defective nanoblock to implement a logic function as long as the effective cross-points are defect free.

Consider configuration C5 where in all the cross-points are tested. This needs to be modified to test only the required cross-points. The optimized configuration C5 is

shown in Fig. 14. Here again only the effective cross-points are tested based on the assumption that nanoblocks implement only AND and OR gates. For this, it is required program (k-1) cross-points on one vertical line and (k-1) cross-points on one horizontal line. The number of cross-points programmed (and tested) is thus reduced from k^2 to 2(k-1). This reduces a lot of time-overhead associated with programming the cross-points and also increases the utilization of the nanoblock.

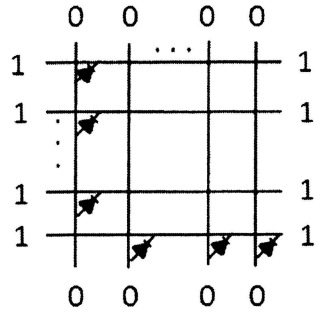


Fig. 14 Customized configuration C5

Now consider the configuration C6 shown in Fig. 13(a) in which all the cross-points are tested. Here optimization using a 3x3 nanoblock is demonstrated for simplicity. Three sub-configurations are needed to test all the cross-points (for a 3x3 nanoblock). Two approaches to optimize C6 are discussed and each has its own pros and cons.

5.1. ELIMINATION OF REDUNDANT CONFIGURATIONS

It is possible to eliminate the efforts required in designing new configurations to test only the effective cross-points by selecting a subset from the defined sub-configurations of C6. Since C6 needs a total of k sub-configurations to test all the cross-points, a table can be laid out which lists the different cross-points tested in each of the configuration of C6, and then from this table select only the set of configurations which are needed to test the effective cross-points. This is demonstrated below.

Table III shows the cross-points tested in each of the sub-configurations of C6 for a 3x3 nanoblock. Here, the assumption is that nanoblocks are used to implement AND and OR gates only. Hence, from Fig. 3, it can be observed that there are only 4 effective cross-points - cp11, cp21, cp32, cp33. From Table III, it can be seen that these four cross-points can be tested in two configurations C61 and C63 and the configuration C62 is redundant and can be avoided. Thus, only 2 configurations are needed instead of 3 as a result of optimization. This can be extended to a k×k nanoblock, wherein the total number of configurations required is now (k-1) instead of k. Moreover, since the number of cross-points to be programmed is reduced, this leads to a considerable reduction in testing time.

Table III. Junction cross-points tested in each configuration

	Junction cross points tested								
Configuration	cp11	cp12	cp13	cp21	cp22	cp23	cp31	cp32	cp33
C61	x				x				x
C62		x				x	x		
C63			x	x				x	

5.2. CUSTOMIZED CONFIGURATIONS

The advantage of the above approach is that there is no need to design new configurations. It is possible to simply select the desired configurations with the help of a table. This is simpler and less time consuming. However, It can be seen from the above table that in addition to the effective cross-points, some additional cross-points are also tested (six cross-points in the above case instead of four) which are not required. This reduces the utilization slightly as opposed to the maximum utilization that can be obtained. To have maximum utilization, custom configurations can be designed wherein only the cross-points which are necessary are tested. Figure 15 shows the customized configurations for C6 for a 3x3 nanoblock.

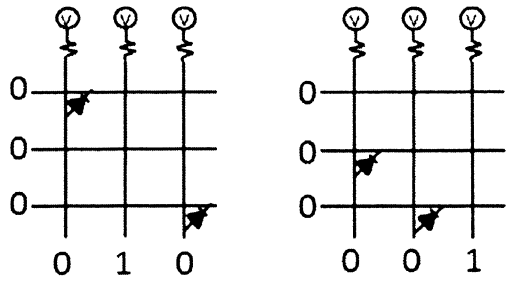


Fig. 15 Customized configuration C6

Here, two configurations are required but only the effective cross-points are tested. This gives the maximum utilization with a slightly reduced testing time than the first approach. However, this demands the design of new customized configurations and not re-using existing ones.

5.3. RECOVERY PROCEDURE

Once the entire testing phase has been completed and the faulty nanoblocks identified, an additional recovery procedure could be used to diagnose the faulty nanoblocks and further increase the utilization of nanoblocks. This is optional and may be carried out only if utilization is a major concern since this will increase the total testing time.

A defective nanoblocks implies that it cannot implement AND and OR logic within the same nanoblock. However since the cross-points used to implement AND and OR logic are distinct, there can be nanoblocks among the defective ones, which can implement either of the two functions but not both. An additional testing phase can be carried out for the defective nanoblocks to test for the AND or OR implementation. The BUT configurations need to be modified accordingly since a nanoblock is tested only for one implementation and hence, only the corresponding cross-points need to be tested. Thus at the end of these test phases, there are three sets of usable nanoblocks- (i) those which can be used to implement both AND and OR function, (ii) those which can implement only OR function, and (iii) those capable of implementing only AND function.

6. ANALYSIS AND DISCUSSION

6.1. NUMBER OF CONFIGURATIONS/TEST PATTERNS

The BIST approaches developed by authors in [15] and [16] have been discussed in Section 3. It can be seen from Table I that the total number of configurations/test patterns required to test the entire nanofabric is 2k+10. While after using the optimization technique, it results in 2(k-1)+10. From the analysis of BUT configurations in Section 4.3 it can be deduced that only configuration C6 depends on the value of k while rest of them are independent of k.

6.2. NANOBLOCK UTILIZATION

The term nanoblock utilization is defined as the probability that a nanoblock can be successfully utilized to implement a logic function in the presence of defects. A mathematical expression will be derived for nanoblock utilization for the optimized technique and compared with that of a non-optimized approach.

Assume a nanoblock of size kxk. Let p = probability of a faulty cross-point. For a non-optimized testing approach, a nanoblock is deemed as faulty if at least one of its cross-points is found to be faulty. Hence, the condition for a nanoblock to be utilized successfully is that all its cross-points have to be fault free.

Utilization (non-optimized) = Probability {all cross-points are fault free}

$$U1 = (1-p)^{k^2} \tag{1}$$

For the optimized technique, as described in Section 5, there are 2(k-1) effective cross-points for a kxk nanofabric. In order for the nanoblock to be successfully utilized to

implement the logic function (AND/OR), these 2(k-1) cross-points need to be fault free. Hence the utilization in this case is given by:

Utilization (optimized) = Probability {2(k-1) cross-points are fault free}

$$U2 = (1-p)^{2(k-1)} \quad (2)$$

6.3. TESTING TIME

The total testing time for the nanofabric can be split into a number of components. Let tconfig denote the total time required to configure the BUT i.e. to program the various cross-points and set up power connections. Thus tconfig would not be the same for all configurations since the number of cross-points programmed is not the same for all the configurations.

Let tp denote the time required to program a single cross-point and ts be the time required to set up the Vdd/Gnd connections to the wires. Let ttest denote the time required to apply a test pattern to the BUT, wait for the comparator outputs to be generated, read the comparator outputs and generate the partial defect map. Here it is assumed that ttest >> tconFig. Using these different components, the total time required for testing a nanofabric can be calculated for a kxk nanoblock.

