Spring 2009

Functional testing of faults in asynchronous crossbar architecture

Sriram Venkateswaran

Department:

FUNCTIONAL TESTING OF FAULTS IN ASYNCHRONOUS CROSSBAR ARCHITECTURE

by

SRIRAM VENKATESWARAN

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

2009

Approved by:

Minsu Choi, Advisor

Sahra Sedigh

Theodore McCracken

PUBLICATION THESIS OPTION

This thesis consists of three articles that have been published or submitted for publication. The details of the same are as follows:

The first paper presented in pages 2-41 entitled “Functional Testing of Faults in Asynchronous Crossbar Architecture” is a journal article which will be shortly submitted to ACM Journal of Emerging Technologies in Computing Systems (JETC).

The second paper presented in pages 42-54 entitled “Post-Configuration of Asynchronous Nanowire Crossbar Architecture” was published in the IEEE International Conference on Nanotechnology, 2008.

The third paper presented in pages 55-70 entitled “Novel Functional Testing Technique for Asynchronous Nanowire Crossbar System” has been submitted to the IEEE International Instrumentation and Measurement Technology Conference, 2009.

ABSTRACT

The challenge of extending Moore's Law past the physical limits of the present semiconductor technology calls for novel innovations. Several novel nanotechnologies are being proposed as an alternative to their CMOS counterparts, with nanowire crossbar being one of the most promising paradigms. Quite recently, a new promising clock-free architecture, called the Asynchronous Crossbar Architecture has been proposed to enhance the manufacturability and to improve the robustness of digital circuits by removing various timing related failure modes.

Even though the proposed clock-free architecture offers several merits, it is not free from the high defect rates induced due to nondeterministic nanoscale assembly. In this work, a unique Functional Test Algorithm (FTA) has been proposed and validated to test for manufacturing defects in this architecture. The proposed Functional Test Algorithm is aimed at reducing the testing overhead in terms of the time and space complexity associated with the existing sequential test scheme. In addition, it is designed to provide high fault coverage and excellent fault-tolerance via post-reconfiguration. This test scheme can be effectively used to assure true functionality of any threshold gate realized on a given PGMB. The main motivation behind this research is to propose a comprehensive test scheme which can achieve sufficiently high test coverage with acceptable test overhead. This test algorithm is a significant effort towards viable nanoscale computation.

This work has been organized into three papers, explaining the proposed algorithm, demonstrating its working, describing the achievable replacement schemes using the proposed tool and providing a performance evaluation metric specifically proposed to evaluate the functional test algorithm.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Almighty God for providing me with all the good health, strength and knowledge to complete my thesis and masters.

I would like to express sincere gratitude and heartfelt thanks to my advisor Dr. Minsu Choi for his guidance, encouragement and invaluable contributions throughout my graduate study as well as in completing this research. I would also like to thank him for providing me with financial support during my master's study.

To my committee members, Dr. Sahra Sedigh and Dr. Theodore McCracken, I extend my deepest appreciation and thanks for their time and effort in serving as committee members and reviewing this dissertation. My sincere thanks to Dr. V A Samaranayake of Mathematics and Statistics department for his important inputs.

Special thanks to Ravi Bonam and Shikha Chaudhary, senior research members of Dr.Choi's research group for their help and ideas. Thanks go to my friends and roommates for all the happy times and memorable moments we shared.

Lastly, but most importantly, I am indebted to my parents, my mother aunt for their unparallel love and patience with me during my endeavors. Without special support and motivation from my uncles, aunt and love from my brothers, Srihari, Varun, Karthik and young sister Vaishnavi, grandparents and all my close relatives and friends back in India, I would not have been able to reach this stage of life.

TABLE OF CONTENTS

PAGE

LIST OF ILLUSTRATIONS

Page

PAPER I

PAPER II

PAPER III

LIST OF TABLES

Page

PAPER I

PAPER II

PAPER III

INTRODUCTION

The future electronic systems face a challenge to adopt to novel nanoelectronic solutions in order to ensure that Moore's Law successfully extends past the physical barriers of the present semiconductor technology. A clockless nanowire structure, called the asynchronous crossbar architecture has been quite recently proposed as an improvement over its clocked counterparts. However, in order to be a viable technological paradigm, several intrinsic issues associated with nanowire crossbar architecture such as imperfect nanoscale fabrication needs to be addressed.

This thesis spotlights the dawn of a new test algorithm, called the Functional Test Algorithm. It is an extremely significant improvement over the previously existing raw testing scheme. The proposed algorithm uses input test tuple set unique to the function being realized. The algorithm identifies unique crosspoint locations specific to each threshold gate. Not only does the proposed algorithm provide complete test coverage, but it also manages to provide excellent fault-tolerance.

The proposed Functional Test Algorithm has been explained in this work in the form of three articles. The test algorithm has been clearly explained with several suitable examples. The usefulness of this algorithm in achieving perfect realization of any threshold gate on a programmable gate macro block (PGMB) has been demonstrated. In addition, a performance evaluation metric has been designed to evaluate the effectiveness of this scheme. This test algorithm can be used in future as a viable diagnostic tool to identify fabrication defects induced due to imperfect assembly. Parametric simulation using MATLAB have been done to validate the results.

I. FUNCTIONAL TESTING OF ASYNCHRONOUS NANOWIRE CROSSBAR ARCHITECTURE

Sriram Venkateswaran and Minsu Choi

Dept of ECE, Missouri University of Science and Technology

Rolla, MO, USA

{svf44, choim}@mst.edu

ABSTRACT

Of late, several novel nanotechnologies are being proposed as an alternative to their CMOS counterparts, with nanowire crossbar being one of the most promising paradigms. Quite recently, a new promising architecture, called the Asynchronous Crossbar Architecture has been proposed. This proposed asynchronous nanowire clock-free crossbar architecture is envisioned to enhance the manufacturability and to improve the robustness of digital circuits by removing various timing- related failure modes. Inspite of being advantageous over the clocked architectures, the asynchronous crossbar architectures are still not free from high defect rates induced by nondeterministic nanoscale assembly. In order to address this problem, there is a burning need to develop an effective test mechanism. In this paper, a novel Functional Test Algorithm has been proposed to achieve effective mapping of threshold gates onto a given Programmable Gate Macro Block (PGMB). The proposed algorithm tests only the relevant crosspoints programmed as ON using input patterns unique to the given threshold gate macro. This test scheme can be used to assure true functionality of any threshold gate realized on a given PGMB. In addition, this test scheme also provides excellent fault-tolerance and

fault-coverage. Parametric simulation results using MATLAB have been used to show the performance of this testing scheme.

Index Terms-Asynchronous nanowire crossbar system; Functional test algorithm; Programmable Gate Macro Block (PGMB); Defect and fault-tolerance; Parametric simulation.

1. INTRODUCTION

Future electronic systems face a challenge to adopt to novel nanoelectronic solutions in order to ensure that Moore's Law successfully extends past the physical barriers of the present semiconductor technology. Most of the new nanoelectronic technologies present excellent potential for unexampled levels of device density, low power computing and high operating speeds. One of the most common paradigms for nanoelectronics is a crossbar based architecture [1], a two dimensional array formed by intersection of two orthogonal sets of parallel and uniformly spaced nanometer-sized wires. The crossing over of these nanowires forms programmable junctions called crosspoints [2, 3, 4, 5]. Experiments have shown that such wires can be aligned to construct an array with nanometer-scale spacing using a form of directed self-assembly. This set of crosspoints of nanoscale wires can be used as programmable diodes, memory cells or FETs (Field-Effect Transistors); thus making it possible to realize nanoscale logic devices [6].

In order to be a viable nanotechnology, nanowire based systems should be:

1. Structurally simple and scalable enough to be fabricated by bottom-up manufacturing technique.

2. Robust enough to tolerate extreme parametric variations.
3. Defect and fault-tolerant enough to overcome defect densities, aging factors and transient faults.
4. Able to support at-speed verification and reconfiguration.

Recently, an asynchronous nanowire crossbar architecture based on delay-insensitive data encoding and self timed logic encoding scheme has been proposed [6]. This proposed architecture being totally clock-free, no clock distribution network is needed.

The biggest challenge however, lies in making these nanoscale structures simple enough to be manufactured and reliable enough to be used in computing applications. Since the nanoscale structures are assembled in a bottom-up manner, they are likely to have much higher fabrication defect densities and parametric variations [7, 8]. The clock-free asynchronous nanowire crossbar architectures have several merits over their clocked counterparts; they are still not free from high defect rates induced due to nondeterministic assembly [9]. The primitive form of testing these defects is to check each location individually for defect.

With an aim to address this issue, a unique testing algorithm has been proposed in this paper. The proposed Functional Test Algorithm (FTA) can be used to test asynchronous nanowire crossbar structures for defect. In this paper, the functional test algorithm has been proposed. The features and advantages of using this algorithm have been discussed and analyzed using numerous examples in the proceeding sections of the paper. Section 4 explains the FTA thoroughly. Section 5 expands on this and explains

how the proposed algorithm can be used to achieve fault-tolerance. Section 6 provides a performance analysis model and also defines a set of performance analyzers to quantify the effectiveness of the functional test algorithm.

2. PRELIMINARIES AND REVIEW

2.1 NULL CONVENTIONAL LOGIC

Traditional Boolean circuits exhibit time dependent relationships as well as symbolic- value-dependent relationships [4]. Time dependent relationships depend upon propagation delay times required to express validity of data values. Symbolic-value-dependent relationships depend upon interconnection of logic gates and their truth tables. Most traditional boolean circuits are clock driven. These circuits are symbolically incomplete in terms of evaluating expressions as they are dependent on the clock.

NULL Convention Logic (NCL) [10, 11, 12] is complete in terms of theory and is also feasible in terms of implementation and economics as compared to delay insensitive circuits. NCL logic makes use of two signals, DATA and NULL. DATA signal represents the data signal used by the combinational circuit. NULL represents synchronization and I/O control. It is used to reset the gates in the combinational circuit. These circuits use dual-rail or quad-rail logic to achieve delay insensitivity.

Figure 1 shows the framework for NCL systems. In the DATA evaluation period, the combinational circuitry processes the data passed on by the register. The results are stored in the successive register. The successive register generates the Request for NULL signal in the DATA completion Acknowledgement period and propagates the signal to the previous register. The previous register transfers a NULL to the combinational

circuitry evaluated during the NULL combinational evaluation period. The evaluated result is passed to the successive register which generates a Request for DATA signal. The DATA to DATA timing diagram is shown in figure 2.

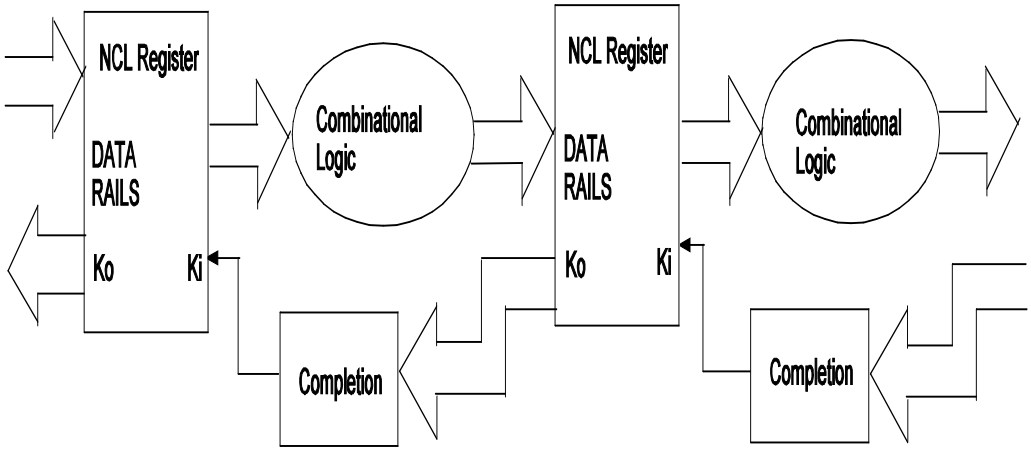

Figure 1 Pipelined NCL

Ko and Ki signals are connected between the registers to synchronize the operation of the cumulative circuit. If the output of a particular gate is NULL, it does not change until and unless all the inputs to the gate are DATA. The dual rail- encoding scheme used in NCL architecture is described effectively in table 1.

Table 1 Dual Rail Encoding Scheme

Dual Rail Encoding Scheme

Rail 1	Rail 0	Represented State	DATA Value
0	0	NULL	--
0	1	DATA	0
1	0	DATA	1
1	1	UNDEFINED	--

When all the inputs receive DATA then the output changes to data and remains asserted as long as all the inputs do not change to NULL. This attribute of the threshold gates helps in achieving the completeness feature enabling the circuits to function without the clock [11]. To achieve this property, a dual rail encoding scheme is used, as shown in table 1. NCL uses symbolic completeness [12] of expression to achieve self-timed behavior.

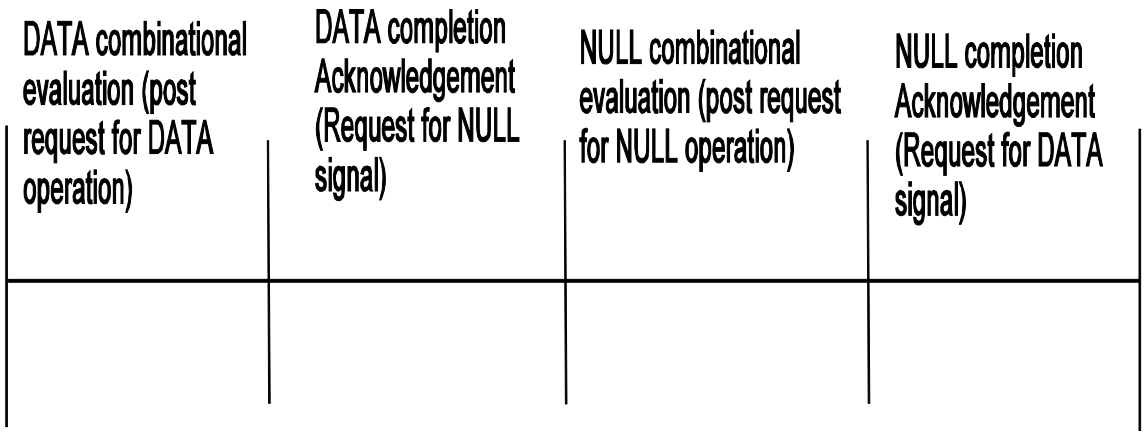

Figure 2 Timing Diagram

The main advantages of using NCL are as follows [10]:

Ease of Design:

NCL circuits are self completed circuits in that their operation does not involve any clock signals for synchronization. They do not use any external trigger, clock or controller to accept data values or express readiness of circuit. NCL circuits can be fully expressed in high level languages. In addition, since the system is independent of clocks, the logic can

be designed in parts which can be directly composed later. The issues associated with global synchronization are totally eliminated.

Lower Power Consumption:

The NCL systems operate in terms of synchronized wave fronts of monotonic level transitions. There are no pulses or edge triggering involved in the circuit behavior. The NULL state used here is an idle power state. The cumulative power consumption is significantly lower than that of clock driven circuits [10].

Convenient Technology Migration:

NCL is insensitive to the behavioral properties of the physical implementation. The NCL circuits are insensitive to implementation technology, scale changes and propagation delay changes due to aging [10].

Adaptability to Physical Properties:

Since NCL is delay insensitive, the delays due to changes in physical parameters like temperature, manufacturing variations, voltage do not have an effect on these circuits. These circuits continue to operate correctly under these variations.

Operation Speed:

Although NCL cycles require two propagation cycles per unit of processing, there are no delay margins added to account for the propagation delay as in case of clocked circuits. Integration of the registration in logic gates allows more finely grained pipelining and consequently higher throughput rates than conventional clocked techniques [10].

A total of 27 Threshold gates are implemented in NCL [11]. The importance of the 27 threshold gates is that any possible expression having a maximum of four variables can be implemented using these functions. Inversion can be implemented by interchanging the rail 1 and rail 0 in case of a dual rail encoding scheme. The basic PGMB block is shown in figure 3.

