Built-In Self-Test of the Programmable Interconnect in Field Programmable Gate Arrays

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

Bobby Earl Dixon Jr.

Certificate of Approval:

Victor P. Nelson
Professor
Electrical and Computer Engineering

Charles E. Stroud, Chair
Professor
Electrical and Computer Engineering

Adit D. Singh
Professor
Electrical and Computer Engineering

George T. Flowers
Dean
Graduate School

Built-In Self-Test of the Programmable Interconnect in Field Programmable Gate Arrays

Bobby Earl Dixon Jr.

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
December 19, 2008

Built-In Self-Test of the Programmable Interconnect in Field Programmable Gate Arrays

Bobby Earl Dixon Jr.

Permission is granted to Auburn University to make copies of this thesis at its discretion, upon the request of individuals or institutions and at their expense. The author reserves all publication rights.

Signature of Author

Date of Graduation

Vita

Bobby Earl Dixon Jr., son of Bobby E. and Martha J. Dixon, was born in Dothan, Alabama on September 15, 1984. In the Spring of 2006, he graduated with a Bachelor of Electrical Engineering majoring in Computer Engineering from Auburn University. Upon graduating he immediately began working on his Master of Science degree at Auburn University under the advisement of Dr. Charles E. Stroud.

Thesis Abstract

Built-In Self-Test of the Programmable Interconnect in Field Programmable Gate Arrays

Bobby Earl Dixon Jr.

Master of Science, December 19, 2008
(B.S., Auburn University, 2006)

90 Typed Pages

Directed by Charles E. Stroud

Testing programmable interconnect resources in Field Programmable Gate Arrays (FPGAs) is difficult because of the large number of wire segments and switches that must be tested. The adoption of Built-In Self-Test (BIST) for programmable interconnect testing has proven to be an effective method for ensuring the fault-free status of the interconnect network for previous FPGA architectures. The BIST approaches used in previous FPGA interconnect testing relied on several assumptions to obtain adequate fault coverage within the device. With the advancement of technology and complexity of next generation FPGAs, the assumptions used in previous work can no longer be applied. New BIST approaches must be developed that alleviate these assumptions, yet still obtain high fault coverage.

BIST approaches used in previous FPGA interconnect testing are modeled and simulated for gate-level stuck-at and bridging fault coverage. New BIST approaches are proposed and also modeled and simulated. The fault simulation results are used to compare and evaluate the fault detection capabilities and effectiveness of these BIST approaches for testing programmable interconnect resources in FPGAs. A cross-couple parity approach that best

suits the Xilinx Virtex-4 FPGA architecture is chosen and implemented for routing BIST of the global double lines in the interconnect network.

Acknowledgments

First and foremost I would like to thank God for giving me the strength and determination to achieve my graduate degree. I would like to thank Dr. Stroud for his support and advice during my tenure at Auburn University during both my undergraduate and graduate studies. I would also like to thank Dr. Nelson and Dr. Singh for their contribution to this thesis by serving on my graduate committee. To all of the professors and staff that ever helped me big or small, I am forever grateful. To my research colleagues, Daniel, Lee, Jie, Brad, Jia, Mary, and Brooks, I am thankful for all your help throughout my research. To my parents, sister, and family I thank you for all the support, encouragement, and ever constant prayer that inspired me to always do my best. To my godparents, Ben and Mary, I would have never come to Auburn without your help. I owe you my education. I thank all of my friends that have been there for me throughout both the good and bad times. Without you giving me a break from research, I would have never made it to the end. Lastly, I want to thank all of my students. I learned more from you than I ever taught you. If I can give you one last piece of moral fiber I would say to stay good, study hard, and make me proud. Always remember:

"A great engineer is not one that knows everything, but is the one that is willing to learn anything." -Bobby Dixon

WAR EAGLE

Style manual or journal used Journal of Approximation Theory (together with the style known as "aums"). Bibliograpy follows IEEE Transactions.