Let T_i denote the testing time for configuration Ci (as described in Section 4.3), T_i^* denote the testing time for the configuration Ci after using the optimization technique and T_i^{**} denote the testing time after using customization approach. The total testing time is given by the sum of the testing times for each of the individual configurations. Table IV gives a summary of the testing time for optimization and customization approaches.

$$T_1 = T_2 = T_5 = 2 * \{(k^2 * t_p) + t_s + t_{test}\} \quad (3)$$

$$T_3 = T_4 = 2 * \{(k * t_p) + t_s + t_{test}\} \quad (4)$$

$$T_6 = 2 * \{[(k * t_p) + t_{test}] * k + t_s\} \quad (5)$$

$$T_1^* = T_2^* = 2 * \{(k^2 * t_p) + t_s + t_{test}\} \quad (6)$$

$$T_3^* = T_4^* = 2 * \{(k * t_p) + t_s + t_{test}\} \quad (7)$$

$$T_5^* = 2 * \{(2(k-1) * t_p) + t_s + t_{test}\} \quad (8)$$

$$T_6^* = 2 * \{[(k * t_p) + t_{test}] * (k-1) + t_s\} \quad (9)$$

$$T_5^{**} = 2 * \{(2(k-1) * t_p) + t_s + t_{test}\} \quad (10)$$

$$T_6^{**} = 2 * \{[(k-1) * t_p + t_{test}] * (k-1) + t_s\} \quad (11)$$

Table IV. Summary comparison of the testing time

		Change due to Optimization	Change due to Customization
Wire Testing Time	C1	$\mathbf{T_1^* - T_1 = 0}$	$\mathbf{T_1^{**} - T_1 = 0}$
	C2	$\mathbf{T_2^* - T_2 = 0}$	$\mathbf{T_2^{**} - T_2 = 0}$
	C3	$\mathbf{T_3^* - T_3 = 0}$	$\mathbf{T_3^{**} - T_3 = 0}$
	C4	$\mathbf{T_4^* - T_4 = 0}$	$\mathbf{T_4^{**} - T_4 = 0}$
Cross-point Testing Time	C5	$\mathbf{T_5^* - T_5 = -2t_p[k-1]^2}$	$\mathbf{T_5^{**} - T_5 = -2t_p[k-1]^2}$
	C6	$\mathbf{T_6^* - T_6 = -2[kt_p + t_{test}]}$	$\mathbf{T_6^{**} - T_6 = -2k[(k+2)t_p - t_{test}]}$
Total Difference		$\mathbf{-2[t_p(k^2 + k - 1) - t_{test}]}$	$\mathbf{-2[t_p(2k^2 - k + 1) - kt_{test}]}$

6.4. RECOVERY PROCEDURE

The recovery procedure is discussed in Section 5.3. Here, the nanoblock would be usable if it can implement either an AND or OR logic but not both. Though the functionality of the nanoblock is reduced, it still is usable. Hence an increase in the utilization of the nanoblock is expected.

Assuming the parameters defined in Section 6.2, the utilization after using recovery procedure would be given as

$$U_3 = (1 - p)^{k-1} \tag{12}$$

7. RESULTS

Figure 16 shows the comparison between our method and the approaches discussed in [15] and [16] with respect to the number of configurations required for testing a kxk nanofabric.

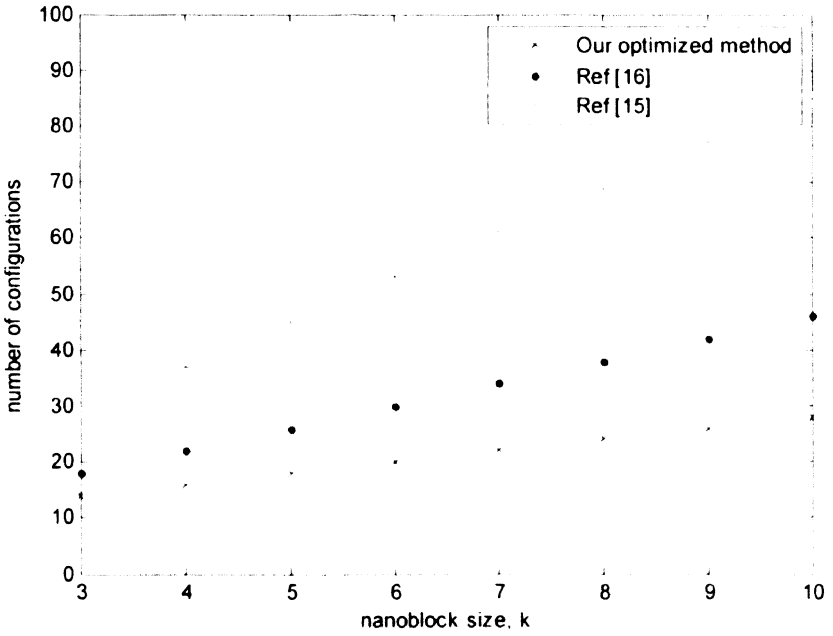


Fig. 16 Number of configurations required in comparison with the approaches discussed in [15] and [16]

Figure 17 shows the plots of (1) and (2) for a value of k between 3 and 10. It can be seen that the optimization technique leads to a significant improvement in the utilization of the nanoblock. This implies that the probability of a nanoblock being rendered unusable is greatly reduced.

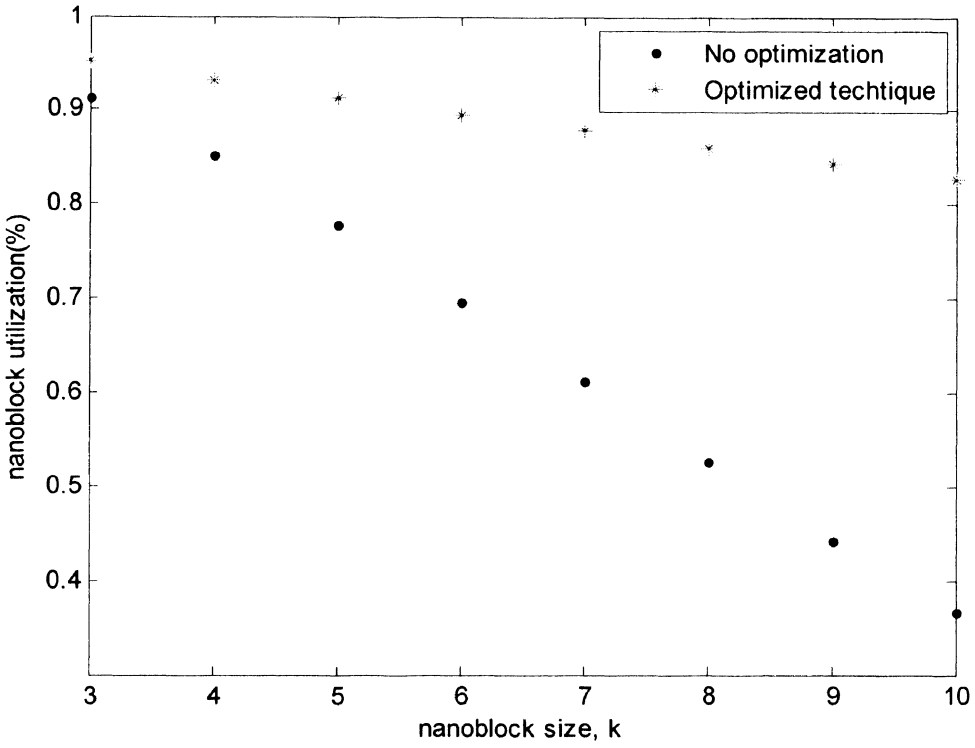


Fig. 17 Improvement in utilization of the nanoblock

Figure 18 shows Total testing time for a nanoblock size varying between 3 and 10. The improvement in testing time increases with the nanoblock size (k) since the fraction of effective cross points to the total number of cross points decreases with the nanoblock size. The number of effective cross points grows linearly with the nanoblock size, while the total number of cross points increases exponentially (k^2).

Figure 19 shows the plot of (12) for a nanoblock size between 3 and 10. The utilization of the nanoblocks improves since the customized configurations target only the effective cross points instead of fewer configurations. Consequently, only defects related to the effective cross points are detected and will mark the block as faulty.