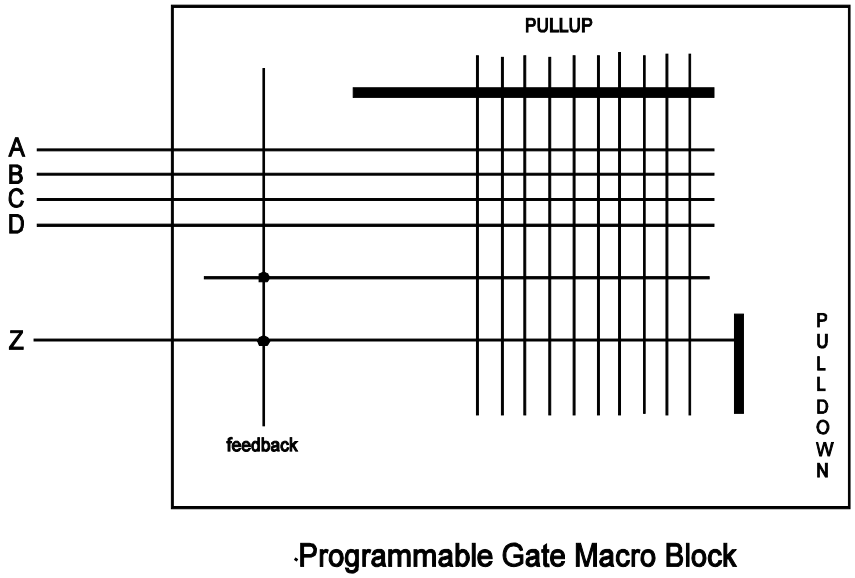

Figure 3 Programmable Gate Macro Block

2.2 ASYNCHRONOUS CROSSBAR ARCHITECTURE

The normal crossbar architecture will be similar to the conventional clocked circuits. Synchronization in this conventional crossbar architecture will be provided by the clock which circulates throughout the circuit and helps decide when to receive and release data. Compared to the clocked counterparts, the asynchronous crossbar

architecture was proposed to be data driven [6]. This architecture employs threshold gates [11] that recognize only certain simultaneous combinations of values unique to each gate. List of all threshold gates is provided in table 2. A total of 27 threshold gates are listed in table 2.

Table 2 List of all Threshold gates with their functional expressions

NCL Macros

NCL Macros	Boolean Function
TH12	A + B
TH22	AB
TH13	A + B + C
TH23	AB + AC + BC
TH33	ABC
TH23w2	A + BC
TH33w2	AB + AC
TH14	A + B + C + D
TH24	AB + AC + AD + BC + BD + CD
TH34	ABC + ABD + ACD + BCD
TH44	ABCD
TH24w2	A + BC + BD + CD
TH34w2	AB + AC + AD + BCD
TH44w2	ABC + ABD + ACD

Table 2 List of all Threshold gates with their functional expressions (cont'd)

TH34w3	A + BCD
TH44w3	AB + AC + AD
TH24w22	A + B + CD
TH34w22	AB + AC + AD + BC + BD
TH44w22	AB + ACD + BCD
TH54w22	ABC + ABD
TH34w32	A + BC + BD
TH54w32	AB + ACD
TH44w322	AB + AC + AD + BC
TH54w322	AB + BC + BCD
THxor0	AB + CD
THand0	AB + BC + AD
TH24comp	AC + BC + AD + BD

NCL circuit being data driven, each of these gates acts as a "synchronization node" and makes the circuit symbolically complete. This completeness is achieved as follows: The DATA state follows the Null state and is processed by the gates and output is passed on to a register. The register contains completion circuitry that enables synchronization and checks the state of the output and generates an appropriate signal indicating the previous register to send the complementary state i.e. if the circuit is processing a Null state then the register on arrival of the output will send a request for data signal requesting for data to the previous register. The notable advantages of this architecture are [6]:

1. Manufacturability: Absence of clock would mean all clock related circuits can be removed from the design. This would make the overall hardware design easier and

less complex. As compared to their clocked counterparts, these circuits would be easy to manufacture.

2. Scalability: Since timing information is integrated with data in encoding, the timing complexity remains the same irrespective of the size of the circuit.
3. Robustness: Due to non-determinism of the directed self-assembly paradigm, nanowire crossbar circuits are anticipated to exhibit large variations in physical parameters. Since any physical variation in an electrical parameter may have its own negative effect on the timing behavior of the circuit, being able to design delay- insensitive circuits (i.e., correct operation of the circuit is independent of the timing) is a significant capability and it would greatly increase the robustness of the circuit to design parameter variations. As explained in Null Conventional logic subsection, there is no delay in processing data due to clock cycles as in clocked synchronous circuits. Instead data would be processed as and when it is available.
4. Defect and Fault Tolerance: As NCL circuits have a definite flow pattern i.e. DATA or NULL and vice versa the output can be checked if it is a data or null. In addition to the complete removal of all timing-related failure modes, testing complexity is reduced in that stuck-at-1 faults simply halt the circuit, since the NCL circuit cannot make a transition from DATA to NULL. Also, in case of dual-rail encoding, 11 is considered an invalid code. So, any permanent or transient fault that results in this invalid codeword can be eventually detected. Only stuck-at-0 faults and some other transient faults need to be exercised with applied patterns. Design time and risk as well as circuit testing requirements are expected to decrease because of elimination of the clock and its critical timing issues.

The basic unit of the asynchronous nanowire crossbar architecture is the programmable gate macro block (PGMB). A typical PGMB is as shown in figure 3. The PGMB has AND and OR crossplanes formed by diode crossbars. The dimensions of the PGMB can be adjusted according to the efficiency of programming and manufacturing defect rate. The vertical wires with pull up resistors form the product terms and the horizontal wires with pull down resistors form the OR logic. There is also a feedback logic incorporated. It has been demonstrated that each of the threshold gates can be realized on a defect free PGMB having 6 rows and 10 columns [6, 13].

NCL (Null Conventional Logic) a delay insensitive paradigm, which helps in eliminating the clock from the circuit, can be implemented on nanowire crossbar architecture to realize asynchronous crossbar architecture. Table 3 gives the truth table for few THmn gates. With a total of n inputs atleast m out of n are needed for assertion. All signals deasserted is the reset condition. In case the weights of inputs are not specified, the default value is 1. With TH33w2 gate, the weight of the higher order bit input bit (i.e. bit A in ABC input pattern) is 2 and that of B and C is 1 respectively. Each gate has a boolean expression that gives its functionality. In case of TH24 gate, the functionality expression is $F = AB + BC + AC + AD + BD + CD$ +F' $(A + B + C + D)$ where F' represents the output feedback. The terms F'(A + B + C + D) account for the hysteresis behavior. Once the output is asserted, the only way to get it back to zero is to reset all primary inputs.

Table 3 Truth Table for THmn gates

Truth Table for TH gates (F' represents previously asserted output)

ABCD	TH23	TH24	TH34	TH33w2	TH44w3
0 0 0 0	0	0	0	0	0
1 0 0 0	F'	F'	F'	F'	F'
0 1 0 0	F'	F'	F'	F'	F'
1 1 0 0	1	1	F'	F'	F'
0 0 1 0	F'	F'	F'	F'	F'
1 0 1 0	1	1	F'	1	F'
0 1 1 0	1	1	F'	1	F'
1 1 1 0	1	1	1	1	1
0 0 0 1	-	F'	F'	-	F'
1 0 0 1	-	1	F'	-	1
0 1 0 1	-	1	F'	-	1
1 1 0 1	-	1	1	-	1
0 0 1 1	-	1	F'	-	1
1 0 1 1	-	1	1	-	1
0 1 1 1	-	1	1	-	1
1 1 1 1	-	1	1	-	1

3. PROBLEM DESCRIPTION AND PROPOSED SOLUTION

The nanoscale structures are assembled in a bottom-up manner and are hence likely to have much higher fabrication defect densities and parametric variations [7, 8] as compared to those which use a top-down fabrication approach. Unfortunately, the current fabrication methods have not been able to manufacture a defect-free nanowire crossbar matrix. According to researchers, current fabrication processes have defect rates of about

10% [14, 15] in a nanowire crossbar. Scientists are yet to discover a standard fabrication technique which would have a consistent defect rate. Each threshold gate that is programmed on a PGMB has a predefined pattern of crosspoint placement. This mapping pattern gives the corresponding functionality of the threshold gate. The crosspoint placement locations are unique for each threshold gate. Due to manufacturing defects, some of these ON programmable crosspoints may not be programmable. Such a manufacturing defect may result in a stuck- at-OFF fault at a programmable location. A crosspoint location having a stuck-at-OFF fault cannot be programmed as ON. In a 6x10 grid used to implement TH23 gate, 18 out of the total available 60 potentially programmable crosspoints are used to implement the TH23 gate macro. In figure 4, a defect at the leftmost crosspoint in the first row results in a faulty function $F^{¤} = B + BC + AC + AF^{0} + BF^{0} + CF^{0}$. One or more of such faults can completely alter the functionality of the THmn gate being realized. This results in a need for functionally testing the programmable PGMB after gate mapping.

The most primitive form of testing PGMBs is the raw testing scheme. In this test scheme, each and every crosspoint is tested for ON and OFF state separately. This is an extremely laborious method and introduces a great amount of overhead [1]. In case a 6 x 10 grid has to be tested, each of the 60 crosspoints will have to be individually tested. Another drawback of this scheme is that although a crosspoint is tested for ON/OFF state, there is still no guarantee that it is completely programmable.

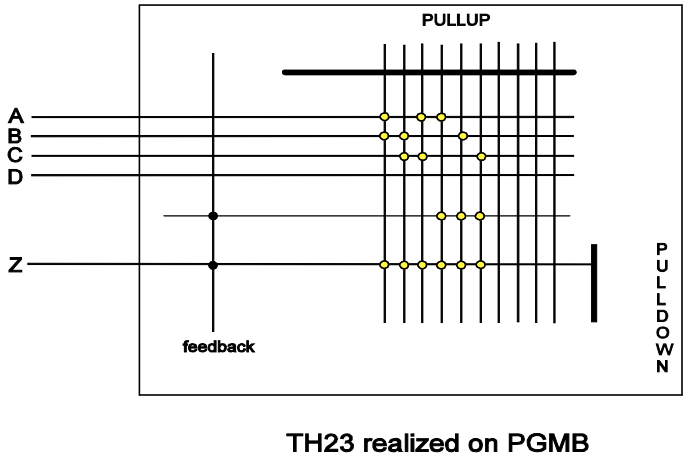

Figure 4 TH23 gate implemented on a PGMB

The raw testing scheme cannot provide complete assurance that the threshold gate is functionally correct. In addition to these reasons, the testing overheads introduced in terms of time and space complexities call for a more reliable and practical form of testing the PGMBs. The prime motivation behind proposing the Functional Test Algorithm is to propose a test scheme which will address the issues associated with raw testing of PGMBs. By addressing the stuck-at-0 faults using applied input patterns, this novel test scheme provides a realistic solution to solve the current problem. The features and advantages of using the proposed functional test algorithm are discussed and illustrated with numerous examples in the proceeding sections of the paper.

4. FUNCTIONAL TEST ALGORITHM

The proposed functional test algorithm is a post configuration test scheme [16] that makes use of the boolean function of the threshold gate being implemented to test the programmable ON crosspoint locations on the PGMB. Each THmn gate has its own unique and distinctive ON programmable co-ordinate locations. The proposed test

scheme aims to test only those programmable ON crosspoints in the given programmable PGMB. This algorithm uses the functional expression that is unique to each THmn gate. As shown in figure 4 there are 18 programmable locations in the 6 x 10 grid. The functional test scheme uses "test tuples" for the purpose of testing the programmable crosspoints. Test tuples are joint combinations of input bit patterns and previously asserted output. Table 4 can be used to clearly understand this. Consider the implementation of TH23 gate. Assume there is a fault at the coordinate location (1, 3). The fault at this ON point gives a faulty output F*=1 when input 001 is used. The desired output in case there is no fault at any crosspoint is F=F'(the previously asserted output). In case the previously asserted output is set to 0 and then followed with an input pattern 001, an erroneous output of 1 will be obtained. By using a combination of F' and input bits, faults at the ON programmable crosspoints can be detected. The set of inputs used to detect the faulty crosspoints are called "test tuples". On close examination of table 4, it can be noticed that a single test tuple can be used to determine correctness or fault at multiple locations. In other words, certain test patterns have one-to-many correspondence with programmable ON crosspoint locations. Input bits 001 with F'=0 can be used to test crosspoints having coordinates (1, 3), (2, 2), (5, 5). Input bits 010 with F'=0 can be used to test ON crosspoints having co-ordinate locations (1, 1), (3, 2), (5, 5). A total of 10 test tuples are needed to test all the 18 programmable ON crosspoints of TH23 gate macro.

Table 4 Truth Table for TH23 gate and all faulty functions that can be resulted from single crosspoint defect (F' is previously asserted output and (i, j) are defective crosspoint coordinates). Faulty outputs that can be used to test are highlighted.

ABC	F	F* (1,1)	F* (1,3)	F* (1,4)	F* (2,1)	F* (2,2)	F* (2,5)	F* (3,2)	F* (3,3)	F* (3,6)
0 0 0	0	0	0	**F'=1**	0	0	**F'=1**	0	0	**F'=1**
0 0 1	F'	F'	**1**	F'	F'	**1**	F'	F'	F'	F'
0 1 0	F'	**1**	F'	F'	F'	F'	F'	**1**	F'	F'
0 1 1	1	1	1	1	1	1	1	1	1	1
1 0 0	F'	F'	F'	F'	**1**	F'	F'	F'	**1**	F'
1 0 1	1	1	1	1	1	1	1	1	1	1
1 1 0	1	1	1	1	1	1	1	1	1	1
1 1 1	1	1	1	1	1	1	1	1	1	1

ABC	F	F* (5,4)	F* (5,5)	F* (5,6)	F* (6,1)	F* (6,2)	F* (6,3)	F* (6,4)	F* (6,5)	F* (6,6)
0 0 0	0	0	0	0	0	0	0	0	0	0
0 0 1	F'	F'	F'	**1**	F'	F'	F'	F'	F'	**0**
0 1 0	F'	F'	**1**	F'	F'	F'	F'	F'	**0**	F'
0 1 1	1	1	1	1	1	**F'=0**	1	1	1	1
1 0 0	F'	**1**	F'	F'	F'	F'	F'	**0**	F'	F'
1 0 1	1	1	1	1	1	1	**F'=0**	1	1	1
1 1 0	1	1	1	1	**F'=0**	1	1	1	1	1
1 1 1	1	1	1	1	1	1	1	1	1	1

Based on the number of points tested by each tuple, they can be prioritized as higher and lower order test tuples. Test tuples having one to one correspondence with the programmable crosspoints are called lower order test tuples. Higher order test tuples can test for defects at more than a single crosspoint location simultaneously. Table 5 shows the number of crosspoints tested by using test tuples for a set of THmn gates. The TH23 gate has only two priority levels as shown in figure 5.

Table 5 Table giving tested crosspoint coverage with respect to total Non crosspoints.

		Prioritized Test Tuple (TT) Count				
Gate	*Non*	3 TTs	4 TTs	5 TTs	6 TTs	7 TTs
TH23	18	50%	66.67%	72.22%	77.8%	83.3%
TH24	30	40%	53.33%	66.67%	70.0%	73.3%
TH34	28	28.57%	35.71%	42,86%	50.0%	57.17%
TH33w2	15	53.3%	66.67%	73.3%	80.0%	86.67%
TH44w3	21	47.6%	57.14%	66.67%	71.42%	76.19%

The first set of 4 test tuples each cover 3 crosspoints. The total coverage provided by the first set is 12 crosspoints. The remaining 6 test tuples each cover only 1 crosspoint and are placed in the lowest level of priority. TH23 has 2 priority levels. TH34, on the other hand, has 3 input priority levels with the first tuple testing 4 ON-crosspoints, the second set testing 2 points each (test tuple number 2, 3, and 4 test 3 crosspoints each) and finally the lowest level providing direct correspondence.