Computer software used The document preparation package TeX (specifically LaTeX) together with the departmental style-file `aums.sty`. Tables were generated using *Microsoft Excel* and figures were drawn in *Microsoft Visio* and *Microsoft Word*.

TABLE OF CONTENTS

LIST OF FIGURES xi

LIST OF TABLES xiii

1 INTRODUCTION 1
- 1.1 Field Programmable Gate Arrays 2
- 1.2 Programmable Routing Resources in FPGAs....4
- 1.3 FPGA Programming Technologies....6
- 1.4 The Testing Problem....6
- 1.5 Built-In Self-Test 7
- 1.6 BIST for Programmable Interconnect....8
- 1.7 Thesis Statement 9

2 BACKGROUND 10
- 2.1 Xilinx Virtex-4 Architecture....10
  - 2.1.1 Virtex-4 Programmable Logic Blocks 12
- 2.2 Virtex-4 Programmable Interconnect 15
  - 2.2.1 Virtex-4 Programmable Interconnect Switch Matrix 15
  - 2.2.2 Virtex-4 Global, Local, and Dedicated Routing Resources....16
- 2.3 BIST for FPGAs 19
- 2.4 BIST for Programmable Interconnect....20
  - 2.4.1 Routing BIST Fault Models....21
  - 2.4.2 Previous Routing BIST Approaches....23
    - 2.4.2.1 Comparison-Based Counter Approaches....25
    - 2.4.2.2 Parity-Based Approaches....29
  - 2.4.3 Previous Routing BIST Approach Assumptions 30
- 2.5 Thesis Restatement 32

3 ROUTING BIST ANALYSIS 33
- 3.1 Parity-Based Routing BIST Approaches 33
  - 3.1.1 Cross-Coupled Parity-Based Approach 34
- 3.2 Linear Feedback Shift Register Based TPG/ORA Combination Approach 36
  - 3.2.1 4-bit LFSR with Internal Feedback....36
  - 3.2.2 4-bit LFSR with External Feedback 37
  - 3.2.3 8-bit LFSR with Internal Feedback....38
  - 3.2.4 8-bit LFSR with External Feedback 38

3.3 Cellular Automata Register TPG/ORA Combination Approach........................39
3.3.1 4-bit CAR with all 150 Rules..40
3.3.2 8-bit Cyclic Boundary CAR...41
3.3.3 8-bit Maximal Length Sequence CAR..41
3.4 Routing BIST Approach Fault Simulations...42
3.4.1 Parity-Based Approaches...43
3.4.2 Counter-Based Approaches 44
3.4.3 LFSR TPG/ORA Combination Approaches...45
3.4.4 CAR Simulation Results...46
3.5 Summary..48

4 VIRTEX-4 ROUTING BIST IMPLEMENTATION 49
4.1 Virtex-4 Routing BIST Approach...49
4.2 PLB Column Double Lines...51
4.3 Non-PLB Column Double Lines 62
4.4 Virtex-4 FX Device Configurations...69
4.5 Virtex-4 Routing BIST Results...71

5 SUMMARY AND CONCLUSIONS 72
5.1 Summary of Routing BIST Approaches..72
5.2 Summary of Virtex-4 Routing BIST..73
5.3 Future Work..75