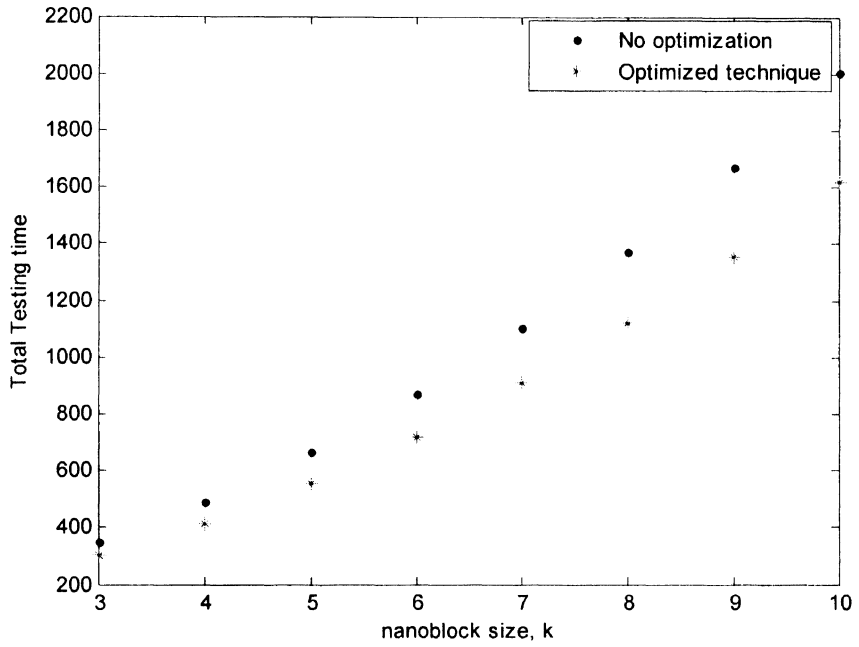


Fig. 18 Reduction in total testing time after optimization

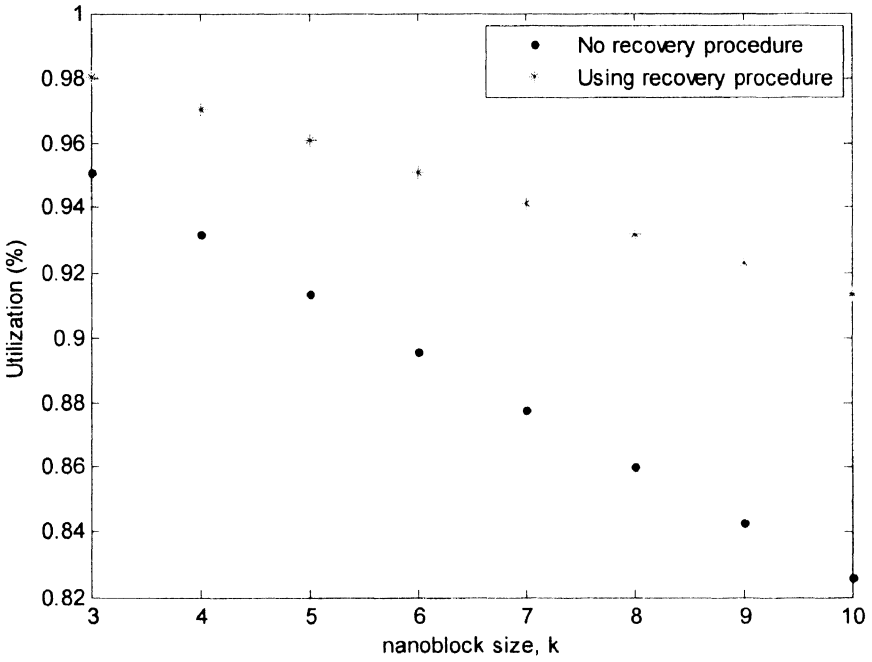


Fig. 19 Improvement in utilization using recovery procedure

In the case of optimization technique, unnecessary configurations were removed from test while still testing the redundant cross points in the remaining configurations. However, since only partial functionality is tested, additional tests need to be carried out. This results in an increase in the number of tests and testing time. Figure 20 shows the increase in testing time as a result of using recovery procedure. Hence, the improvement in utilization is obtained at the expense of increased testing time. As a result, the recovery procedure should be used only in cases where utilization is more important than testing time.

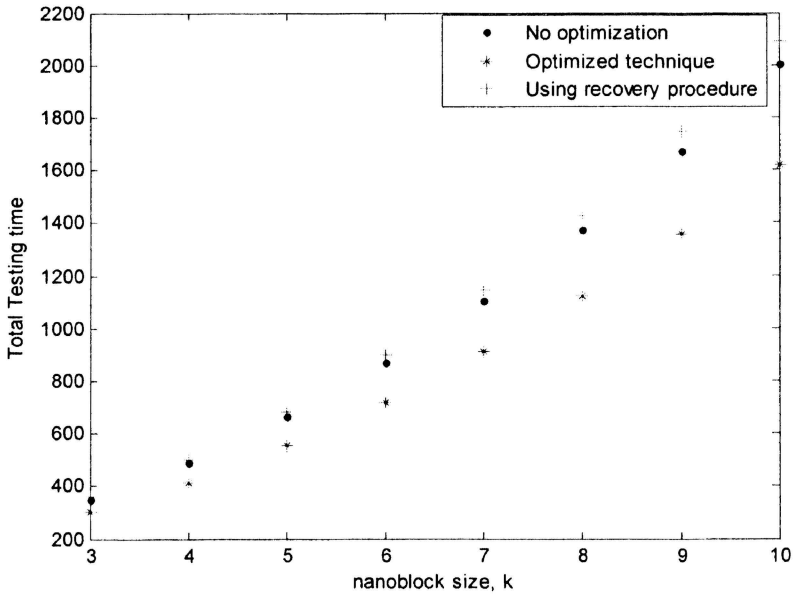


Fig. 20 Increase in testing time as a result of using recovery procedure

8. CONCLUSION

The presented, novel testing procedure has been show to outperform the existing testing approaches. The proposed configurations and test patterns provide 100% fault coverage for stuck-at, stuck-open, connection and bridging faults. The parallel testing architectures is does not increase testing time with nanofabric size thus making it suitable for the dense architectures. Also, it requires only 2k+10 configurations, which corresponds to 23% reduction over the method discussed in [15] and 55% reduction over the method proposed in [16] for nanoblock size of k=5. Moreover, using the optimization technique described here, the nanoblock utilization is increased from 76% to 91% for nanoblock size k=5. A significant reduction in testing time of the nanofabric is also observed. For example for k=5, the testing time is reduced by 29%. Moreover, the proposed recovery procedure can tailor testing to a particular functionality (either AND-gate or OR-gate). This further increases the utilization of nanoblocks from 91% to 96% for k=5. In summary, the proposed technique is simple, efficient, quick, and outperforms the existing approaches. Further work will include theoretical analysis to demonstrate optimality of the proposed approach and later development of a new, fault-tolerant design methodology.