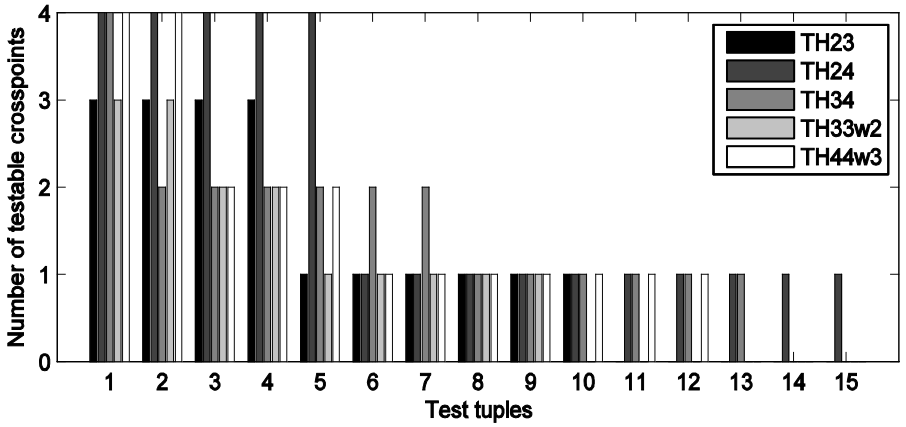

Figure 5 Testable crosspoints with each input for THmn Gates

The proposed functional test scheme applies input tuples in the order of their priority level and validates outputs from those input tuples. For TH23 gate, the first set of 4 Test Tuples test 12 out of the 18 possible programmable ON locations. In case of the

TH23 gate, using the first 3 most highly ranked test tuples cover 50% of the total test space. Using another input increases this to 66.67%. This rate rises to 72.2, 77.8 and 83.3% respectively with each additional input. To achieve total testability for TH23 gate, 10 test tuples need to be applied. In similar fashion, a total of 15 test tuples need to be applied to achieve total testability for TH24 gate.

Pseudo Code for the functional test algorithm is described in figure 6. The following illustrative examples will help understand the working of functional test protocol.

```
START
1. Define PGMB dimensions and threshold gate
   to be programmed.
2. Map THmn gate onto the PGMB and generate
   the corresponding truth table.
3. IF   OR plane defect detection is the prime
        objective  THEN
        Set test tuples with low priority tuples
        having direct correspondence with
        programmable crosspoints to be used
        ahead of the high priority ones.
   ELSE
        Set test tuples in order of decreasing
        priority levels.
4. Use next test input tuple to check for
   programmable crosspoint defect.
5. IF    False output is observed  THEN
        Increment fault count and note possible
        defect location.
   ELSE
        The tested location is defect free.
6. Go to Step 4.
7. Summarize the results of the test.
STOP
```

Figure 6 Pseudo Code for Functional Test Algorithm

Case 1:

Consider figure 7. The smaller dots represent programmable locations for TH23 gate. The bold circles represent the randomly present defective crosspoints on the PGMB. The defect rate considered here is 10%. The locations of these defects are not known prior to mapping of the TH23 gate. They have been shown in the figure for easy understanding of the concept.

The functional test algorithm works as follows:

1. The TH23 gate is mapped on to the given PGMB.
2. The set of prioritized test tuples are generated for TH23 gate.
3. Check co-ordinate locations (1, 1), (3, 2) and (5, 5) for defect using the first test tuple (000, 010).
4. As no faulty outputs are generated, all 3 points are cleared of having any defect and are tested good.
5. The next test tuple (000, 001) tests locations (1, 3), (2, 2) and (5, 6). No faulty output is observed.
6. Steps 4 and 5 are repeated till all the 18 locations have been tested.
7. Since no undesirable outputs are observed, the TH23 gate has been perfectly realized.

In this case, the TH23 gate is 100 % programmable since none of the defective locations coincide with the programmable locations. FTA looks for defects only at programmable locations. Defects can co-exist at non programmable locations without being located and identified. This allowance can be provided since the defective locations do not alter the functionality being realized.

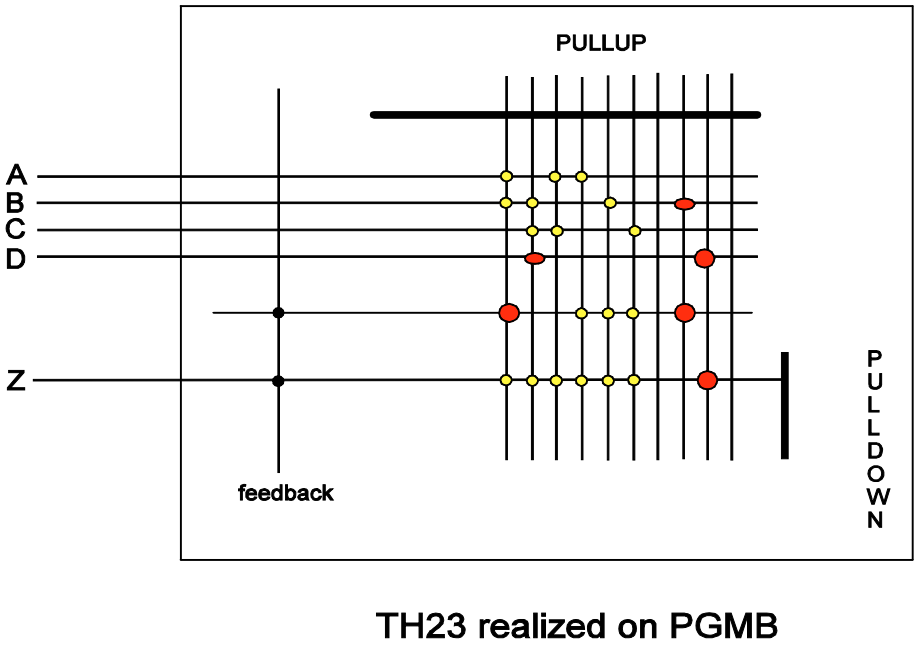

Figure 7 TH23 gate realized on a PGMB having 10% defect rate

Case 2:

Consider figure 8 which shows TH34w2 gate mapped onto a defective PGMB. The defect rate considered is 10 % for the worst case scenario. The circles indicate the programmable locations for the THmn gate which must be programmed as ON. The stars denote programmable ON crosspoint locations overlapping the defective crosspoint locations. The other points marked as X in the figure show defective crosspoint locations which will not be programmed for realizing TH34w2 gate. These locations will not alter the functional behavior since they do not overlap with the ON programmable crosspoint locations. The proposed functional test algorithm will work as follows in this case:

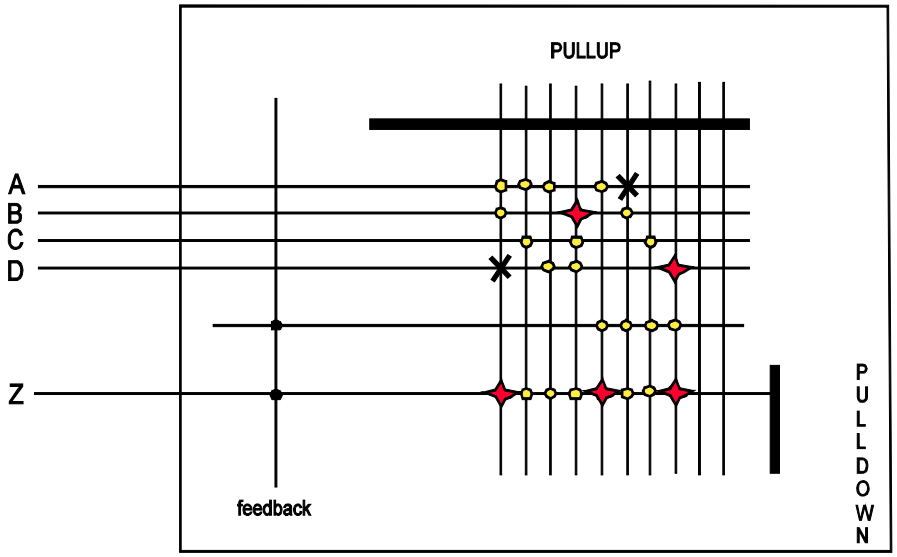

Figure 8 TH34w2 gate implemented on a PGMB having 10% defect rate

1. TH34w2 is mapped on to the given PGMB.
2. The set of prioritized inputs are generated.
3. The first set of prioritized test tuple (0000, 0100) tests for locations (2, 1), (3, 2), (4, 3) and (5, 5). The first part in the tuple set 0000 is used to prepare the PGMB for testing and 0100 is the input pattern used to test the crosspoint location.
4. The second prioritized test tuple (1111, 0000) tests (1, 5), (2, 6), (3, 7) and (4, 8). This time, the observed output is different from the desired one. This implies there is a fault at either one or more crosspoints from the set of 4 locations tested.
5. The third set (0000, 0010) is then used to test two locations, (1, 2) and (5, 7). No fault is observed.
6. The forth (0000, 0001) and fifth (0000, 0100) set also give desired results.
7. The next sets of input tuples give one to one correspondence. The next test tuple (0000, 0011) tests the crosspoint location (2, 4) which is a defective location.

With one-to- one mapping present in this case, the faulty crosspoint can be directly isolated.

8. Similarly, two more input tuples (0000, 0101) and (0000, 0110) are applied and all AND programmable locations are tested.
9. Once the product term locations are tested, the OR programmable plane is considered. All the OR programmable points give one-to-one mapping. Locations (6, 1), (6, 5) and (6, 8) can be successfully tested for fault.
10. Summary of Test: Out of the 5 potentially defective programmable crosspoints, 4 have been isolated successfully. These locations are (2, 4), (6, 1), (6, 5), (6, 8). There is a defect at potentially one or more locations from the following set: (1, 5), (2, 6), (3, 7), (4, 8).

5. FAULT-TOLERANT PLACEMENT SCHEMES USING THE PROPOSED FUNCTIONAL TEST ALGORITHM

Table 6 gives the number of OR locations utilized to implement few of the THmn gates. Having studied the mapping patterns of THmn gates and defect distributions, it has been noticed that the OR plane is vulnerable to have a physical defect overlap with a programmable ON location. Since a majority of programmable ON crosspoints fall on a single OR plane, it is essential to ensure OR plane redundancy. With the inclusion of a redundant OR wire, the reliability of the OR plane can be enhanced. With a redundant OR wire, in case an OR point is defective, the OR connection can be moved to the redundant wire without re-programming other crosspoints in the column which contribute to the product term. Another advantage is that since the OR planes are ORed together, the AND plane realization is not altered in any manner. This reduces the number of

crosspoints being retested and reprogrammed in case of using alternate placement schemes. As far as testing overheads are concerned, with the addition of a redundant row, only single additional input test tuple needs to be used to test the single OR location. Consider the scenario where a redundant OR row is not introduced and there is a defect at OR crosspoint location. In this case, the entire column will have to be moved to another location and all the corresponding crosspoint locations will have to be tested using additional test tuples. Not only will the number of programmable locations increase with this approach, but the testing space will also increase drastically. In case of some of the THmn gates such as TH12, TH23w2 where no more than 50% of the potentially programmable OR locations are used, it would be possible to rearrange the columns instead of using a redundant OR row. It is hence imperative to use suitable modeling and placement schemes to address these mapping issues.

Consider the following example where a redundant OR plane and column shift are used to realize TH34w2 gate on a defective PGMB having a 10% defect rate. Consider TH34w2 gate shown in figure 8. The functional test algorithm predicted a fault at one or more locations from the set (1, 5), (2, 6), (3, 7), (4, 8). Incase column 5 is moved to a parallel location and functionally tested; the observed output does not match with the desired one. This implies the fault location has not been detected. With (4, 8) moved to (4, 9) and tested, the input tuple generates desired output. The entire column (column 8) is moved to column 9 and tested functionally. Column 4 is moved to column 10 and results are validated using additional test tuples. The remaining 2 OR defective crosspoints with initial locations (6, 1) and (6, 5) are moved to (7, 1) and (7, 5) respectively. The reconfigured PGMB looks as shown in figure 9.

Figure 10 shows TH23 gate implemented on a defective PGMB. Location (6, 1) is a programmed ON location having a coinciding defect. This defect can be functionally tested using OR crosspoint input test tuple. One possible solution to work around this defect can be to use the inherently available locations. By using this approach, column 1 can be moved to column 7. In doing so, all the three ON crosspoints in column 1 are relocated to column 7. Another approach would be to introduce a redundant OR plane. In this case, location (6, 1) can be moved to location (7, 1). The input test tuple can then be used to validate the result. With the second approach, only the OR location would have to be relocated. Figure 11 shows TH23 gate realized successfully using a redundant OR plane row and Functional Test Algorithm. The following realization assumes there is no defect in the redundant OR plane. There exists a probability of defect locations present in the redundant plane. However, since defect rates are not seen to be greater than 10 percent, the probability of both OR locations in the same column being defective simultaneously are very low. Hence, for simplicity purposes a defect free assumption is made here. In case a defect at both OR locations in the same column are observed, the either column has to be shifted or the PGMB has to be discarded all together.

Table 6 Table giving number of programmable OR locations for THmn gates

Number of Programmable OR crosspoints							
TH12	TH23	TH24	TH23w2	TH34	TH33w2	TH34w2	TH44w3
4	6	10	5	8	5	8	7

Replacement and remodeling schemes result in an increase in number of programmable crosspoint locations tested. This increased test overhead can be mathematically calculated as follows:

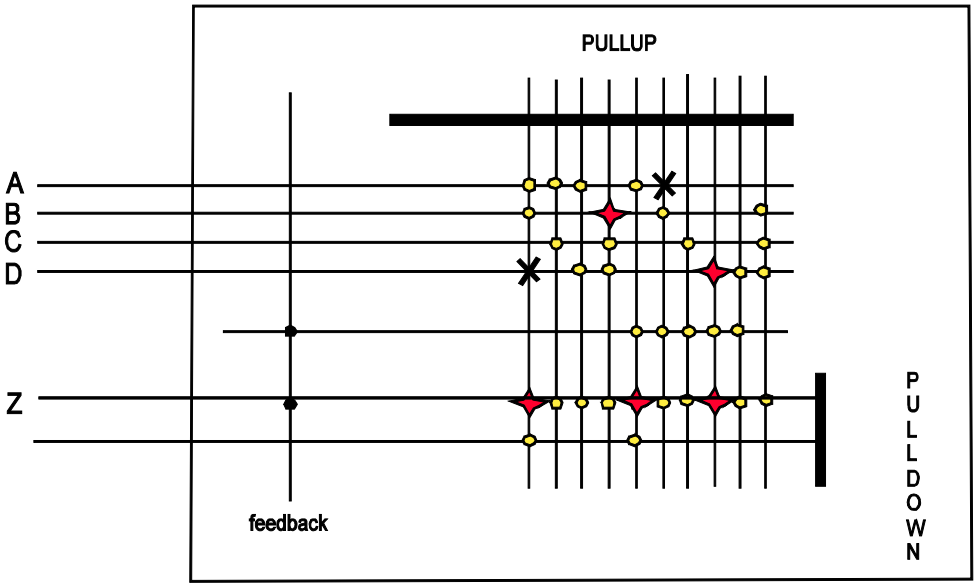

Figure 9 TH34w2 realized successfully using a redundant OR plane row and Functional Test Algorithm

- The total minimum number of programmable crosspoints = N
- The minimum number of test tuples applied to test 'N' programmable crosspoints = T
- Number of new programmable locations tested for true realization of function = n
- Number of additional test tuples used to test these locations = t
- Percentage increase in additional crosspoints = n / N
- Percentage increase in additional test tuples used for realization of function = t / T

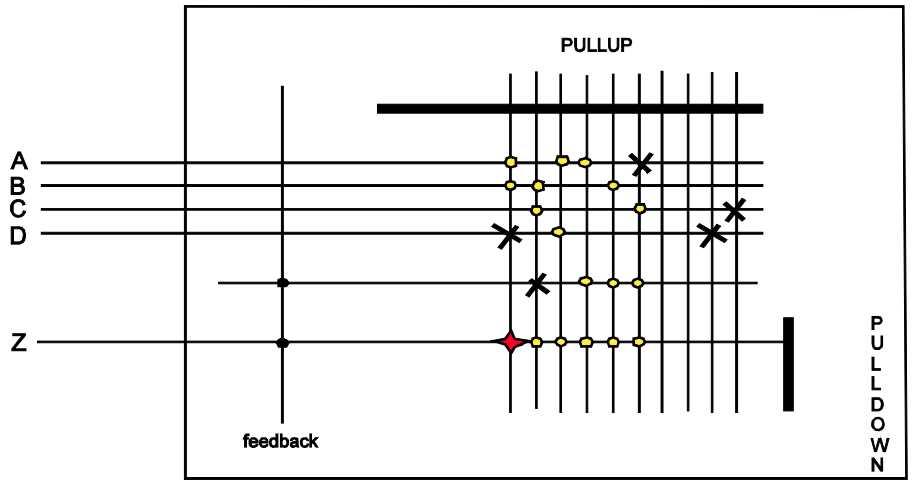

Figure 10 TH23 gate implemented on a defective PGMB

Using the above formulae, the test overhead introduced in realizing TH34w2 and TH23 gates over the PGMB can be obtained from table 7. The table shows the comparison between introducing a redundant OR row and using the available free columns in the original 6x10 grid.