BIBLIOGRAPHY 76

LIST OF FIGURES

1.1 Simple FPGA Architecture ... 3

1.2 Simple PLB ... 4

1.3 PIP Types ... 5

1.4 BIST Process ... 8

2.1 Example Virtex-4 Architecture ... 12

2.2 Virtex-4 Programmable Logic Block [17] ... 13

2.3 Virtex-4 SLICEL [19] ... 14

2.4 Virtex-4 Global Routing Resources ... 17

2.5 Global and Local Routing Resources of the PLB ... 18

2.6 Stuck-at 0 and 1 Fault Models ... 22

2.7 DOM, DAND, and DOR Fault Models ... 22

2.8 Single 2-bit Counter Approach ... 25

2.9 Dual 2-bit Counter Approach ... 26

2.10 Single 4-bit Counter Approach ... 27

2.11 Dual 4-bit Counter Approach ... 28

2.12 Original Parity-Based Approach [11] ... 29

2.13 Modified Parity-Based Approach ... 30

3.1 Cross-Coupled Parity Approach with Numbered WUTs ... 35

3.2 Internal Feedback 4-bit LFSR with Numbered WUTs 37
3.3 External Feedback 4-bit LFSR with Numbered WUTs 37
3.4 Internal Feedback 8-bit LFSR with Numbered WUTs 38
3.5 External Feedback 8-bit LFSR with Numbered WUTs 39
3.6 4-bit CAR with all 150 Rules with Numbered WUTs 40
3.7 8-bit Cyclic Boundary CAR with Numbered WUTs 41
3.8 8-bit Maximal Length Sequence CAR with Numbered WUTs 42
3.9 Parity-Based Approach Results 44
3.10 Counter-Based Approach Results 45
3.11 LFSR TPG/ORA Combination Approach Results 46
3.12 CAR Approach Results 47
4.1 Cross-Coupled Parity Implementation in Virtex-4 50
4.2 Example North Double Line Implementation 52
4.3 Example East Double Line Implementation 58
4.4 END Terminal Connection Shift 64
4.5 Non-Connecting Terminals 65
4.6 Example North Non-PLB Double Line Implementation 66
4.7 Loopbacks Used to Ensure East MID Testing 67
4.8 Example South Non-PLB Double Line Implementation 68
4.9 Loopbacks Used to Ensure West MID Testing 69
4.10 FX Configuration Generation Process 70

LIST OF TABLES

2.1 Virtex-4 Family Device Characteristics [17] 11

3.1 CAR Rules [2] 39

4.1 North Double Line Configuration 1 54

4.2 North Double Line Configuration 2 55

4.3 South Double Line Configuration 1 56

4.4 South Double Line Configuration 2 57

4.5 East Double Line Configuration 1 59

4.6 East Double Line Configuration 2 60

4.7 West Double Line Configuration 1 61

4.8 West Double Line Configuration 2 62

4.9 Summary of Virtex-4 Routing BIST Global Double Line Configs 71

CHAPTER 1

INTRODUCTION

Since the invention of the first transistor, the electronics industry has been a hotbed of technological growth. Every new generation of integrated circuits (ICs) is in some way better than the previous generation. In 1965, Intel cofounder Gordon E. Moore noticed this trend and predicted the roadmap that has guided technology improvements for the past four decades. He proclaimed that the number of transistors that can be inexpensively put on an IC is doubling about every two years [13]. Since then, industry has kept up the trend and went from ICs consisting of only a few thousand transistors to now over a billion transistors. The major problem with this type of growth is the overall reliability of the IC. Larger chips with smaller transistors have caused new types of defects to emerge and an increased chance of faults occurring in the chip [2]. The more complex the chip, the more complex and extensive the testing must be to achieve acceptable defect levels.

1.1 Field Programmable Gate Arrays

Some of the largest ICs currently on the market are Field Programmable Gate Arrays (FPGAs). A simple FPGA, as seen in Figure 1.1, is a two-dimensional array of programmable logic blocks (PLBs) that are interconnected by a network of programmable routing resources and Input/Output blocks (IOBs) [1]. FPGAs are ideal in applications where the logic of the chip might need to be changed over time [3]. Unlike an application specific integrated circuit (ASIC), an FPGA is not fabricated for a specific task. Instead it is user programmed, often many times, with the function needed at the current time. This ability to be reprogrammed over and over is one of the biggest advantages of FPGAs because it eliminates the design and development cost of fabricating custom ASICs to perform a new function. An FPGA loses its advantage over an ASIC when considering operating speeds and chip area [12]. Typically an FPGA will draw more power than an ASIC as well [21]. Another disadvantage is overall selling cost. More often than not, an FPGA will be more expensive than an ASIC. For this reason most system designs utilizing an FPGA will be limited to low volume production or prototype designs [12].