9. REFERENCES

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

[2] M. Mishra and S. C. Goldstein, "Scalable Defect Tolerance for Molecular Electronics," workshop on Non-Silicon Computing, 2002

[3] M. Mishra and S. C. Goldstein, "Defect Tolerance at the End of the Roadmap," in Proc. Int. Test Conf. (ITC'03), pp. 1201-1210, 2003

[4] R.M.P. Rad; M. Tehranipoor, "A Reconfiguration-based Defect Tolerance Method for Nanoscale Devices," 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2006. DFT '06, pp. 107 – 118

[5] J. Dai, L. Wang; F. Jain, "Analysis of Defect Tolerance in Molecular Crossbar Electronics," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2009, pp. 529 – 540

[6] A. Al-Yamani; S. Ramsundar,; D. Pradhan, "A Defect Tolerance Scheme for Nanotechnology Circuits," IEEE Transactions on Circuits and Systems I: Regular Papers, 2007, pp. 2402 – 2409

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

[8] C. Stroud, S. Konala, P. Chen and M. Abramivici, "Built-In Self-test of Logic Blocks in FPGAs (finally, a free lunch: BIST Without Overhead)," in Proc. IEEE VLSI Test Symposium (VTS'96), pp. 387-392, 1996

[9] M. B. Tahoori, E. J. McCluskey, M. Renovel and P. Faure, "A Multi-Configuration Strategy for an Application Dependent Testing of FPGAs," in Proc. IEEE VLSI Test Symposium (VTS'04), pp. 154-159, 2004

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

[11] C. Metra, G. Mojoli, S. Pastore, D. Salvi and G. Sechi, "Novel Technique for Testing FPGAs," in proc. Design, Automation and Test in Europe (DATE'98), pp. 89-94, 1998

[12] S. J. Wang and T.M. Tsai, "Test and Diagnosis of Faulty Logic Blocks in FPGAs," in IEE Proceedings: Computers and Digital Techniques, pp. 100-106, 1999

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

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

[15] Z. Wang and Chakrabarty, K., "Built-in Self-Test of Molecular Electronics-Based Nanofabrics," European Test Symposium, 2005, pp. 168 – 173

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

[17] M. V. Joshi and W. K. Al-Assadi, "A BIST approach for configurable nanofabric arrays," 8th IEEE Conference on Nanotechnology, 2008, pp. 695 – 698

[18] R. Zamanlooy, B. Ayatollahi, A., "Modified CAEN-BIST algorithm for better utilization of nanofabrics," International Conference on Electrical and Computer Engineering, 2008. ICECE 2008, pp. 297 – 301

[19] G. Wang, "On the Use of Bloom Filters for Defect Maps in Nanocomputing," Proceedings ICCAD 2006, pp: 743-746

II. INTRODUCTION TO A NOVEL DEFECT-AWARE LOGIC MAPPING APPROACH FOR CROSSBAR-BASED NANOFABRICS

Sambhav Kundaikar, Maciej Zawodniok

Electrical and Computer Engineering

Missouri University of Science and Technology

Rolla, MO 65401, USA

Email: sdk8v5@mail.mst.edu, mjzx9c@mst.edu

Abstract

Nanofabric architectures using nanowire crossbars have shown promising potential for future nano-scale circuit design. However, due to the chemical self-assembly and alignment process, the defect rates are much higher than those of the conventional CMOS technology thus posing different design challenges. Specifically, the nanoblocks have a high level of redundancy that allows implementing a correct logic function also inside a partially defective block. In this paper a novel defect-aware logic mapping and routing technique is proposed. Additionally, compared with the previous works, our technique considers logic mapping at the nanoblock level. The proposed method is suitable for AND/OR function realizations and is a more time- and cost-effective design when compared with the existing techniques. Also, the impact of defects on logic mapping and routing is discussed and analyzed mathematically.

1. INTRODUCTION

Conventional lithography-based CMOS technology faces serious challenges due to its fundamental physical limits such as ultra-thin gate oxides, short channel effects, doping fluctuations cross the chip and increasingly difficult and expensive lithography. Therefore, new technologies are needed to replace CMOS in the future to continue the Moore's Law. It has been shown that using bottom-up self-assembly techniques, it is possible to build nano-scale devices, such as carbon nanotubes (CNTs) and silicon nanowires (NWs), without relying on lithography to define the smallest feature size [1]. Each cross point can be made to operate as a reprogrammable switch and can be used as a memory cell. These switches can be programmed using the signal lines by applying an appropriate (high) voltage. Such devices offer a prospect of high integration density (10^{10} to 10^{12} gate equivalents /cm^2) [2]. The alternative technology has been referred to as Chemically-Assembled Electronic Nanotechnology (CAEN) [3][4][5][6]. A 2-dimensional array of orthogonal NWs forms a nanoblock. These nanoblocks arranged in a regular grid form a nanofabric. In addition to high density, these architectures offer an advantage of re-configurability since they consist of reprogrammable switches (junctions).

High defect rate in a nanofabric is a major challenge and reduces yield of the nanoblocks, that is a number of nanoblocks required to implement a given logic function. Due to the non-deterministic nature of the self-assembly process, the defect rates in such devices could be as high as 10% [7]. This number is orders of magnitude larger than for the conventional CMOS technology thus potentially negating the benefits of nanotechnology. Therefore, advanced fault-tolerant design and testing techniques are

needed to increase nanofabric yield and fully exploit the strengths of nano-architectures [8][9][10][15]. Specifically, new, efficient logic mapping and routing techniques are desired. Those should counter the high defect rates in the nanofabric. Simplistic approach would be to perform a brute force, exhaustive search of possible mappings in order to find the best possible route. However, this would increase the testing and design overhead in terms of time and complexity. Moreover, the complexity of the design process increases due to the random location of defects in nanofabric and the non-deterministic, potentially dynamically reconfigurable functionality to be implemented.

The proposed mapping technique exploits the existing standard configurations of nanoblocks for the AND/OR function. The technique discussed in [10][16] explains the testing of nanoblocks implementing AND/OR functionality. As a result, the defect free nanoblocks are used to implement AND/OR logic. The nanoblock configurations for AND/OR have also been discussed in [10][16].

In this paper, the drawbacks of traditional logic mapping techniques in the context of the CAEN technology are discussed. Then, a new technique is proposed improves efficiency and simplifies design process over the existing approaches. Also, the effects of faults and defects on logic mapping and routing are analyzed.

The main contributions of the paper are: (a) A novel logic mapping approach for nanofabrics tested by the testing technique discussed in [10], (b) Study of the effects of various parameters such as the defect rate, location of defects, complexity of the function to be implemented, on the proposed mapping technique.

The rest of the paper is organized as follows. Section 2 describes the problem statement and the traditional logic mapping scheme for crossbar based nanofabric. In

Section 3 the new proposed mapping technique has been described and illustrated with a few examples. Analysis and results have been presented in Sections 4 and 5, respectively. Finally, Section 6 concludes the paper.

2. BACKGROUND AND PREVIOUS WORK

The re-configurability and functional redundancy within nanofabric and nanoblocks enables development of robust, defect- and fault-tolerant design mechanisms. When a defective resource or component is identified in the chip using test and diagnosis, the post-fabrication configuration and logic mapping can counter effects of those defects. Logic mapping onto nanoblocks refers to programming the various cross-points of the nanoblock to implement different gates and connecting them together to obtain the required logic function. A nanoblock consists of two groups of parallel nanowires which are perpendicular to each other in the same plane and programmable cross-points at the junction of these nanowires [3]. These cross-points can be programmed to behave as diodes by application of a suitable voltage levels. Nanoblock can implement AND/OR gate functionalities by programming the different cross-points and by configuring pull up and/or pull down resistors. Figure 1 illustrates example implementation of a simple logic by programming the cross-points in a nanoblock and the use of pull-up/pull-down resistors. Note that not all cross points are employed in this realization.

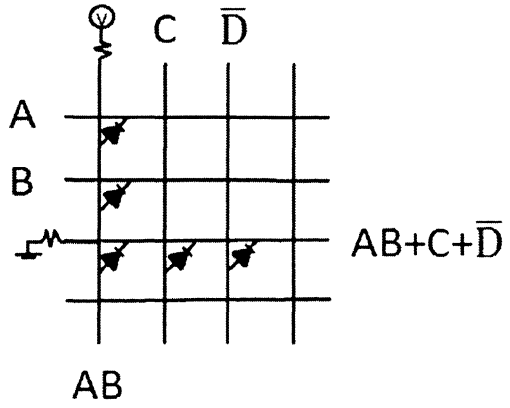


Fig. 1 Simple logic implementation

All used nanowires and cross-points need to be fault free. If any of the nanowire and/or any of the cross-points is faulty, the configuration needs to be redesigned. In most of the existing testing techniques, a nanoblock is deemed to be fault-free only if all its constituting nanowires and cross-points are fault-free. Consequently, the typically high defect rate of about 10% leads to a reduced yield and low effective logic density. In order to map logic onto a nanofabric in the fashion shown in Fig. 1, it is important to know the location of defects. If the defective nanowires and cross-points are known, the nanoblocks can be programmed to avoid these cross-points and wires while still implementing correct logic function.