Table 7 Table giving overhead estimates

Overhead Table

Description	TH34w2	TH23 (7x10)	TH23 (6x10)
Total minimum programmable crosspoints	25	18	18
Minimum no. of test tuples required	14	10	10
No. of new programmable locations tested	9	1	3
No. of additional test tuples used	9	1	3
Percentage increase in additional crosspoints	36%	5.55%	16.67%
Percentage increase in additional test tuples	64.28%	10%	30%

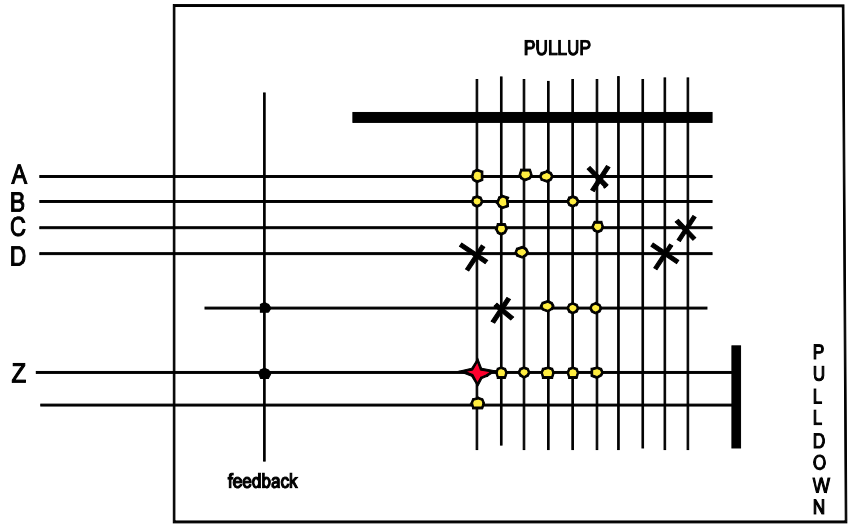

Figure 11 TH23 realized successfully using a redundant OR plane row and Functional Test Algorithm

6. PERFORMANCE EVALUATION MODEL

A performance evaluation model, as represented in figure 12 has been presented for understanding the performance of functional test scheme. This node model gives a diagrammatic representation of all possible categories the tested PGMBs can fall under. Each circle or node has a mathematical probability of being true which can be expressed in terms of three main parameters:

1. Defect rate varying from 0 to 10 percent.
2. N_{on}: the total number of programmable ON-input crosspoints for a given THmn gate. For TH33w2 gate, N_{on} is 15, whereas for TH23 the count is 18 and 30 for TH24.

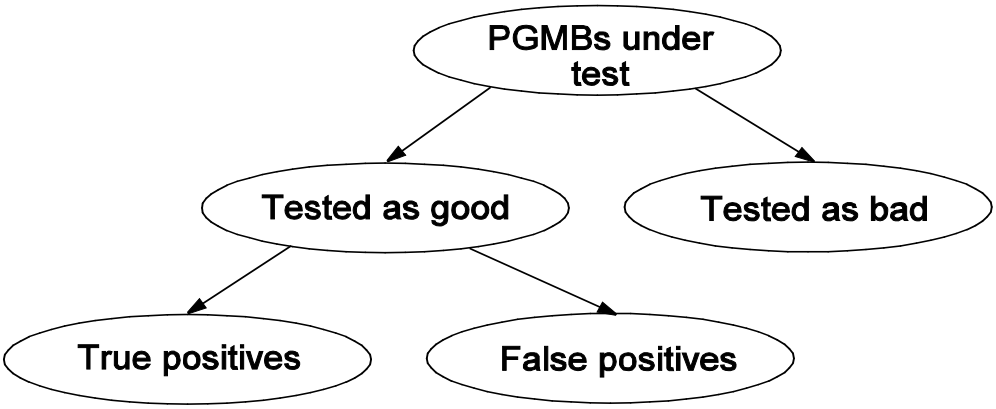

Figure 12 Performance Evaluation Model for Functional Test Approach

3. N_{test}: the number of programmable ON-input crosspoints being tested.

   N_{test} is typically a subset of N_{on}. It represents the number of programmable ON-input crosspoints being tested. With every test tuple applied, this count increases. With all test tuples applied, $N_{test} = N_{on}$. Range of N_{test} can be confined as follows: $0 < N_{test} \leq N_{on}$.

Let p be the defect rate induced during PGMB manufacture. The probability of an error free crosspoint can be represented as $1 - p$. With defect rates expected to vary anywhere from 0% to 10%, the fraction of good and defective PGMBs can be expressed using the following probability expressions.

- Fraction of PGMBs tested-as-good : $(1 - p)^{Ntest}$
- Fraction of PGMBs tested-as-bad : $1 - (1 - p)^{Ntest}$
- Fraction of correctly programmed PGMBs : $(1 - p)^{Non}$
- Fraction of incorrectly programmed PGMBs: $1 - (1 - p)^{Non}$
- Fraction of indeed good PGMBs : $(1 - p)^{Non}$
- Fraction of indeed bad PGMBs : $1 - (1 - p)^{Non}$

Tested-as-good represents those PGMBs which have been cleared to be good after testing only N_{test} number of ON programmable crosspoints. Tested-as-bad PGMBs are those which had a coinciding defect at atleast one programmable ON location from the set of N_{test} locations. Fractions of indeed good PGMBs are nothing but the fraction of correctly programmed PGMBs. This fraction represents "True positives", as shown in figure 12. Correctly programmed, also called indeed good PGMBs do not have a coinciding defective location on a programmable ON location. An indeed good PGMB is one which has been cleared of any defect after testing all N_{on} ON programmable locations.

The performance model has been designed to analyze the quality metrics associated with the functional test scheme. This model has been developed to setup performance indicators for THmn gates. Performance indicators explained below can be used to quantify the performance of the proposed functional test algorithm.

6.1 ACCURACY

Accuracy of the functional test scheme can be defined as the ratio of number of tested as bad PGMBs over indeed bad PGMBs. Figure 13 and figure 14 shows the accuracy plot for TH23 gate obtained from both simulation and mathematical models. The results based on mathematical model and simulations bare a close resemblance. The mathematical plots are generated using the mathematical formulas explained above. It is evident from the results that accuracy ratio increases with increase in the number of test tuples covered. For N_{test} = 0, no bad PGMBs are observed and this ratio cannot be defined. As N_{test} approaches N_{on}, more ON locations are covered, increasing the accuracy factor.

For $N_{test} = N_{on}$, accuracy is 1 since all ON programmable locations have been tested for defect. Figure 15 and figure 16 show accuracy plots for TH33w2 gate based on simulation and mathematical results. In order to test 10 ON crosspoint locations, 4 test tuples have been used and 9 in total to test all 15 ON crosspoint locations.

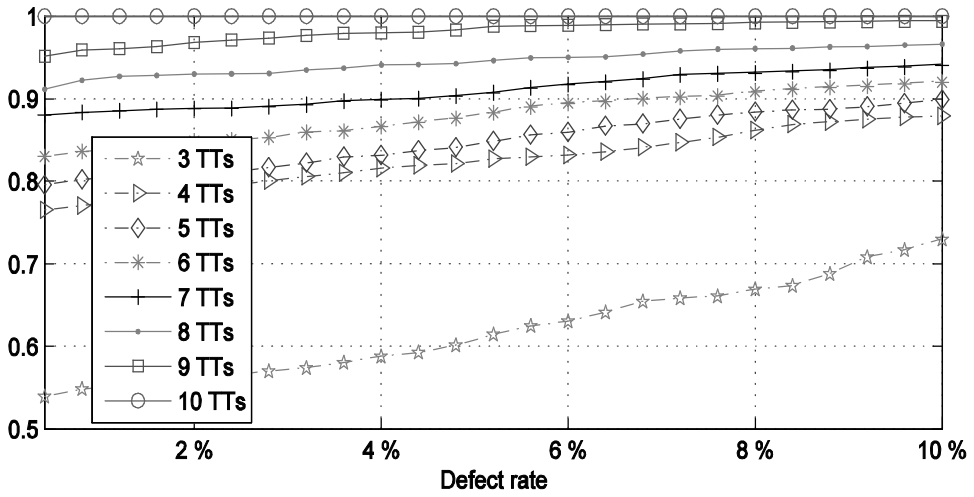

Figure 13 Simulation based accuracy plot for TH23 gate

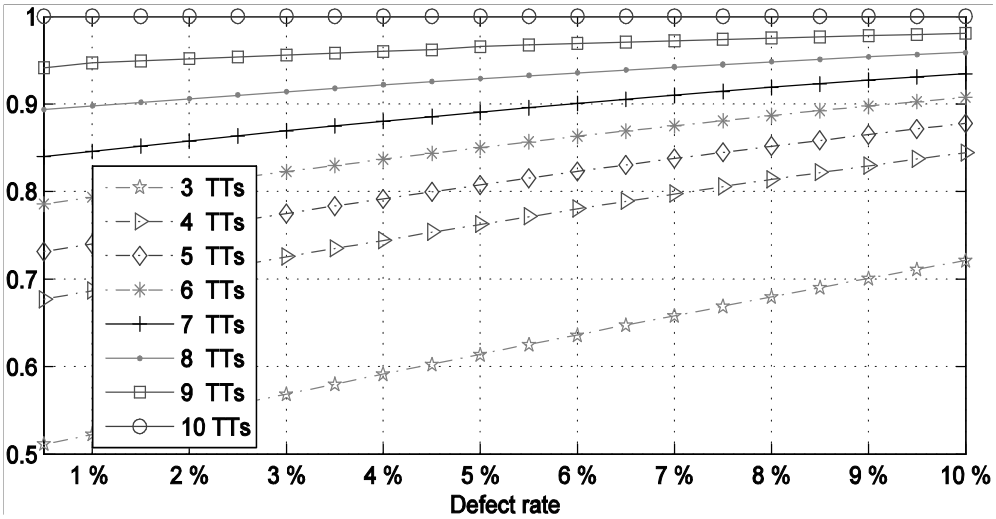

Figure 14 Mathematical formula based accuracy plot for TH23 gate

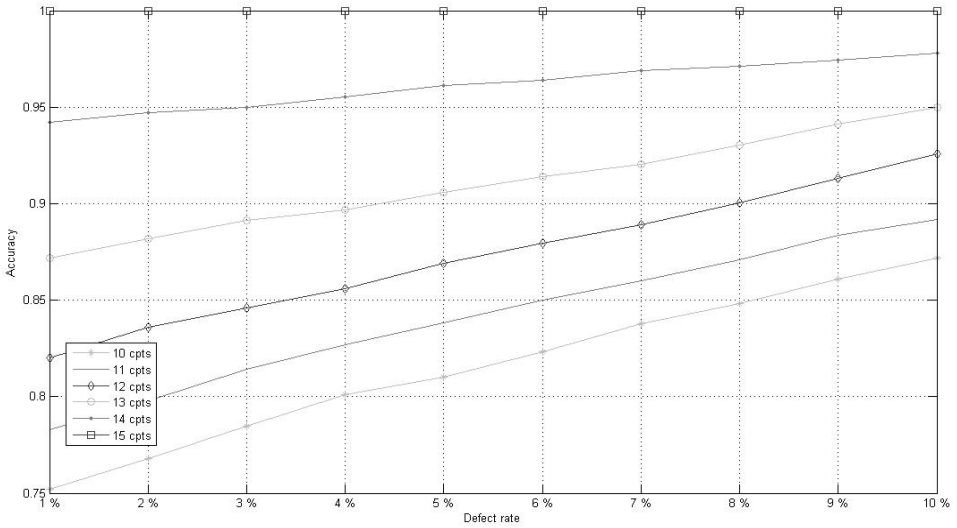

Figure 15 Accuracy plot for TH33w2 gate based on simulation results

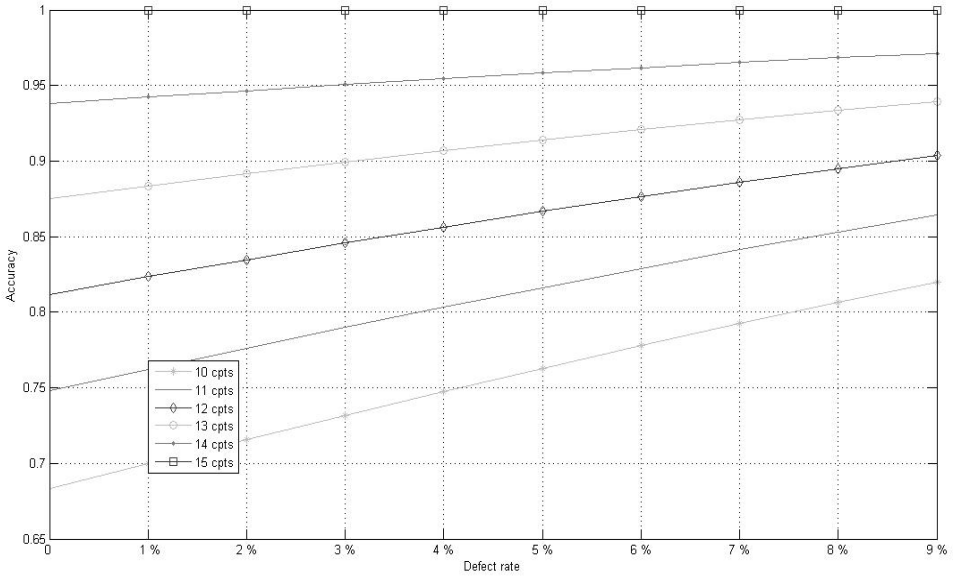

Figure 16 Accuracy plot for TH33w2 gate based on mathematical formula

6.2 ESCAPE FACTOR

Escape factor is another “Figure of Merit” parameter, which is complementary to Accuracy. Escape Factor = 1 - Accuracy.

Escape factor relates to the fraction of bad PGMBs that have escaped the scan due to reduced set of test tuples. The escaped PGMBs are anticipated to have atleast one defective location coinciding with ON programmable location.

6.3 ESCAPE TOLERANCE

Escape tolerance is another “Figure of Merit” that has been derived from the node model. The ratio of indeed good PGMBs over the tested as good ones is the escape tolerance for a THmn gate. 100 % escape tolerance implies that all the indeed good PGMBs have been covered in the set of tested as good. Higher the value better is the performance. Referring to figure 19 it can be seen that escape tolerance falls from 1 for 0 % defect rate to 0.38 for 10 % defect rate with three test tuples. Three test tuples cover nine out of 18 crosspoints, which signifies an ON crosspoint coverage of 50 %. Increasing defect rate will increase the probability of defective crosspoints which will bring down the fraction of indeed good PGMBs. This justifies why escape tolerance ratio falls with increasing defect rate. It can be observed from figure 17 that escape tolerance with 1% defect rate and Ntest = 10, 11 12, 13, 14 and 15 is greater than 0.95. However, the same values fall to as low as 0.70 with Ntest= 10 for 10% defect rate. With increased defect rates, number of defective location in a PGMB will increase. With this, the chances of an overlap between ON crosspoint and a defective location increase, thus reducing the total number of good PGMBs. As Ntest approaches Non, the escape tolerance increases.