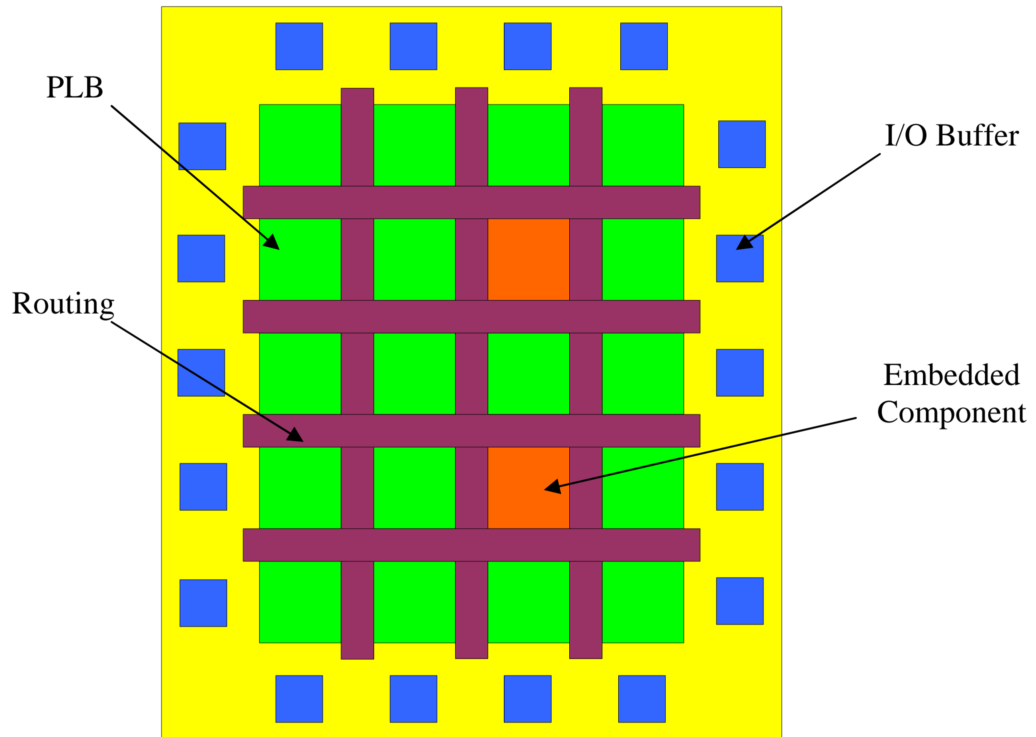


Figure 1.1: Simple FPGA Architecture

The PLBs perform most of the combinational and sequential logic functions within the chip. The routing resources connect the PLBs to other PLBs, the IOBs, and embedded dedicated components such as the random access memories (RAMs) and digital signal processors (DSPs). The IOBs handle all signal communications between the PLBs and the package pins of the chip [11]. A typical PLB, as seen in Figure 1.2, consists of logic gates, multiplexers, flip-flops, and Look-up Tables (LUTs) [15]. The logic gates and multiplexers perform combinational logic signal path selection within the PLB. The flip-flops can sometimes be configured as latches and are used to handle sequential logic functions of the PLB. LUTs, or function generators, are programmed with the truth table

of the desired combinational logic function to be performed by the PLB. They can also be configured as small RAMs in some cases [15].