Consider nanoblock with two defective cross points as shown in Fig. 2 by "x". Also, an alternative implementation of the same logic function as in Fig. 1 is shown. As demonstrated, the functional redundancy of the nanoblock enabled us to overcome minor defects in the nanoblock.

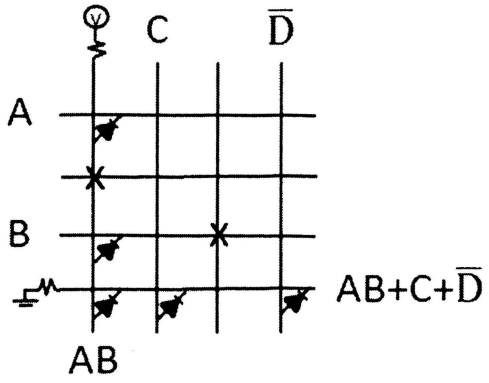


Fig. 2 Logic implementation in the presence of defects

The precise knowledge of the location of cross-points is known, the nanoblock can be programmed in a different way to implement a particular function. Also, there might be cases wherein it may not be possible to implement a function within a nanoblock if there are not enough fault-free cross-points available. Note, that any standard configurations of nanoblocks would not be able to overcome such defects and would render this block unusable. Consequently, to overcome defects each nanoblock needs to be configured in a different way depending on the number of faults, location of faults, and the logic function to be implemented. There are numerous testing techniques [8][9][15] described to diagnose the faulty nanoblocks. However, typically these techniques test for various faults in the nanoblocks and mark them as defective such that they can be avoided during the reconfiguration step. Consequently, these techniques do not support detailed information about the location of a fault within the nanoblock. Exhaustive testing has to be done to identify the fault location. This places a lot of demands on the testing algorithm to be used and also increases the complexity and testing time of the algorithm.

3. A NOVEL LOGIC MAPPING TECHNIQUE

Once the nanofabric is manufactured, it needs to be tested thoroughly to identify the faulty blocks. The testing methodology used will test for certain types and number of faults. Once the faulty areas in the chip have been identified, the next step is to map logic onto the nanofabrics by avoiding the faulty blocks. A logic mapping technique was demonstrated in Section 2. The process of logic mapping could be simplified if there were standard configurations of nanoblocks available which could implement specific functionality. This would reduce the effort needed to design custom configurations for each nanoblock. Configuring a nanoblock refers to programming the various cross-points within the block to implement logic functionality.

A novel logic mapping technique is proposed which would use the result of the testing technique discussed in [10]. Here the types of faults targeted are stuck-at faults, bridging and cross-point faults. It is assumed that each nanoblock implements only AND/OR logic and is tested to make sure it can successfully implement such a configuration. This requires only a certain set of cross-points to be fault free in order to successfully implement the function. This results in a substantial increase in the yield of the nanofabric. Also, the nanoblocks are assumed to be implementing only AND/OR gates, and the nanoblock configurations are fixed. A kxk nanoblock implementing a (k-1) input AND/OR gate is shown in Figure 3. The F1 output implements the AND function for inputs $A_1,A_2,..A_{(k-1)}$, while F2 implements the OR function for the $B_1,B_2,..B_{(k-1)}$ inputs. For such configuration, the effective cross-points (required for function realization) are known a priori. Hence, the customized testing, such as the one in [16], provides map of useful nanoblocks that can be utilized even if a partial defects are present. Consequently,

the nanoblocks marked as fault-free can implement AND/OR logic using standard configurations, such as the one in Fig. 3.

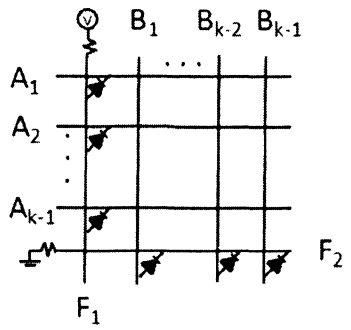


Fig. 3 AND/OR implementation

The main advantage of such an implementation is that knowledge of the location of defects inside the nanoblock is not required. Also, no effort would be needed in designing the nanoblock configurations for each nanoblock since standard AND/OR implementations are used. This implementation method requires the function to be in the sum of products (SOP) or product of sums (POS) form. Once the function is available in the required form, the number of input blocks required to implement the function is determined and the functionality is mapped to each of the blocks used. i.e. each block is mapped to implement wither AND logic or OR logic. Once functionality has been mapped to the nanoblocks, the blocks need to be connected to each other through switchblocks and the result is routed to the output of the nanofabric. Here this method is illustrated using a few examples.

Consider a simple function Z1 = AB+C+DE. A total of two AND and two OR gates are needed to implement this function. Consider a fault free nanofabric consisting of 36 nanoblocks and each nanoblock is 3x3. The implementation of this function is shown in Fig. 4. The circles represent switchblocks which are used to connect the output of one block to the input of next one. The rectangles represent the nanoblocks which are configured to implement either an AND or OR gate or are just used for routing purposes. As can be seen, the mapping procedure is relatively simpler in the absence of defective blocks. However, in the presence of defects, it is necessary to avoid the defective blocks and route the result to the output of the nanofabric.

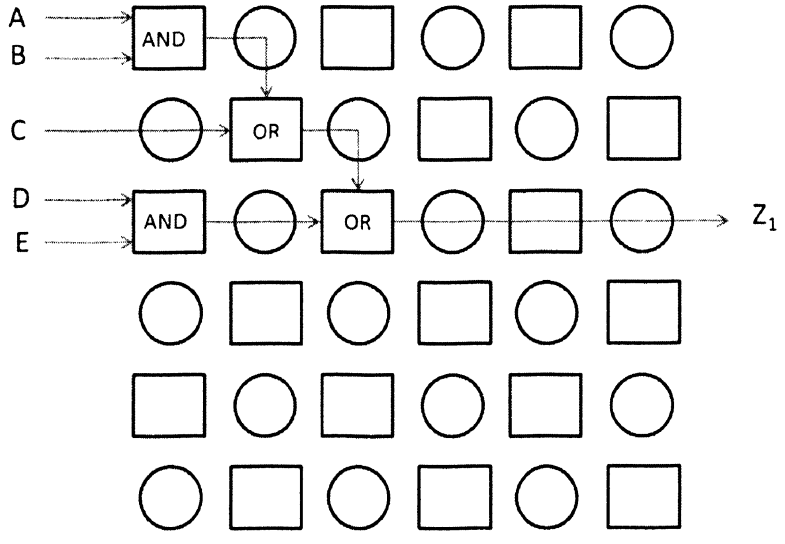


Fig. 4 Logic mapping using AND/OR blocks

Consider a similar nanofabric with a few defective blocks as shown in Fig. 5. Here the defective blocks have been identified by the testing procedure and indicate that they cannot be used to implement any logic. However, in some cases they may be used to transfer signals from one block to the next. This means they can be used for routing purposes. Consider the implementation of a half adder with inputs A, B and outputs sum (S) and carry (C) in the presence of faulty blocks. As can be seen, the faulty blocks need to be avoided to find a fault free path. Thus in the presence of faulty blocks, a number of nanoblocks are used just for routing around the faulty blocks.