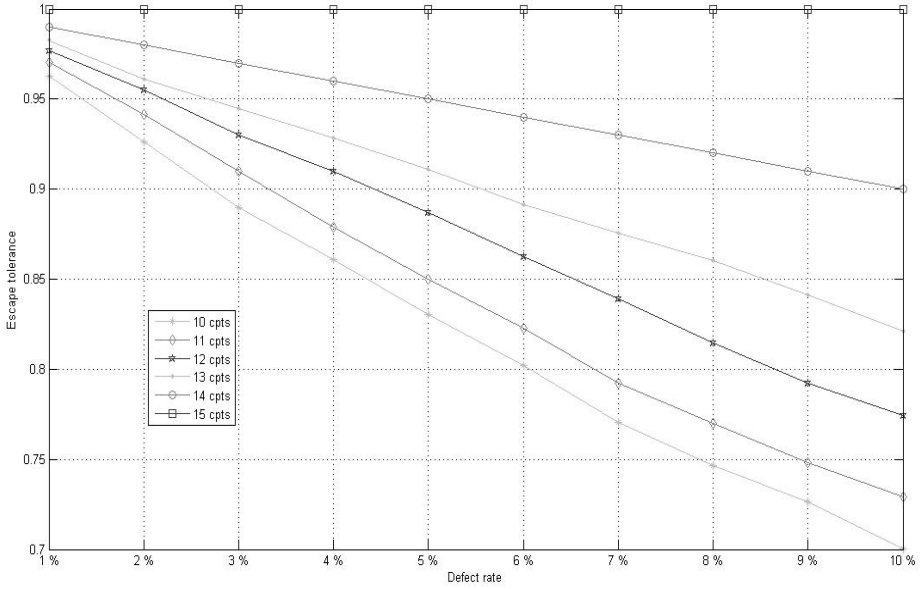

Figure 17 Escape Tolerance for TH33w2 gate based on simulations

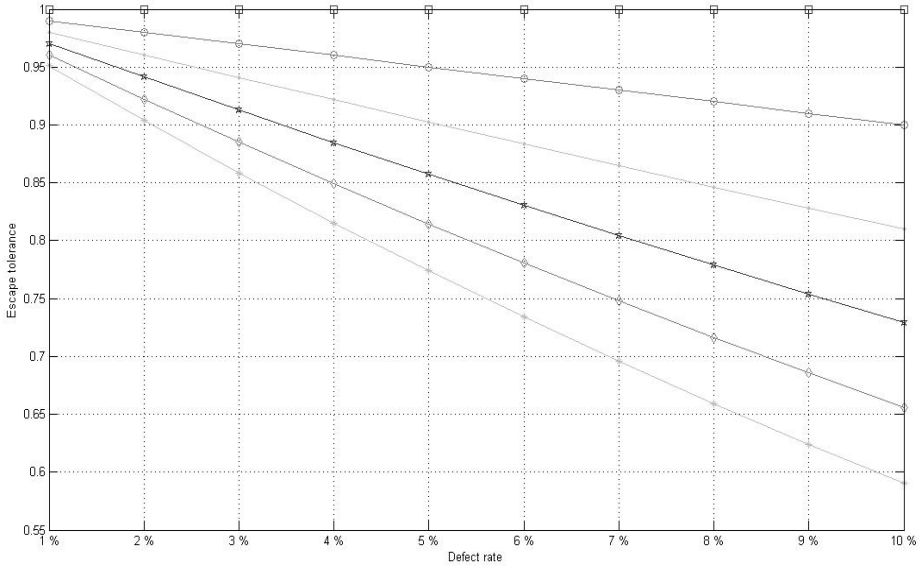

Figure 18 Escape Tolerance for TH33w2 gate based on mathematical model

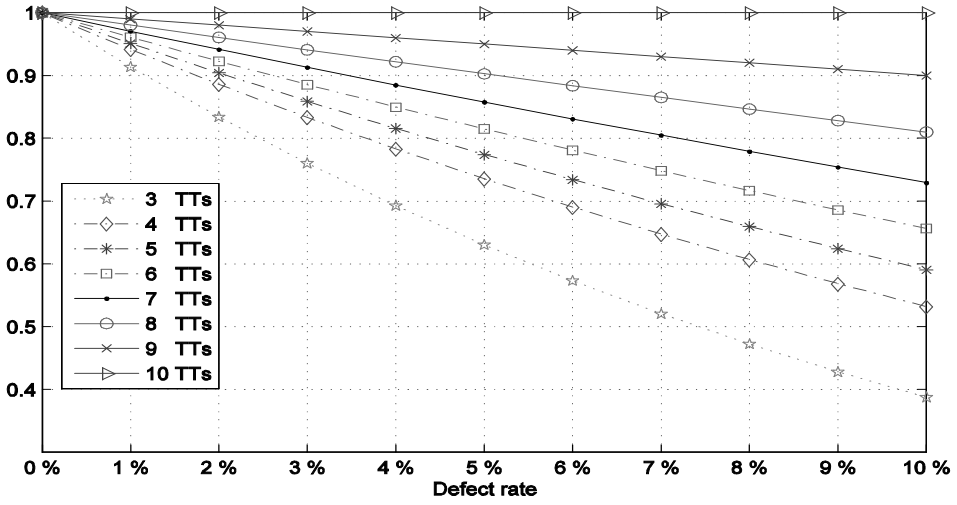

Figure 19 Escape Tolerance for TH23 gate based on mathematical formula

Figure 21 shows the accuracy plots for TH23 gate with test tuples applied in reverse order of priority. By using test tuples in reverse order of priority, the OR plane ON crosspoints are tested first, followed by AND plane. It can be observed that accuracy is very low and increases gradually with each test tuple. For test tuples applied in order of decreasing priority, higher accuracy can be using comparatively lesser number of test tuples applied. Consider figure 21 shown below. In case an accuracy factor of 0.9 with a defect rate of 10% is aimed, with test tuples applied in reverse order, atleast 15 programmable locations will have to be tested. To test these 15 ON crosspoint locations, 9 out of the 10 test tuples will have to be applied. For achieving the same accuracy factor of 0.9 with higher order test tuples applied first, only 6 out of 10 test tuples will have to be used.

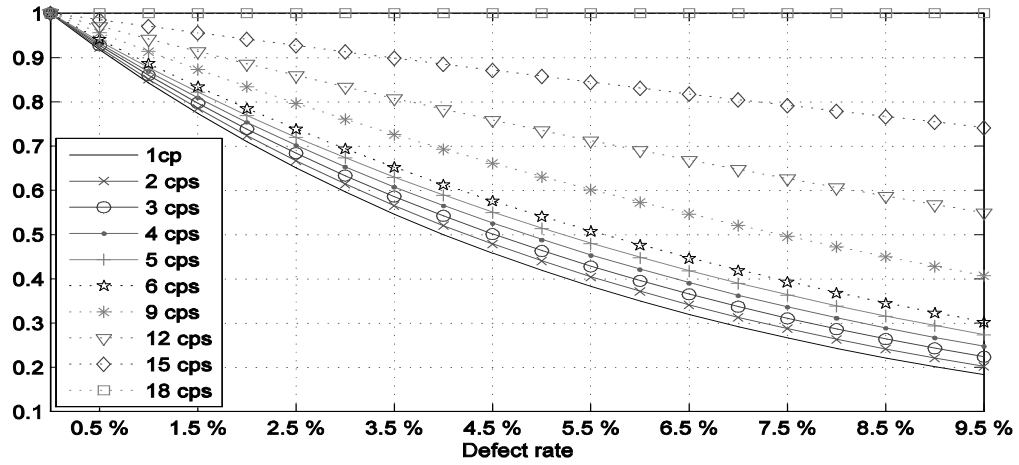

Figure 20 Escape Tolerance plot for TH23 gate using reverse order of priority inputs

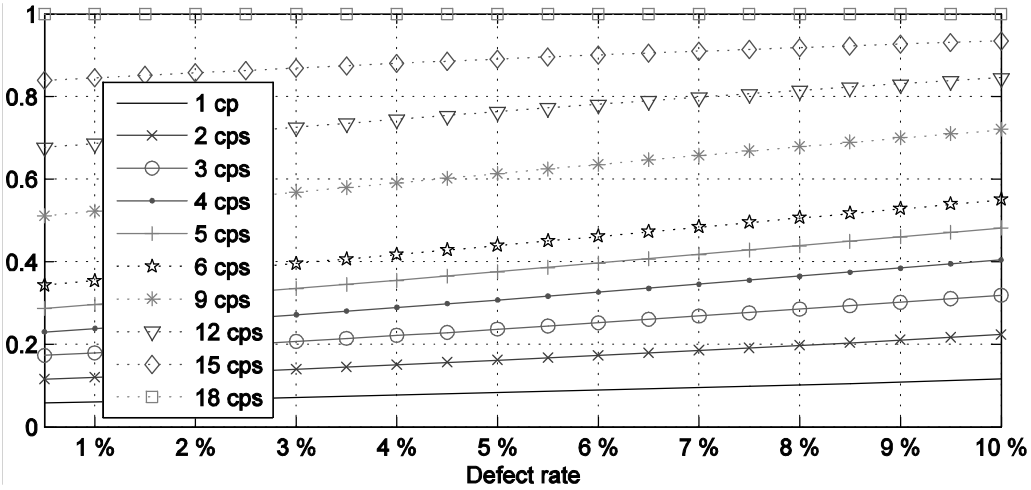

Figure 21 Accuracy plot for TH23 gate using reverse order of priority inputs

7. CONCLUSION

The proposed post configuration testing scheme is aimed to identity all Stuck-at-0 faults that overlap with programmable ON crosspoint locations. The testing scheme uses unique input test tuple set and identifies unique crosspoint locations specific for each

threshold gate. Not only does the proposed algorithm provide complete test coverage, but it also manages to provide excellent fault-tolerance. The proposed algorithm also manages to reduce the testing overhead significantly, as compared to the raw testing scheme. Performance analyzers like accuracy and escape tolerance further validate the effectiveness of the proposed test scheme.

Using the results of the Functional Test Algorithm, defective ON crosspoint locations can be shifted to alternate defect-free locations. Using a redundant OR plane the defective OR crosspoint locations can be moved to the corresponding redundant location without disturbing the AND plane ON crosspoints. Another approach can be to shift an entire column to an alternate available non-programmed defect-free column. A combination of the two above approaches can also be considered based on the threshold gate being realized. These approaches can help in correct realization of the threshold gate inspite of inherent defects.

8. REFERENCES

[1] Mathew M. Ziegler and Mircea R. Stan, "CMOS/Nano Co-Design for Crossbar Based Molecular Electronic Systems", IEEE Transactions on Nanotechnology, Vol. 2, No. 4, December 2003

[2] Mathew M. Ziegler and Mircea R. Stan "Design and Analysis of Crossbar Circuits for Molecular Nanoelectronics", IEEE Nanotechnology Conference, pp. 323-327, 2002.

[3] D. Whang, S. Jin and C. M. Lieber "Large Scale Hierarchical Organization of Nanowires for Functional Nanosystems", Japanese Journal of Applied Physics, Vol 43, No. 7B, 2004.

[4] Y. Cui and C.M. Lieber "Functional Nanoscale Electronic Devices Assembled Using Silicon Nanowire Building Blocks", Science, Vol. 291, pp. 851-853, 2001.

[5] Nicolas A. Melosh, Akram Boukai, Frederic Diana, Brian Geradot, Antonio Badolato, Pierre M. Petro®, James R. Health, "Ultrahigh-Density Nanowire Lattices and Circuits", Science, Vol. 300, pp. 112-115, 2003.

[6] R. Bonam, S. Chaudhary, Y. Yellambalase and M. Choi, "Clock-Free Nanowire Crossbar Architecture based on Null Convention Logic (NCL)", 7th IEEE International Conference on Nanotechnology (IEEE-Nano), Apr 2007.

[7] J. Huang, M.B. Tahoori and F. Lombardi, "On the defect tolerance of Nano-Scale Two Dimensional Crossbars"' IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp 96-104, Oct 2004.

[8] M. Jacome, C. He, G. de Veciana and S. Bijansky, "Defect Tolerant Probabilistic Design Paradigm for Nanotechnologies" IEEE/ACM Design Automation Conference on Application-Specific Systems, Architectures and Processors, pp 261-273, 1996.

[9] Yadunandana Yellambalase, Minsu Choi and Yong-Bin Kim "Inherited Redundancy and Configurability Utilization for Repairing Nanowire Crossbars with Clustered

Defects", 21st IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, DFT' 06.

[10]Karl M. Fant, Scott A. Brabdt, "NULL Convention Logic System", US patent 5,305,463 April 19, 1994.

[11]S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb, "Optimization of NULL Convention Self-Timed Circuits", Integration, The VLSI Journal, Vol. 37, No. 3, pp. 135-165, 2004.

[12] S. C. Smith, R. F. DeMara, J. S. Yuan and D. Ferguson, "Delay-Insensitive Gate-Level Pipelining", Integration, The VLSI Journal, Vol. 30, pp. 103-131, 2000.

[13]R. Bonam, Y. Yellambalase and M. Choi, "Redundancy Optimization for Clock-Free Nanowire Crossbar Architecture", 7th IEEE International Conference on Nanotechnology (IEEE-Nano), Apr. 2007. .

[14]M. Mishra and S. Goldstein, "Scalable Defect Tolerance for Molecular Electronics", Workshop on Non-Silicon Computation (NSC-1), pp.78, 2002. pp.78, 2002.

[15]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, Oct. 2005, pp. 305-313.

[16]Sriram Vekateswaran and Minsu Choi, "Post-Configuration Testing of Asynchronous Nanowire Cross- bar Architecture", 8th IEEE International Conference on Nanotechnology (IEEE-Nano), Aug 2008.

II. POST-CONFIGURATION OF ASYNCHRONOUS NANOWIRE CROSSBAR ARCHITECTURE

Sriram Venkateswaran and Minsu Choi

Dept of ECE, Missouri University of Science and Technology

Rolla, MO 65409-0040, USA

svf44, choim*g*@mst.edu

ABSTRACT

Asynchronous nanowire crossbar architecture has been recently proposed to eliminate the clock distribution network from conventional clocked counterpart. The proposed clock-free architecture is envisioned to enhance the manufacturability with simpler periodic structure and to improve the robustness by removing various timing-related failure modes. Even though the proposed clock-free architecture has numerous merits over its clocked counterpart, it is still not free from high defect rates induced by nondeterministic nanoscale assembly. In order to address this issue, our research team has been working on developing test schemes for effective mapping of threshold gates onto Programmable Gate Macro Blocks (PGMB). We have come up with a novel functional test approach which uses prioritized input tuples to effectively stimulate coinciding defects in configured PGMB. Numerous preliminary plots and results obtained till date prove that this scheme can be used to achieve high test efficiency for any threshold gate. The main motivation behind this research is to propose a comprehensive test scheme which can achieve high enough test coverage with acceptable test overhead. Parametric simulation results using MATLAB have been used to show potential performance of this testing scheme.

1. INTRODUCTION

The recently proposed asynchronous nanowire crossbar architecture is based on the delay insensitive data encoding and self timed logic - therefore it is totally clock-free [1]. This helps eliminate all the failure nodes related to timing. The other potential benefits of using this architecture include enhanced manufacturability, scalability, robustness and defect and fault tolerance [2]. The proposed asynchronous nanowire crossbar architecture is based on a delay-insensitive logic paradigm known as Null Conventional Logic (NCL) [3]. NCL logic can be realized using 27 threshold gates [3]. These gates can be used to implement any expression involving upto four variables.

In the proposed architecture, every threshold gate macro that can be programmed on to a PGMB has a certain predefined pattern of crosspoint placement that would give the corresponding functionality of the gate. A TH23 gate on a PGMB is shown in figure 1. For instance, a TH23 gate can be expressed as $F = AB+BC+AC+AF^{0}+BF^{0}+CF^{0}$, where A, B, C are the primary inputs and F' is the output feedback. The first three product terms in this Boolean equation are for the threshold behavior of the gate since the quorum of this gate is 2. Also, the last three product terms (which is also equivalent to $(A + B + C) F^{0}$) are for the hysteresis behavior. Once the output F is asserted, the only way to make it back to zero is reset all primary inputs.