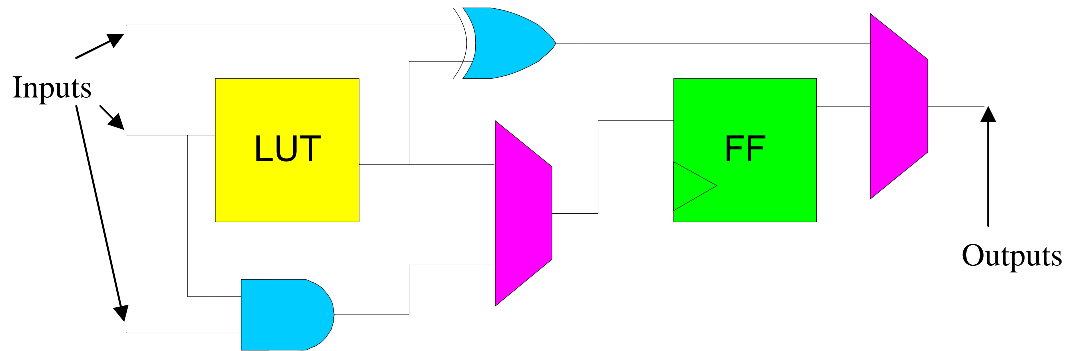


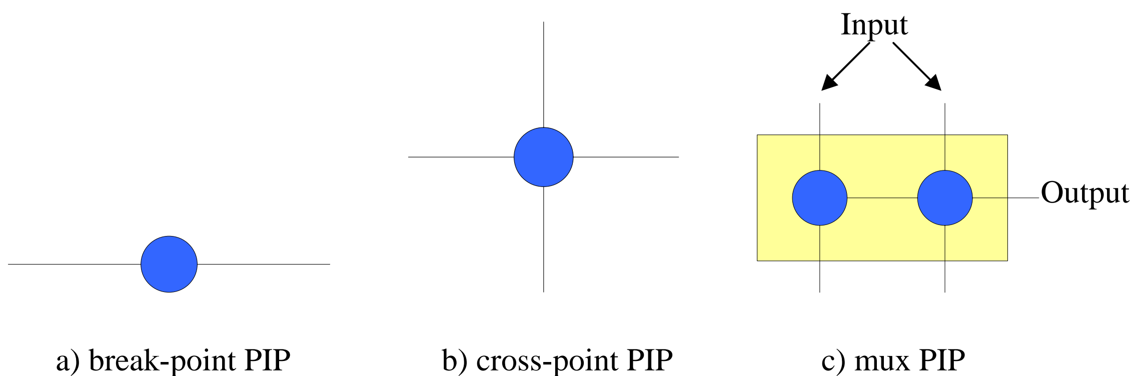
Figure 1.2: Simple PLB

1.2 Programmable Routing Resources In FPGAs

The programmable interconnect of an FPGA is typically a horizontal and vertical mesh of wire segments that are interconnected by electronically programmable switches called programmable interconnect points (PIPs) [12]. The number and length of the wire segments vary among device types and manufacturers. A balanced distribution of wire segment types is sought to maximize the functional density of the FPGA. If the balance is heavily weighted towards long wires, or global routing, then local interconnections suffer from wasted chip area and increased signal delay. Conversely, a heavy weight of short wires, or local routing, causes long routes to become congested with many PIPs. Local routing resources are used to connect PLBs to adjacent PLBs or the global routing

resources. Global routing resources are used to connect PLBs to IOBs, non-adjacent PLBs, and other embedded components.

The PIPs themselves are simple transmission gates that are controlled by configuration memory bits [5]. The FPGA is composed of several different types of PIPs: the break-point PIP, the cross-point PIP, and the multiplexer (MUX) PIP. A break-point PIP, illustrated in Figure 1.3a, connects two wire segments within the same plane; either vertical or horizontal. A cross-point PIP, illustrated in Figure 1.3b, connects vertical and horizontal wire segments. The MUX PIP, illustrated in Figure 1.3c, can either be decoded or non-decoded. A decoded MUX PIP is a group of 2^n cross-point PIPs connected to a single output wire and configured by n configuration bits [5]. A non-decoded MUX PIP has a configuration bit for every wire segment, making n configuration bits control n segments. Only one configuration bit can be active in a given configuration for a non-decoded MUX PIP. Most current FPGA routing resources are primarily made up of buffered non-decoded MUX PIPs to prevent signal degradation [5].