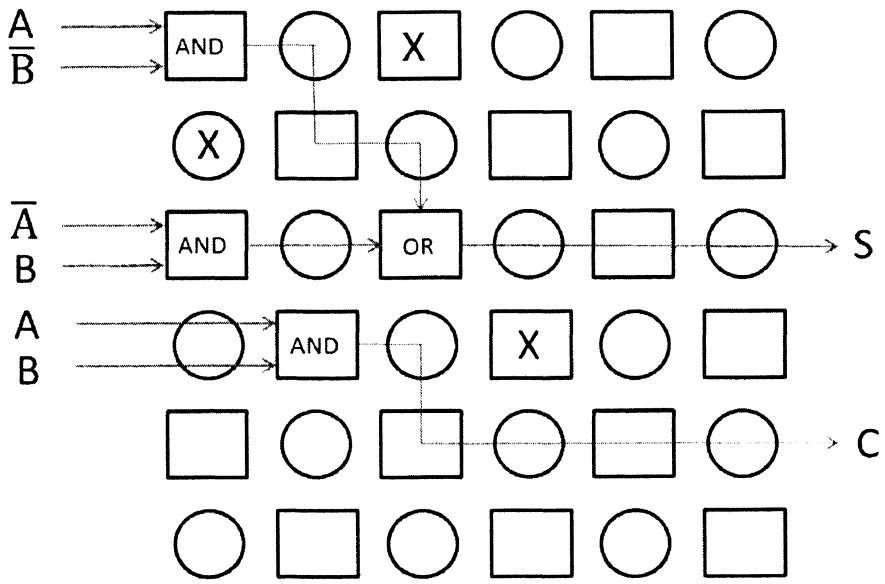


Fig. 5 Half adder implementation in the presence of defects

4. ANALYSIS AND DISCUSSION

In this section, the effect of presence of faults on the proposed technique is analyzed. In the presence of faults, it is required to identify alternate routing around the faulty blocks and to the output. This would increase the complexity of this method depending on the number and location of faults. Sometimes it might not be possible to implement a given function using a set of nanoblocks.

Firstly, a few new terms will be defined and the impact of faults on each of these terms is studied. Consider a nanofabric of fixed depth as shown in Fig. 6. The term utilization in terms of the number of rows required to implement a given function for a given depth of the nanofabric. It is defined as the inverse of the row number (the row at which the output is available). Greater the number of rows required, lesser the utilization.

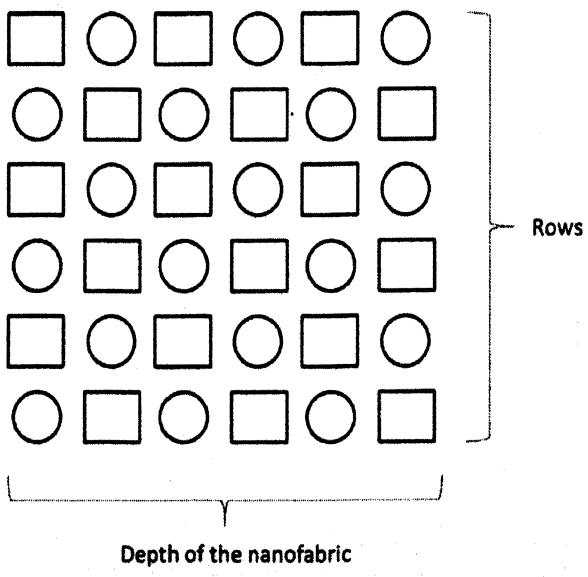


Fig. 6 Sample nanofabric

Next, the term logic density is defined as the number of nanoblocks used to the total number of nanoblocks in the nanofabric. Here the number of nanoblocks used includes the nanoblocks which are used to implement AND/OR logic and the nanoblocks which are simply used to pass the signal to the output. This gives an idea about how much logic can be packed in a given nanofabric. Effective logic density is defined as the ratio of the number of nanoblocks which are configured to implement logic to the total number of nanoblocks used.

Next, consider the location of faulty blocks and how it affects the mapping process. Here, three distinct cases are defined based on the location of faults: faulty input blocks, faulty blocks in the output path and randomly located faulty blocks. This is shown in Fig. 7.

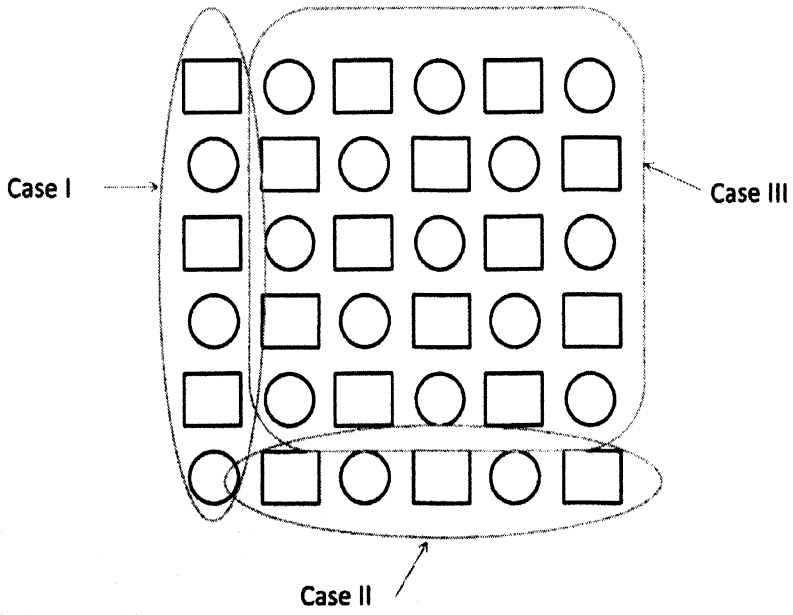


Fig. 7 Location of fault cases

4.1. CASE I: FAULTY INPUT BLOCKS

Input block refers to a block on the input side of the nanofabric. Any faulty block in the input column will shift the output down and increase the output row number by one. This is because it is required to use the next possible input block which is the immediate row below the faulty block. Thus the presence of a faulty block at the input will directly have an impact on utilization.

4.2. CASE II: FAULTY BLOCKS IN THE OUTPUT PATH

Once the functionality has been mapped onto the nanoblocks and the required result obtained, it is necessary to route the result to the output of the nanofabric. This is known as output routing. Here the nanoblocks would be configured to simply pass the signal from one block to the next. And the output path refers to the horizontal path from the block where the result is generated to the output block of the nanofabric. If there are faulty blocks in this path, it is required to move one row down in order to avoid using the faulty block. This again has a direct impact on utilization and increases the output row number by one.

4.3. CASE III: RANDOMLY LOCATED FAULTY BLOCKS

These are the faulty blocks located in the region shown in Fig. 7. These faulty blocks do not have a major impact on the output row number. Since there are always more blocks available to go around the faulty block without increasing the output row number.

For a fixed nanofabric depth, the best possible mapping needs to be determined. i.e. one with the least output row number. This depends on the complexity of the function

to be implemented, the number of input blocks required and the number and location of faults as described above. Here, an expression for the probability that a function can be implemented at a particular row number is derived.

Consider a nanofabric of size N. This means a total of N2/2 nanoblocks and N2/2 switchblocks.

Let p = probability of a faulty nanoblock.

Let i = number of input logic blocks required.

Assume there are m faulty input blocks (Case I), n faulty blocks in the output path (case II), and l randomly located defects (case III). With i input blocks, the best possible mapping that can be obtained is at output row number i. The probability of m input defective blocks out of i is

$$P(m\ input\ defective\ blocks\ out\ of\ i) = \binom{i}{m} p^m\ (1-p)^{i-m} \quad (1)$$

The output will be evaluated at the (i+m)th row and column.

Probability (n def blocks out of N-(i+m)) is

$$P(n\ defective\ blocks\ out\ of\ N-(i+m)) = \binom{N-(i+m)}{n} p^m\ (1-p)^{N-(i+m)-n} \quad (2)$$

The output will be at the $(i+m+n)^{th}$ row.