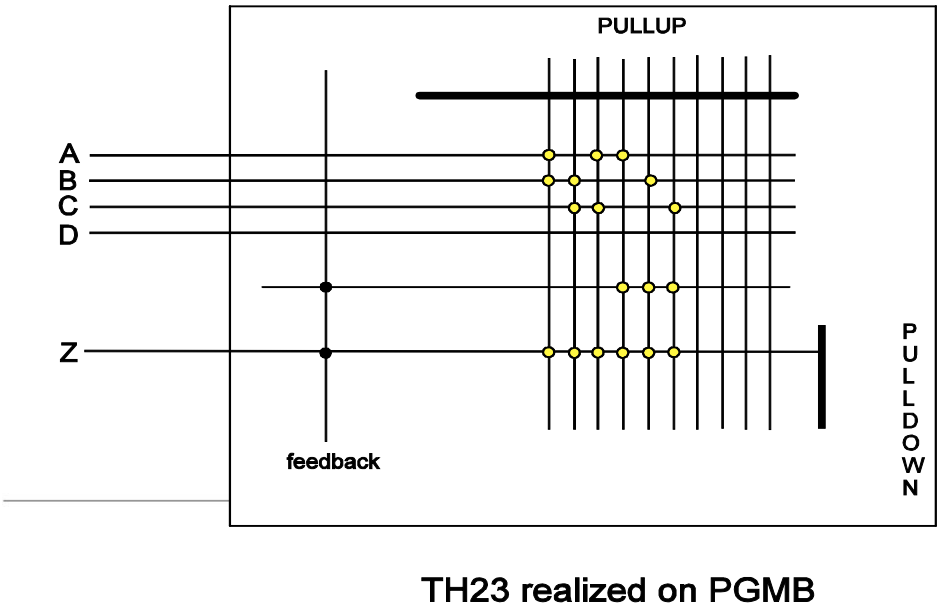

Figure 1 TH23 gate configured on a PGMB

Defect rates arising due to fabrication vary on an average from 0% to 10% [4]. Researchers are still not able to accurately predict the defect rate in these PGMBs. The effect of these defects on the logical operation of the circuit needs to be scrutinized. These defects have to be tolerated to maintain proper functionality of the circuit.

2. FUNCTIONAL TEST APPROACH

The most primitive way of testing a nanowire crossbar is to test individual crosspoints one by one by sequentially scanning through them and generate a defect map. This is not only a very laborious scheme, but also introduces a considerable amount of testing overhead in time/space complexity [2]. The functional test scheme proposed in our paper is designed to test maximum number of programmable crosspoints using the minimal number of test inputs. The test inputs are nothing but logical inputs based on the

logical expression realized by any THmn gate. As shown in the algorithm, the first step is to map the THmn gate onto the PGMB following which the truth table for the specific gate is generated. A list of prioritized inputs is generated for testing the ON crosspoints. In case our objective is to scan the PGMS for defects, then inputs are applied in order of decreasing priority. In this manner, the entire ON programmable space is successfully scanned. In case locating the defect is essential, then partial isolation and location can be achieved. This is however confined only to the OR plane crosspoints. The reason being they have direct correspondence with the test tuples. The fault count thus generated from either of the approaches specifies the number of defective crosspoints generated. Another feature of our approach is that the functional test scheme being proposed in this work avoids the issues associated with this raw crossbar testing. The crosspoints under test are limited by the number of ON-inputs (i.e., crosspoints that should be programmed as ON) of the given threshold gate macro. Minimizing the test space helps reduce the test time. In addition, since Boolean inputs are used to check for defects, these programmable inputs can be prioritized according to the number of ON-inputs they can cover. The other advantage of this approach is the minimal number of test inputs it takes to cover the test space. On close comparison of desired functional output due to defect free mapping and one generated due to defective crosspoints at programmable locations, prioritized input tuple levels have been set for each threshold gate. These prioritized test tuples can be applied sequentially to validate the programmed gate function. Table 1 shows the prioritized test tuples for TH23 gate. Table 2 shows the percentage coverage attained by each test tuple for a set of threshold gates.

Table 1 Test Tuples and their corresponding number of testpoints for TH23 gate

Test Tuple	Number of programmable locations tested	
1	3	crosspoints
2	6	crosspoints
3	9	crosspoints
4	12	crosspoints
5	13	crosspoints
6	14	crosspoints
7	15	crosspoints
8	16	crosspoints
9	17	crosspoints
10	18	crosspoints

Table 2 Total crosspoints tested vs prioritized test tuple count

		Prioritized Test Tuple Count				
Gate	*Non*	3	4	5	6	7
TH23	18	50%	66.67%	72.22%	77.8%	83.3%
TH24	30	40%	53.33%	66.67%	70.0%	73.3%
TH34	28	28.57%	35.71%	42,86%	50.0%	57.17%
TH33w2	15	53.3%	66.67%	73.3%	80.0%	86.67%
TH44w3	21	47.6%	57.14%	66.67%	71.42%	76.19%

Let's consider TH23 gate. The three primary inputs will generate 8 input bit patterns ranging from 000 to 111. Figure 1 shows a TH23 gate configured on a PGMB. We can see that there are 18 ON-inputs represented by highlighted dots. Imperfect assembly may cause any one or more of these points to be OFF. For example, a defect at the left-most crosspoint in the first row results in a faulty function of $F^{¤} = B + BC + AC + AF^{0} + BF^{0} + CF^{0}$. Notably, one or more test input tuples can be found by comparing

output columns of F and $F^{¤}$ in their truth table. The proposed functional test scheme also applies input tuples in the order of their priority level and validates outputs from those input tuples. As the number of applied test tuples increases, the total number of testable ON-input crosspoints increases. Table 1 shows the number of testable ON input crosspoints as a function of test tuple count. The first sets of 4 inputs test 12 out of the 18 possible programmable locations for defects. In case a particular test input results in an undesired output, then the ON-crosspoints under test are tested as bad. In case of the TH23 gate, using the first 3 most highly ranked input tuples cover 50% of the total test space. Using another input increases this to 66.67%. This rate rises to 72.2, 77.8, and 83.3% respectively with each additional input. Table 2 shows the coverage values for all 5 gates under consideration. This is a very important point especially when we have a large input sample space. For example, in order to test 75% of ON- crosspoints, 6 input tuples should be applied. With this set level, we can achieve a relative testability (i.e., number of total tested- good crosspoints / number of total crosspoints tested) of greater than 90% on average.

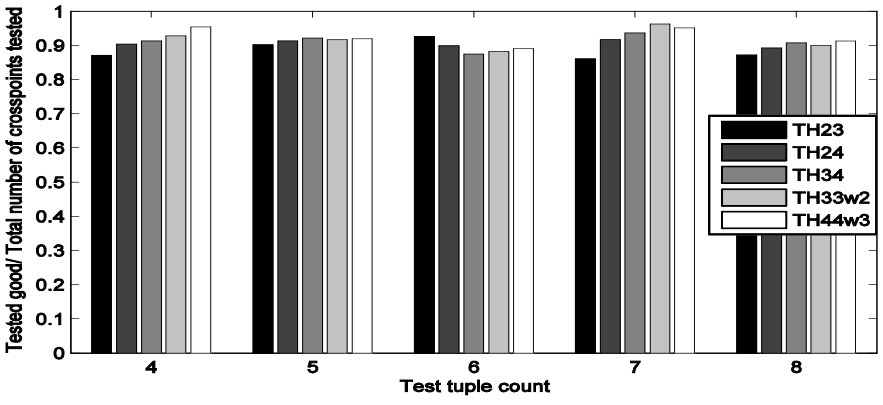

Figure 2 Relative Testability of THmn gates

Figure 2 shows a plot of relative testability for 5 different threshold gates. These 5 threshold gates have been considered in the following plots because they cover the maximum possible input combinations and can be considered as representatives of the several other types of gates. The TH23 gate has only two priority levels as shown in figure 3. TH34, on the other hand, has 3 input priority levels with the first tuple testing 4 ON-crosspoints, the second highest set testing 2 points each and finally the lowest level providing one to one correspondence.

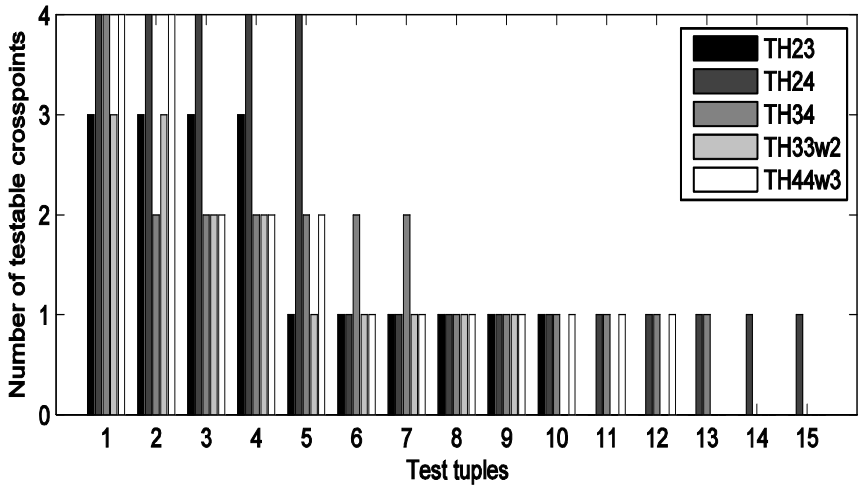

Figure 3 Testable crosspoints with each input for THmn gates

Consider figure 2 which gives the tested good over the tested bad PGMB ratio. This plot helps us understand the relative distribution of the two types of PGMB in the sample. The nature of the plots show that as the defect rate decreases and as the number of crosspoints under test increases, the ratio of tested good over tested bad falls considerably. This count is of extreme significance especially when we require the distribution of bad crosspoints for the purpose of repair. In case of repair being the

priority, the inputs are applied in the order of increasing priority. This will enable maximum one to one correspondence to be achieved. In the set of programmable crosspoints, the OR plane has highest priority. In order to account for any potential failure in any programmable OR crosspoint, we have proposed a unique solution. Our solution suggests implementing OR plane redundancy. A parallel OR plane can be introduced. Figure 4 represents the distribution of bad PGMBs due to at least one defect in any of the programmable OR crosspoint locations. TH24 gate has the highest number of defective PGMBs since it uses all the 10 programmable crosspoints. TH33w2 on the other hand has only 5 out of the available 10 which are programmed. This concentration of defects over a single OR plane especially for higher defect rates suggests the need to focus on the OR plane. For minimizing the defective PGMBs due to defective OR plane, we need to test this plane by using low priority inputs. This can help locate the defects which can be repaired or corrected accordingly in future.

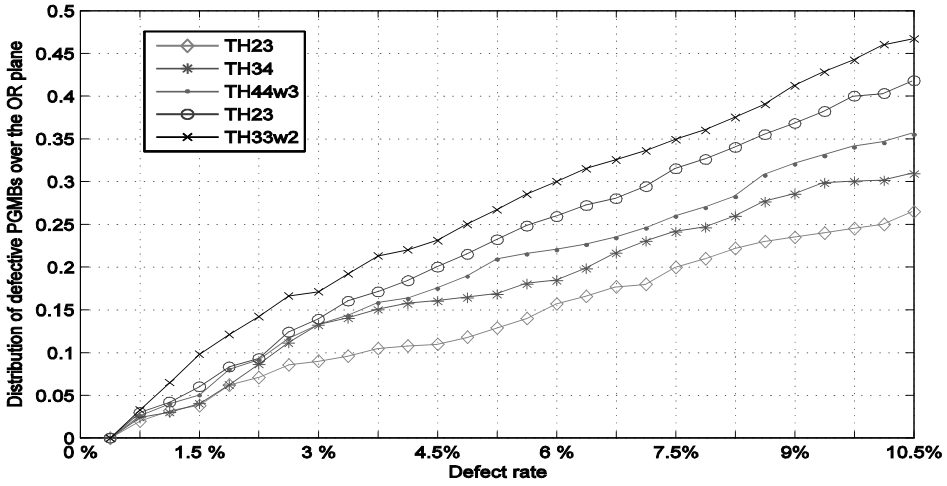

Figure 4 Distribution of defective PGMBs due to defective OR plane crosspoints

Consider a 6x10 grid and a TH24 gate is to be implemented on this. We have 10 programmable OR locations in this gate. By introducing a redundant wire we increase the PGMB dimensions to 7x10. In case the OR crosspoint of the j_{th} row is defective; we can program the crosspoint on the j-1 th row and corresponding to the same column number. Only if both the points are defective simultaneously will there be a manipulation in the desired output. In case one of them is defective, we can still achieve efficient programmability with this approach. The plots and results have been obtained considering the defect rate of 10%, which is the worst case under the current prediction.

Accuracy is a figure of merit which has been used to quantify our test approach. Accuracy of the functional test scheme can be defined as the ratio of number of tested as bad PGMBs over the total number of bad PGMBs. It is evident that the accuracy ratio increases with increase in defect rate and the number of test tuples covered. For lower defect rates and lesser number of test tuples, the numbers of bad crosspoints are few. Of the two dependent parameters, only the number of test tuples can be varied.

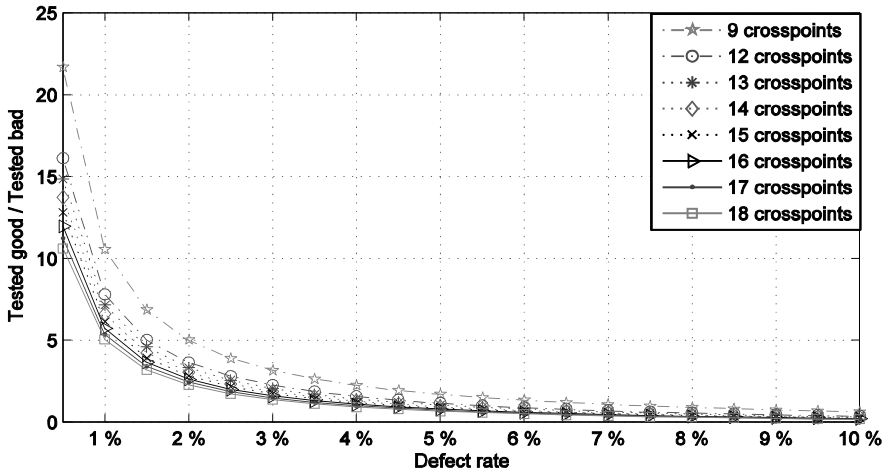

Figure 5 Tested good over tested bad PGMB ratio for varying defect rates and variation in number of crosspoints

Hence, test tuple count should be suitably selected with due consideration to required accuracy. In figure 6 and figure 7, accuracy plots for TH23 gate with varying number of test tuples and increasing defect rates have been generated. It is interesting to note that in both the plots, the accuracy rates increase with defect rate. This is due to the increase in total number of bad PGMBs with increase in defect rate. When the prioritized inputs are applied in reverse order, the accuracy is very low and increases slowly with each test tuple. For test tuples applied in order of decreasing priority, we can achieve higher accuracy for comparatively lesser number of tuples applied. Having said that, if location of defect is essential, then a compromise needs to be made on the accuracy front.

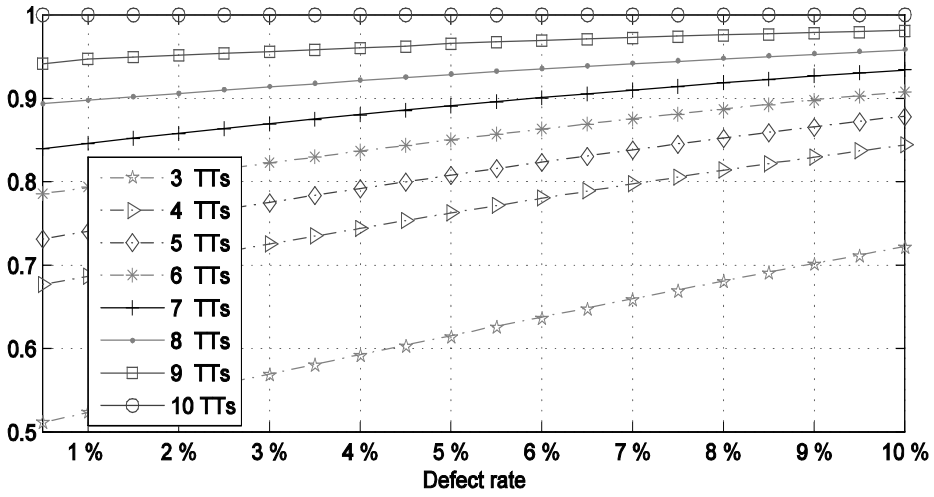

Figure 6 Accuracy plot for TH23 gate

This is a necessary tradeoff. Another complementary factor that can be generated is escape factor. It is the ratio of actually bad PGMBs over total identified bad PGMBs. Actual bad ones are those which have been subject to all the test tuples possible to cover the entire programmable space. Total identified bad PGMBs are those which have been identified as bad when a reduced set of test tuples have been applied. This reduced set,

called as N_{test} is a subset of the total test points, denoted by N_{on}. It is clear from definition that accuracy and escape factor are complementary to each other. Escape factor is greater when lesser number of test tuples is applied. For increasingly larger number of test tuples, the number of indeed bad PGMBs is lesser, bringing down the escape factor. A low value for escape factor means lesser the chances of an indeed bad PGMB escaping as a tested good one.