a) break-point PIP b) cross-point PIP c) mux PIP

Figure 1.3: PIP Types

1.3 FPGA Programming Technologies

Over time several FPGA programming methods have been implemented in chips. These methods include fuse/anti-fuse, floating gate, and static RAM (SRAM). SRAM-based FGPAs tend to be the most popular due to their fast re-programmability and standard fabrication process [12]. They are also optimal for Built-In Self-Test (BIST) implementations because of their ability to be reconfigured in-circuit an arbitrarily large number of times [1] [12]. A set of all programming bits make up the logic configuration of the PLBs [1]. These bits also configure the I/O blocks and routing resources. An SRAM-based FPGA is programmed by writing these configuration bits to the on-chip configuration memory [4]. Being able to configure FPGAs repeatedly has made them a major implementation medium in the digital design industry [15].

1.4 The Testing Problem

FPGAs of today consist of millions of transistors. With technology improvements, they are achieving greater logic capacities and higher operating frequencies [3]. The increase in FPGA functionality has made them front runners in mission critical and fault tolerant applications [10]. Due to such popularity, thorough testing is a must for these devices. The problem incurs in that FPGA testing becomes harder and more costly as FPGA architectures become more complex. Every new generation of FPGAs usually requires more test configurations to detect all faults [3]. An increase in test configurations, in turn, increases the testing time. Longer testing times

cost more money which is then reflected in the increased price of the FPGA. The use of automatic test equipment (ATE) to externally apply test vectors to FPGAs is only applicable at device-level testing [4]. Therefore, a solution must be implemented for in-system testing and fault tolerant applications.

1.5 Built-In Self-Test

One solution for in-system FPGA testing is Built-In Self-Test (BIST) [1]. BIST, as seen in Figure 1.4, usually consists of four components: a test pattern generator (TPG), a circuit under test (CUT), an output response analyzer (ORA), and a BIST controller [2]. In the case of BIST for FPGAs, the TPG consists of one or more PLBs configured to drive one or more CUTs in parallel with test vectors [8]. Embedded components such as BRAMs and DSPs have also been used to store and generate test vectors that are applied to the CUTs [15]. The CUT is merely a part of the FPGA that is being tested whether it is other PLBs, IOBs, embedded components, or routing resources. The ORA is one or more PLBs configured to analyze the output of the CUT and report a pass or fail status [2]. The BIST controller oversees the testing of the CUT by initializing and issuing the starting and ending commands to the architecture. These BIST components are programmed into the FPGA to test its resources thereby eliminating the need for external ATE. Since the BIST circuitry is on-chip, BIST also allows for at-speed testing [1].