5. RESULTS

Equations (1) and (2) can be combined and plotted for different values of p, N and i. Figures 8, and 9 gives the probability that a function can be implemented at the required output row number for varying values of the parameters.

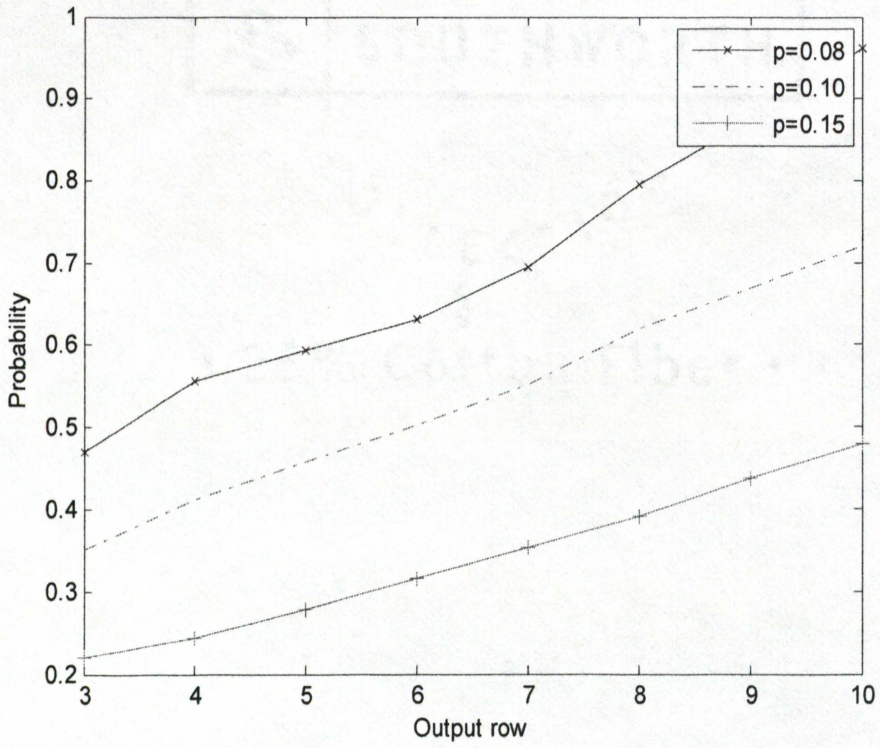


Fig. 8 Output row vs probability plot for variable defect rate, p

Figure 8 shows the plot for different values of defect rates and N=10 and i = 3. It can be seen that as the defect rate increases, the probability reduces. This is because there exist more number of faulty blocks which need to be avoided which increases the output row number. Figure 9 shows the plot for varying number of input blocks for a defect rate of 8% and N=10. The number of input blocks required depends on the complexity of the function to some extent. As the number of input blocks increases, the function becomes more and more complex the output row number also increases for a constant defect rate and nanoblock size.

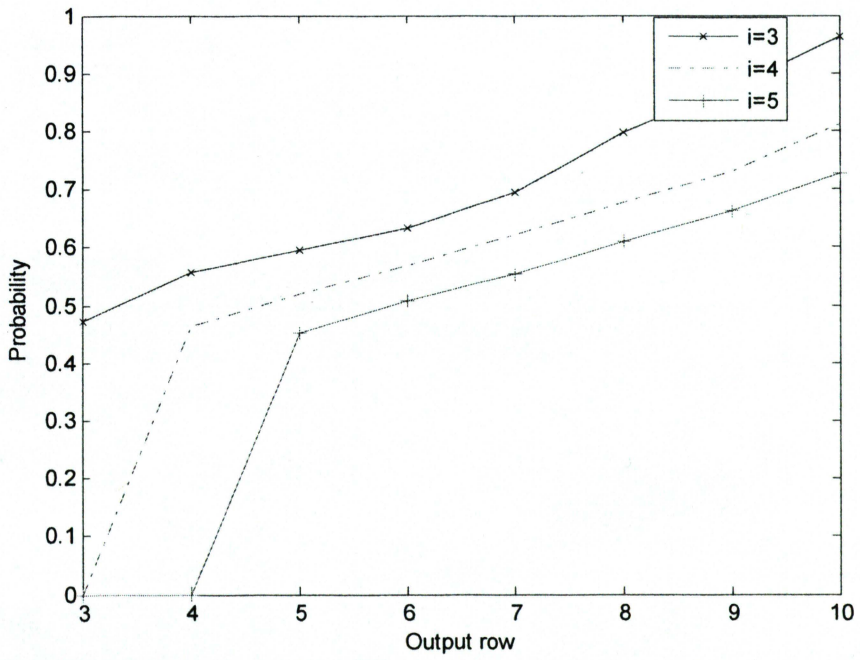


Fig. 9 Output row vs probability plot for variable number of input blocks, i

Logic density and effective logic density are two more terms of importance. Several simple functions were implemented with varying values of i and recorded the

number of blocks utilized to implement a function and in turn the logic density and effective logic density for p = 10% and varying values of m and n. These values have been plotted against the value of i in Fig. 10 and Fig. 11 for logic density and effective logic density respectively. The number of input blocks is a measure of the complexity of the function to be implemented.

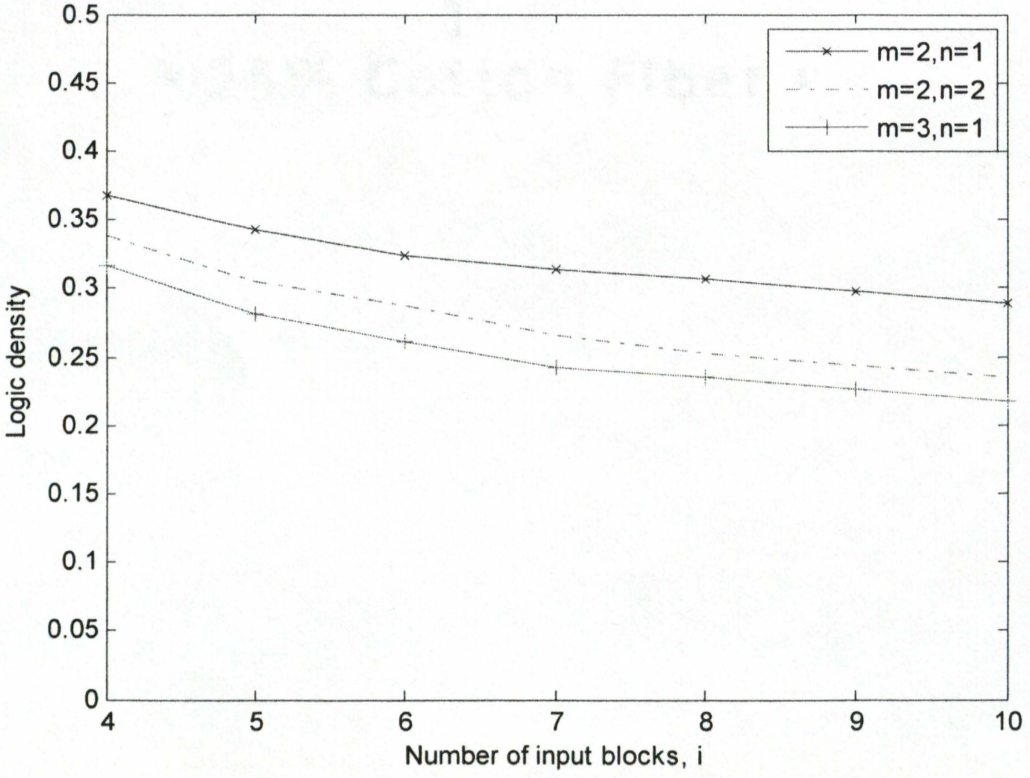


Fig. 10 Logic density vs. input blocks for different values of m,n

From Fig. 10, it can be seen that the logic density reduces with the increase in the complexity of the function. This is because as the output row shifts down, more and more number of blocks are left unused in the upper right of the nanofabric which would otherwise be used to go around any defects. However, the effective logic density is seen to be increasing slightly with I as is evident from Fig. 11. This is because effective logic density is a measure of how many blocks are actually used to implement logic out of the total blocks used. Thus more the number of input bocks, better is the effective logic density. When compared with different values of m an n, greater the values of m and n, lesser the logic density as well as the effective logic density. This is because m and n are case I and case II types of faulty blocks (discussed in Section 4) and have a direct impact on the output.