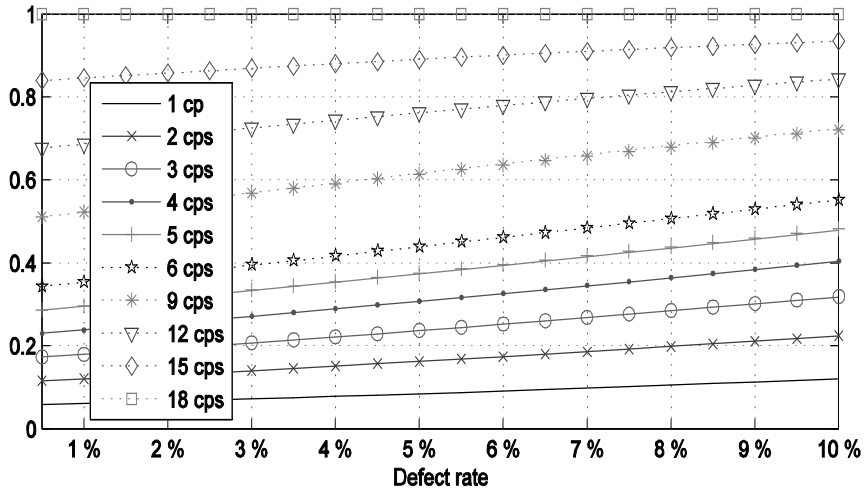

Figure 7 Accuracy plot for TH23 with test tuples applied in order of increasing priority

3. CONCLUSION AND FUTURE WORK

The complete sequential scan testing of nanowire crossbar guarantees the perfect test coverage. However, this scheme is rather laborious in terms of time/space complexity. Thus, we have proposed a novel test approach for the recently proposed asynchronous nanowire crossbar architecture. The proposed testing scheme is to functionally test ON-crosspoints solely by applying a number of input tuples. Notably, some of the input tuples may be used to cover more than one ON- crosspoint. Thus, it is

possible to prioritize them to achieve the desired combination of test coverage and overhead. The trade-off between the performance (i.e., test coverage) and the overhead (i.e., number of total input tuples applied) is shown in preliminary simulation results in this paper. Having said that, in case of locating defects being our priority, we lose one-to-one correspondence with the input tuples with increasing priority. We will hence no longer be able to directly isolate AND plane defects. We will have to use combination of inputs to locate faults. In future, we plan to extend our functional test algorithm to accommodate this. All these approaches are aimed at maximizing the utility of PGMBs in spite of the inherent fabrication defects.

4. REFERENCES

[1] R. Bonam, S. Chaudhary, Y. Yellambalase and M. Choi, "Clock- Free Nanowire Crossbar Architecture based on Null Convention Logic (NCL)", 7th IEEE International Conference on Nanotechnology (IEEE- Nano), Apr 2007.

[2] Ravi Bonam, Yong-Bin Kim, Minsu Choi, "Defect-Tolerant Gate Macro Mapping and Placement in Clock-Free Nanowire Crossbar Architecture", 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07. 26-28 Sept. 2007 Page(s):161 - 169

[3] S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb", Optimization of NULL Convention Self-Timed Circuits", Integration, The VLSI Journal, Vol. 37, No. 3, pp. 135-165, 2004.

[4] J.Huang, M.B. Tahoori and F. Lombardi, “On the defect tolerance of Nano-Scale Two Dimensional Crossbars” IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp 96-104, Oct 2004.

III. NOVEL FUNCTIONAL TESTING TECHNIQUE FOR ASYNCHRONOUS NANOWIRE CROSSBAR SYSTEM

Sriram Venkateswaran[1], Jong-Seok Lee[1;2] and Minsu Choi[1]
[1]Dept of ECE, Missouri University of Science and Technology Rolla, MO, USA
{svf44, leejongs, choim*g*}@mst.edu
[2]Dept of Computer Engineering and Education, Woosuk University, Korea

ABSTRACT

The recently proposed asynchronous nanowire clock-free crossbar architecture is envisioned to enhance the manufacturability and to improve the robustness of digital circuits by removing various timing- related failure modes. Even though the proposed clock-free architecture has numerous merits over its clocked counterpart, it is still not free from high defect rates inherently induced by nondeterministic nanoscale assembly. In order to address this issue, a novel functional test scheme for validating threshold gates on Programmable Gate Macro Blocks (PGMB) has been proposed. The proposed approach tests only the crosspoints programmed as ON state using input patterns unique to the given threshold gate macro. The proposed scheme helps achieve correct programmability with minimal test over- head. This test scheme can be used to assure the true functionality of any threshold gate on a given PGMB. Parametric simulation results using MATLAB have been used to show the potential performance of this testing scheme.

Index Terms - Asynchronous nanowire crossbar system; Functional testing; Defect and fault-tolerance; Parametric simulation.

1. INTRODUCTION

Many of the nanowire crossbar architectures are envisioned to be clocked. They should have the clock being propagated throughout the circuit for synchronizing the functional blocks. The recently proposed asynchronous crossbar architectures however manages to eliminate the need for clock and clock distribution network. The asynchronous nanowire architecture is based on a delay-insensitive data encoding and self timed logic. Since no clock distribution network is needed in this architecture, all failure modes related to timing are eliminated. Potential advantages from the proposed architecture include enhanced manufacturability, scalability, robustness and defect and fault-tolerance [2]. The asynchronous crossbar architecture uses Null Convention Logic (NCL) [3, 4, 5]. Null Conventional Logic integrates data and control into a single signal. The two states, DATA and NULL are used by this technology for achieving synchronization and I/O control. The DATA wave front contains data to be processed by the combinational circuit and the NULL wave front is a non-data value used to reset the logic gates in the circuit. They are used to separate two consecutive DATA wavefronts [3]. The main reasons why NCL is suitable is because these circuits are less complex, insensitive to delay and are more reliable since they do not experience problems such as clock skew and race conditions [2].

The basic unit of crossbar architecture is the programmable gate macro block (PGMB) [2]. The PGMB has AND and OR crossplanes formed by nanowire diode crossbars. The vertical wires with pull-up resistors form the AND terms and the horizontal wires with the pull-down resistors form the OR logic. This is a two level logic consisting of the input and the feedback logic. The feedback logic is implemented by

using the feedback loop which drives the previous output back to the input wire. The asynchronous cross- bar architecture uses NCL (Null Conventional Logic) [2], a delay insensitive paradigm, which helps eliminate clock distribution network from the circuit. There are a total of 27 threshold gate macros [4] that can be implemented in NCL.

A discrete threshold gate [4], represented as THmn has atleast m signals asserted for its set condition. All signals de-asserted is the reset condition. With a total of n inputs atleast m out of n are needed for assertion. In case the weights of inputs are not specified, the default value is 1. For e.g., with TH33w2 gate, the weight of the higher order bit input bit (i.e. bit A in ABC input pattern) is 2 and that of B and C is 1 respectively. Each NCL macro has a boolean function that describes its functionality. The boolean expression has two parts - the set and the hold expression. For e.g., in case of TH24 gate, the output expression is $F = AB + BC + AC + AD + BD + CD + F^0 (A + B + C + D)$ where AB +BC + AC + AD + BD + CD represents the set equation. The hold equation in this case is A + B + C + D. The term F'(A + B + C + D) determines the reset condition for the threshold gate [1] and account for the hysteresis behavior.

2. PROBLEM DESCRIPTION AND PROPOSED SOLUTION

Unfortunately, current fabrication methods have not been able to manufacture a defect free nanowire crossbar matrix. According to researchers, current fabrication processes have defect rates of about 10% [6] in nanowire crossbar. Scientists are yet to discover a standard fabrication technique which would have a consistent defect rate. Due to manufacturing defects, some of these ON programmable crosspoints may not be

programmable. Such a manufacturing defect may result in a stuck-at-OFF fault at a programmable location. In a 6x10 grid used to implement TH23 gate, 18 out of the total 60 potentially programmable crosspoints are used to implement the TH23 gate macro. In figure 1, a defect at the leftmost crosspoint in the first row results in a faulty function $F^{¤} = B + BC + AC + AF^{0} + BF^{0} + CF^{0}$. One or more of such faults can completely alter the functionality of the THmn gate being realized. This results in a need for functionally testing the programmable PGMB after gate mapping. The most primitive form of testing is the raw testing scheme. In this scheme, each and every point is tested for ON and OFF state separately. This is an extremely laborious method and introduces a great amount of overhead [1]. In case a 6 x 10 grid has to be tested, each of the 60 crosspoints will have to be tested ON and OFF. Another drawback of this scheme is that although a point is tested for ON/OFF state, there is still no guarantee that it is completely programmable. The raw testing scheme cannot provide complete assurance that the TH gate is functionally correct. In addition to these reasons, the testing overheads introduced in terms of time and space complexities call for a more reliable and practical form of testing the PGMBs. The prime motivation behind proposing the Functional Test Algorithm is to address these issues associated with raw testing of PGMBs. The features and advantages of using the functional test approach are discussed and illustrated with numerous examples in the proceeding sections of the paper.

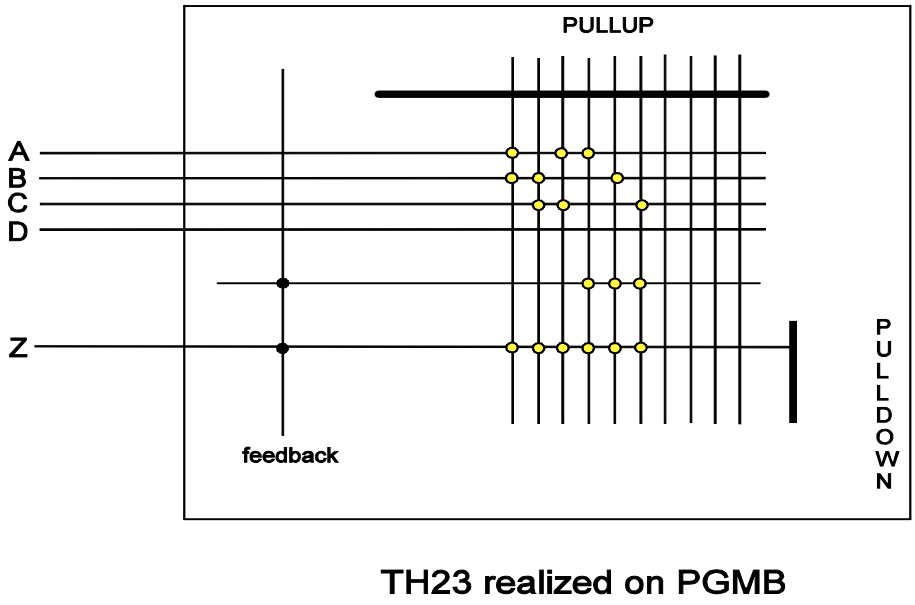

Figure 1 TH23 gate implemented on a PGMB

3. FUNCTIONAL TEST ALGORITHM

The proposed functional test algorithm is a post configuration test scheme [7] which makes use of the boolean function of the threshold gate being implemented to test the programmable crosspoint locations on the PGMB. Each THmn gate has its own distinctive programmable co-ordinate locations. The proposed test scheme aims to test only those programmable ON crosspoints in the given programmable PGMB. This algorithm uses the functional expression that is unique to each THmn gate. In figure 1 there are 18 programmable locations on the 6 x 10 PGMB. The functional test scheme uses "test tuples" for the purpose of testing the programmable crosspoints. Test tuples are joint combinations of input bit patterns and previously asserted output. Table 1 can be used to clearly understand this concept. Consider the implementation of TH23 gate.

Assume there is a fault at the coordinate location (1, 3). The fault at this ON point gives a faulty output F*=1 when input 001 is used. The desired output in case there is no fault at any crosspoint is F=F' (the previously asserted output). In case the previously asserted output is set to 0 and followed up with input pattern 001, it will be possible to stimulate the fault. The testable crosspoint coverage for each THmn gate is given in figure 2.

Table 1 Truth Table for TH23 gate and all faulty functions that can be resulted from single crosspoint defect.

ABC	F	F*(1,1)	F*(1,3)	F*(1,4)	F*(2,1)	F*(2,2)	F*(2,5)
0 0 0	0	0	0	**F'=1**	0	0	**F'=1**
0 0 1	F'	F'	**1**	F'	F'	**1**	F'
0 1 0	F'	**1**	F'	F'	F'	F'	F'
0 1 1	1	1	1	1	1	1	1
1 0 0	F'	F'	F'	F'	**1**	F'	F'
1 0 1	1	1	1	1	1	1	1
1 1 0	1	1	1	1	1	1	1
1 1 1	1	1	1	1	1	1	1

ABC	F	F*(3,2)	F*(3,3)	F*(3,6)	F*(5,4)	F*(5,5)	F*(5,6)
0 0 0	0	0	0	**F'=1**	0	0	0
0 0 1	F'	F'	F'	F'	F'	F'	**1**
0 1 0	F'	**1**	F'	F'	F'	**1**	F'
0 1 1	1	1	1	1	1	1	1
1 0 0	F'	F'	**1**	F'	**1**	F'	F'
1 0 1	1	1	1	1	1	1	1
1 1 0	1	1	1	1	1	1	1
1 1 1	1	1	1	1	1	1	1

Table 1 Truth Table for TH23 gate and all faulty functions that can be resulted from single crosspoint defect (cont'd)

ABC	F	F*(6,1)	F*(6,2)	F*(6,3)	F*(6,4)	F*(6,5)	F*(6,6)
0 0 0	0	0	0	0	0	0	0
0 0 1	F'	F'	F'	F'	F'	F'	**0**
0 1 0	F'	F'	F'	F'	F'	**0**	F'
0 1 1	1	1	**F'=0**	1	1	1	1
1 0 0	F'	F'	F'	F'	**0**	F'	F'
1 0 1	1	1	1	**F'=0**	1	1	1
1 1 0	1	**F'=0**	1	1	1	1	1
1 1 1	1	1	1	1	1	1	1

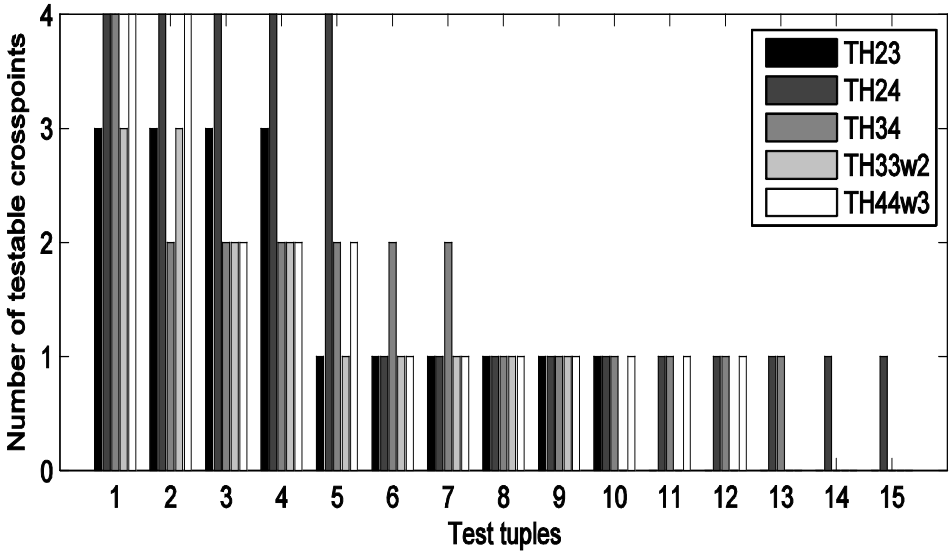

Figure 2 Testable crosspoints with each input for THmn gates

at the programmable location since the erroneous output of 1 is observed. By using a combination of F' and input bits, we can detect faults in the programmable crosspoints.