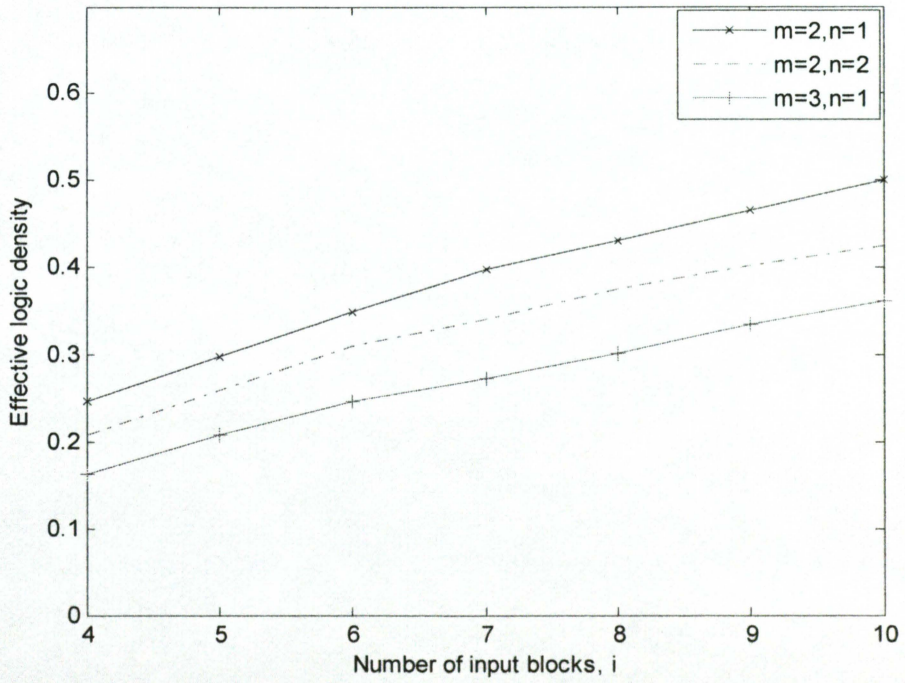


Fig. 11 Effective logic density vs. input blocks for different values of m,n

6. CONCLUSION

This paper presented a novel logic mapping technique for nanofabrics. This technique is based on the nanofabric which assumes standard AND/OR functionality for the nanoblocks. An effective testing approach to obtain such a nanofabric has been discussed in our previous publication. This method is much simpler than the existing ones, since there are standard nanoblocks configurations available for AND/OR. This eliminates the need to program and configure each nanoblock manually thereby simplifying the design process. Also, there is no need to have knowledge about the location of defects within the nanoblock. This reduces the demands on the testing technique used and hence the testing procedure could be simplified as well.It is also illustrated with examples, how several functions can be implemented using this technique in the presence of defects. This paper presents an outline of this new approach and how it can be implemented and future work is planned to do more extensive analysis and test the effectiveness of this approach with the help of performance metrics.

7. REFERENCES

[1] Y. Chen, Z. Wang, "Nanoscale Molecular-switch crossbar circuits," Nanotechnology, 2003, pp: 462-468

[2] M. Butts, X. Weng., "Molecular Electronics: Devices, Systems and Tools for Gigagate, Gigabit Chips," Proceedings ICCAD, 2002, pp: 433-440

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

[4] M. Mishra and S. C. Goldstein, "Scalable Defect Tolerance for Molecular Electronics," workshop on Non-Silicon Computing, 2002

[5] V. V. Zhirnov and D. J. C. Herr, "New frontiers: Self-assembly and nanoelectronics," IEEE Computer, vol. 34, no. 1, pp. 34–43, Jan. 2001

[6] J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams, "A defect-tolerant computer architecture: Opportunities for nanotechnology," Science, vol. 280, no. 5370, pp. 1716–1721, June 1998

[7] Y. Chen, G. Y. Jung, D. A. A. Ohlberg, X. Li, D. R. Stewart, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, and R. S. Williams, "Nanoscale molecular-switch crossbar circuits," Nanotechnology, vol. 14, no. 4, pp. 462–468, Apr. 2003

[8] Z. Wang, and Chakrabarty, K., "Built-in Self-Test of Molecular Electronics-Based Nanofabrics" European Test Symposium, 2005, pp. 168 – 173

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

[10] S. Kundaikar and M. Zawodniok, "Optimized Testing Technique for Defect Tolerance in CAEN-based Nanofabric Systems," to be submitted to the IEEE transactions on nanotechnology

[11] Y. Zheng and C. Huang, "Defect-aware Logic Mapping for Nanowire-based Programmable Logic Arrays via Satisfiability," European Test Symposium, 2005, pp. 189 – 193

[12] A. Bachtold, P. Harley, T. Nakanishi, C. Dekker, "Logic Circuits with Carbon Nanotube Transistors," Science, vol. 294, pp. 1317-1320, 2001

[13] Y. Cui, C. M. Lieber, "Functional Nanoscale Electronics Devices Assembled Using Silicon Nanowire Building Blocks," Science, vol 291, pp. 851-853, 2001

[14] Y. Huang, X. Duan, Y. Cui, L. J. Lauhon, K. H. Kim, C. M. Lieber, "Logic Gates and Computation From Assembled Nanowire Building Blocks", Science, vol 294, pp. 1313-1317, 2001

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

[16] S. Kundaikar, M. Zawodniok, "Optimized Built-In Self-Test Technique for CAEN-based Nanofabric Systems," accepted at IEEE NANO 2011

SECTION

2. CONCLUSIONS

A new testing and design approach was proposed and successfully analyzed. Its performance has been compared with the existing schemes in terms of with respect to a number of parameters including nanoblock size, probability of defect, and nanofabric size. Also, the fault coverage has been shown for a large set of fault types. Theoretical analysis demonstrated that the proposed optimization technique reduces the number of required configurations and testing time while increasing the effective yield of a nanofabric. A reduction of 23% - 55% is obtained in the number of required configurations, which also leads to reduced testing time. Additionally, a customization approach was presented. It further improves the testing time if the specific logic function is known a priori and the yield of the nanofabric could be increased to as high as 96%. Moreover, the testing has been accompanied with a new logic mapping technique that complements each other. The defect map generated by testing procedure is utilized to efficiently map logic functions onto the nanofabric such that partially defective, but functional, nanoblocks are utilized instead of discarding. The main benefit of the proposed approach is simplification of the logic mapping process by using standard nanoblock configurations. The location of defects inside each nanoblock does not have to be known thus avoiding a complex and time consuming testing process.

VITA

Sambhav Dilip Kundaikar was born in Panaji-Goa, India, on August 06, 1986. In July 2007, he received his Bachelor of Engineering with Distinction Honors in Electronics and Telecommunications Engineering from Goa University. He joined Cognizant Technology Solutions in November 2007 as a Programmer Analyst and worked in the field of PeopleSoft-ERP for 20 months. He started his Master of Science program in Computer Engineering at Missouri University of Science and Technology in August 2009. He worked as a research assistant under the guidance of Dr. Maciej Zawodniok from January 2010 to July 2011. He graduated in December 2011.