These sets of inputs used to detect the faulty crosspoint locations are called "test tuples". Test tuples having one to one correspondence with the programmable cross- points are called lower order test tuples. Higher order test tuples can test for defects in more than a single crosspoint simultaneously. The first set of 4 test tuples as shown in figure 2 each covers 3 crosspoints. The total coverage provided by the first set is 12 crosspoints. The remaining 6 test tuples each cover only 1 crosspoint and are placed in the lowest level of priority. TH23 has 2 priority levels. TH34, on the other hand, has 3 input priority levels with the first tuple testing 4 ON-crosspoints, the second set testing 2 points each (test tuple number 2, 3, and 4 test 3 crosspoints each) and finally the lowest level providing one-to-one correspondence. A pseudo code for the functional test algorithm is described in figure 3.

```
START
1. Define PGMB dimensions and threshold gate
   to be programmed.
2. Map THmn gate onto the PGMB and generate
   the corresponding truth table.
3. IF   OR plane defect detection is the prime
        objective  THEN
        Set test tuples with low priority tuples
        having direct correspondence with
        programmable crosspoints to be used
        ahead of the high priority ones.
   ELSE
        Set test tuples in order of decreasing
        priority levels.
4. Use next test input tuple to check for
   programmable defect.
5. IF    False output is observed  THEN
        Increment fault count and note possible
        defect location.
   ELSE
        The tested location is defect free.
6. Go to Step 4.
7. Summarize the results of the test.
STOP
```

Figure 3 Pseudo code describing the Functional Test Algorithm

Consider figure 4 which shows TH34w2 gate mapped onto a defective PGMB. The defect rate considered is 10 % for the worst case scenario. The circles indicate the programmable locations for the TH gate which must be programmed as ON. The stars denote programmable ON crosspoint locations overlapping the defective crosspoint locations.

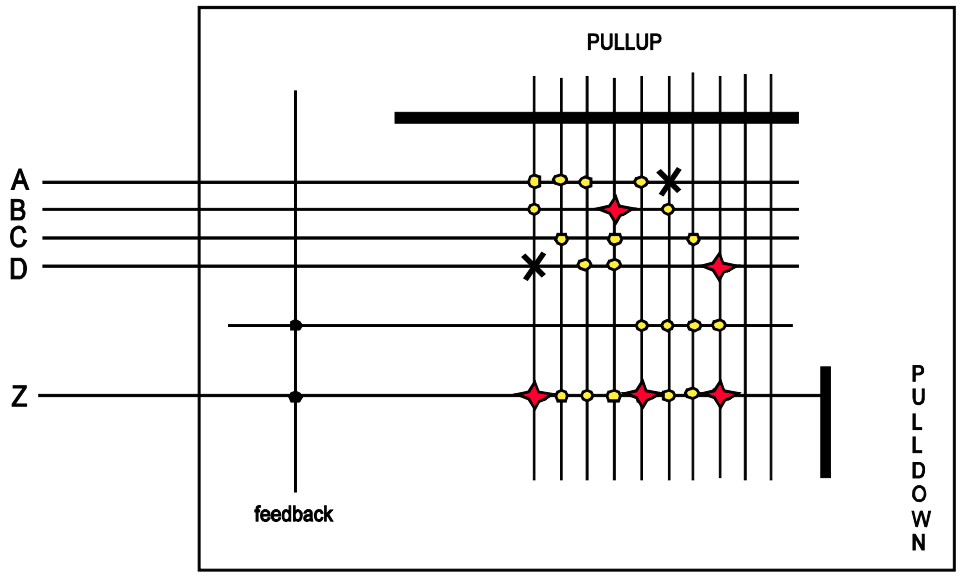

Figure 4 TH34w2 gate on a defective PGMB

The other points marked as X in figure 4 show defective crosspoint locations which will not be programmed for realizing TH34w2 gate. These locations will not alter the functional behavior since they do not overlap with the ON programmable crosspoint locations. The proposed functional test algorithm will work as follows in this case:

1. TH34w2 is mapped on to the given PGMB.
2. The set of prioritized inputs are generated.
3. The first set of prioritized test tuple (0000, 0100) tests for locations (2,1), (3,2),

(4, 3) and (5, 5). The first part in the tuple set 0000 is used to prepare the PGMB for testing and 0100 is the input pattern used to test the cross- point location.

4. The second prioritized test tuple (1111, 0000) tests (1, 5), (2, 6), (3, 7) and (4, 8). This time, the observed output is different from the desired one. This implies there is a fault at either one or more crosspoints from the set of 4 locations tested.
5. The third set (0000, 0010) is then used to test two locations, (1, 2) and (5, 7). No fault is observed.
6. The forth (0000, 0001) and fifth (0000, 0100) set also give desired results.
7. The next sets of input tuples give one to one correspondence. The next test tuple (0000, 0011) tests the crosspoint location (2, 4) which is a defective location. With one-to-one mapping present in this case, the faulty cross- point can be directly isolated.
8. Similarly, two more input tuples (0000, 0101) and (0000, 0110) are applied and all AND programmable locations are tested.
9. Once the product term locations are tested, the OR programmable plane is considered. All the OR programmable points give one-to-one mapping. Locations (6, 1), (6, 5) and (6, 8) can be successfully tested for fault.
10. Summary of Test: Out of the 5 potentially defective programmable crosspoints, 4 have been isolated successfully. These locations are (2, 4), (6, 1), (6, 5), (6, 8). There is a defect at potentially one or more locations from the following set: (1, 5), (2, 6), (3, 7), (4, 8).

4. FAULT-TOLERANT PLACEMENT SCHEMES USING THE PROPOSED FUNCTIONAL TEST ALGORITHM

Table 2 gives the number of OR locations utilized to implement few of the commonly and importantly used THmn gates.

Table 2 Table giving number of programmable OR locations for THmn gates

Number of Programmable OR Crosspoints

TH12	TH23	TH24	TH34	TH33w2	TH34w2	TH44w3
4	6	10	8	5	8	7

Having studied the mapping patterns of THmn gates and defect distributions, it has been noticed that the OR plane is vulnerable to have a physical defect overlap with a programmable ON location of any threshold gate macro. Since a majority of programmable ON crosspoints fall on a single OR plane, it is essential to ensure OR plane reliability. With the inclusion of a redundant OR wire, the reliability of the OR plane can be enhanced. With the inclusion of a redundant OR wire, in case an OR point is defective, the connection can be moved to the redundant wire without programming other crosspoints in the column which contribute to the product term. Another advantage of introducing the OR plane is that since the OR planes are ORed together, the realization is not altered in any manner. As far as testing overheads are concerned, with the addition of a redundant row, only single additional input test tuple needs to be used to test the single OR location. If redundant OR row is not introduced, in case of a defect at OR location, the entire column will have to be moved to another location and all the corresponding crosspoint locations will have to be tested using additional test tuples for defects. Not only will the number of programmable locations increase with this approach, but the testing space will also increase drastically. In case of some of the THmn gates such as TH12, TH23w2 where no more than 50% of the programmable OR locations are used, it

would be better to rearrange the columns instead of using a redundant OR row. In this section, different modeling and placement schemes that could be used to address these mapping issues are presented. Figure 5 shows the reconfigured PGMB realized on the same TH34w2 gate on the defective PGMB shown in figure 4.

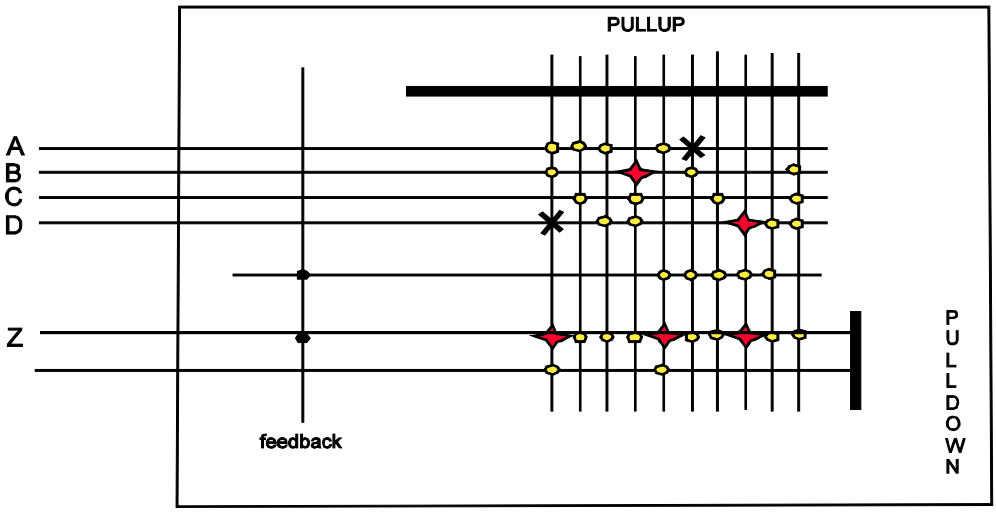

Figure 5 TH34w2 realized successfully using a redundant OR plane row and Functional Test Algorithm

Consider TH34w2 gate shown in figure 4. The functional test algorithm predicted a fault at one or more locations from the set (1, 5), (2, 6), (3, 7) and (4, 8). Incase column 5 is moved to a parallel location and functionally tested; the observed output does not match with the desired one. This implies the fault location has not been detected. With (4, 8) moved to (4, 9) and tested, the input tuple generates desired output. The entire column is moved to column 9 and tested functionally. Column 4 is moved to column 10 and results are validated using additional test tuples. The remaining 2 OR defective crosspoints with initial locations (6, 1) and (6, 5) are moved to (7, 1) and (7, 5) respectively. The

reconfigured PGMB looks as shown in figure 5. Different realizations of THmn gates over PGMBs having variable defect rates ranging from 1% to 10% have been analyzed using the functional test approach proposed in this paper.

5. PERFORMANCE ANALYSIS OF THE FUNCTIONAL TEST ALGORITHM

In this section, some figure of merits to analyze the proposed functional test algorithm will be presented. In cases where partial testing is needed, such performance measurements can help quantify the fault-coverage and fault-tolerance achieved. Accuracy is a figure of merit which has been used to quantify our test approach. Accuracy of the functional test scheme can be defined as the ratio of number of tested as bad PGMBs over the total number of actually bad PGMBs. It is evident that the accuracy ratio increases with increase in defect rate and the number of test tuples used. Consider the figure6 which gives the accuracy plot for TH23 gate. With a partial scan approach, if the accuracy is expected to be greater than 60% with a defect rate of close to 5%, this can be achieved by testing a minimum of 9 programmable ON locations. In figure 6, accuracy plots for TH23 gate with varying number of test tuples and increasing defect rates have been generated. When the prioritized test inputs are applied in reverse order with OR planes tested first, the accuracy is very low and increases slowly with each test tuple. For test tuples applied in order of decreasing priority, we can achieve higher accuracy for comparatively lesser number of tuples applied. Having said that, if location of defect is essential, then a compromise needs to be made with respect to accuracy. This is a necessary tradeoff. Another complementary factor that can be used as a performance indicator is the escape factor. Escape factor is the ratio of actually bad PGMBs over total

identified bad PGMBs. Actual bad ones are those which have been subject to all the test tuples possible to cover the entire programmable space. Total identified bad PGMBs are those which have been identified as bad when a reduced set of test tuples have been applied. This reduced set, called as N_{test} is a subset of the total test points, denoted by N_{on}, where N_{on} is the number of ON crosspoints.

Accuracy and escape factor are complementary to each other. Escape factor is greater when lesser number of test tuples is applied. For increasingly larger number of test tuples, the numbers of indeed bad PGMBs are lesser, bringing down the escape factor. A low value for escape factor means lesser the chances of an indeed bad PGMB escaping as a tested-good one.

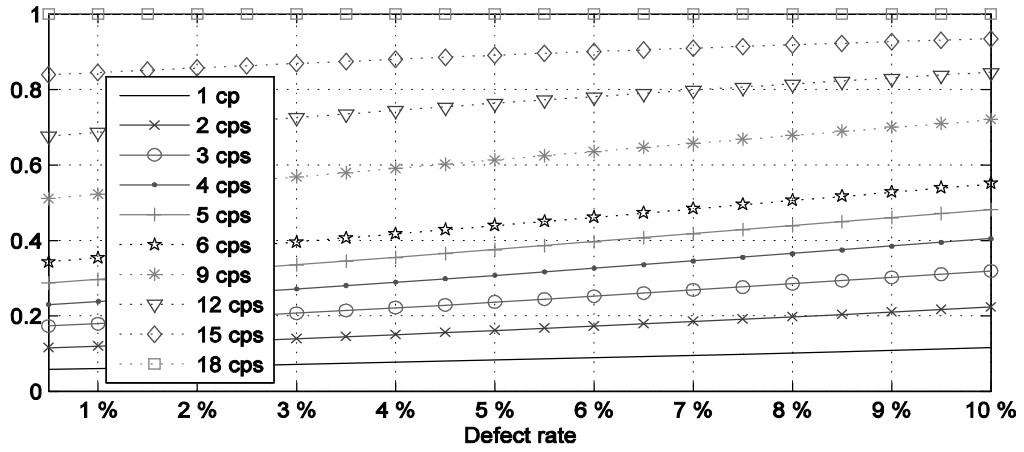

Figure 6 Accuracy plot for TH23 gate with input tuples applied in order of increasing priority

6. CONCLUSION

The proposed post configuration functional test algorithm is a definite improvement over the raw testing approach. The proposed test algorithm is applied after gate mapping is done over the PGMB. Unlike the raw testing scheme, the proposed algorithm uses only the ON programmable crosspoint locations for realizing a threshold gate. Once the algorithm is applied, alter- native placement and reconfiguration of programmable crosspoints can be done using the diagnostic results generated from the test scheme. Another merit of the proposed functional test scheme is that it provides fault cover- age. When used with alternative remodeling and placement approaches, this scheme can also provide fault tolerance. Based on the test results, rearrangement of ON crosspoints, addition of a redundant OR row or a combination of both approaches can be taken to further enhance fault tolerance.

7. REFERENCES

[1] Ravi Bonam, Yong-Bin Kim and Minsu Choi "Defect-Tolerant Gate Macro Mapping and Placement in Clock-Free Nanowire Crossbar Architecture", 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07. 26-28 Sept. 2007 Page(s):161 - 169

[2] Ravi Bonam, Shikha Chaudhary, Yadunandana Yellambalase and Minsu Choi, "Clock-Free Nanowire Crossbar Architecture based on Null Convention Logic (NCL)" 7th IEEE International Conference on Nanotechnology (IEEE-Nano), Apr 2007.

[3] Karl M.Fant and Scott A. Brabdt, "NULL Convention Logic System", US patent 5,305,463 April 19, 1994.

[4] S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb, "Optimization of NULL Convention Self-Timed Circuits", Integration, The VLSI Journal, Vol. 37, No. 3, pp. 135-165, 2004.

[5] S. Smith, R. DeMara, J. Yuan, M. Hagedorn and D. Ferguson, "Delay-Insensitive gate-level pipelining", Integration, The VLSI Journal, Vol. 30, pp. 103-131, 2000.

[6] J.Huang, M.B. Tahoori and F. Lombardi, "On the defect tolerance of Nano-Scale Two Dimensional Crossbars" IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp 96-104, Oct 2004.

[7] Sriram Venkateswaran and Minsu Choi, "Post-Configuration Testing of Asynchronous Nanowire Crossbar Architecture" 8th IEEE International Conference on Nanotechnology (IEEE-Nano), Aug 2008.

VITA

Sriram Venkateswaran was born on 10th Dec, 1985 in Mumbai, India. After completing his primary education from St. Francis D'Assisi High School, Borivali, Mumbai, he enrolled at the South Indian Education Society (SIES) Graduate School of Technology, University of Mumbai to receive his Bachelor of Engineering (B.E) degree with Distinction in Electronics and Telecommunication in July 2007. He enrolled in the department of Electrical and Computer Engineering at the Missouri University of Science and Technology (formerly known University of Missouri, Rolla) in fall 2007 to pursue his masters. During his master's program, he worked as a graduate research assistant at the Micro/Nano Computing Lab (MNCL). He graduated with master's degree in May 2009.