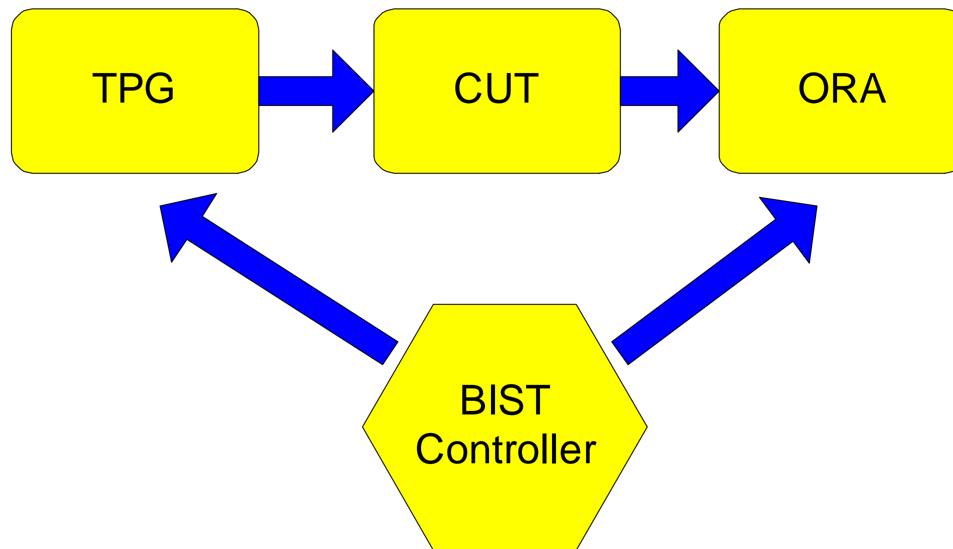


Figure 1.4: BIST Process

1.6 BIST for Programmable Interconnect

Testing the programmable interconnect network of an FPGA is a difficult process. The overall goal in routing BIST is to minimize the number of test configurations by maximizing the number of wires under test (WUT) in a given configuration [1]. Previous work in routing BIST has produced several approaches that achieve adequate fault coverage for different FPGA architectures. The first BIST approach used to test the interconnect of FPGAs utilized a counter-based TPG to drive the WUTs with test patterns that were then compared by the ORAs [1]. Another successful approach was a parity-based BIST approach that was proposed for Xilinx 4000 series FPGAs [11]. This approach also used a counter-based TPG to exhaustively drive the WUTs with test patterns, but also generated a parity bit that was sent to the ORA where a parity check

function was used to detect the faulty wires. Through the years, modifications have been made to both approaches to utilize specific FPGA architecture efficiency [5] [6] [8] [9].

1.7 Thesis Statement

The goal of this thesis is to develop and implement a BIST architecture that will perform BIST on the programmable interconnect of the Xilinx Virtex-4 FPGA. This will be achieved by performing a comparative analysis of prior routing BIST approaches. Based on this analysis, several new approaches will be proposed and evaluated. The main goal of this analysis is to choose an approach that will maximize the number of WUTs per test configuration, which will in turn minimize the overall number of test configurations. Minimizing test configurations will ultimately decrease testing time and cost. By digesting the analysis results, a suitable approach will be chosen and implemented as the routing BIST approach for Virtex-4 FPGAs.

This thesis is organized as follows: In Chapter 2, a background of previous routing BIST approaches will be given along with the theory behind programmable interconnect testing of FPGAs and the Virtex-4 architecture. Chapter 3 will discuss how multiple routing BIST approaches have been modeled and simulated for stuck-at and bridging fault coverage. The results will then be analyzed and evaluated to determine the best approach for routing BIST for the Virtex-4 FPGA. Chapter 4 will discuss how the chosen approach will be implemented for Virtex-4 routing BIST. Chapter 5 will summarize all work presented in this thesis and give insight towards future work.

CHAPTER 2

BACKGROUND

This chapter presents an overview of the Virtex-4 architecture used in this research. The primary emphasis will be focused on the programmable interconnect within this architecture. The idea behind Built-In Self-Test (BIST) will be presented, mainly focusing on programmable interconnect testing. A description of previous programmable interconnect testing techniques will also be presented along with testing assumptions that were made during their development.

2.1 Xilinx Virtex-4 Architecture

The Xilinx Virtex-4 FPGA was released in June of 2004 [16]. Claiming to be the world's most advanced FPGA at the time, the Virtex-4 FPGA incorporated a triple-oxide 90 nanometer CMOS fabrication technology with 11-layer metal interconnect [16]. Developed on the advanced silicon modular block architecture, the Virtex-4 promised to deliver twice the density and performance of any FPGA on the market at the time [16]. It was the first FPGA family to offer multiple domain-optimized platforms as presented in Table 2.1: Virtex-4 LX for functions requiring lots of logic, Virtex-4 SX for signal

processing applications, and Virtex-4 FX for embedded processing and high-speed serial applications.

Table 2.1: Virtex-4 Family Device Characteristics [17]

Device	Row x Col	Slices	DSPs	BRAMs	PPCs	I/O
XC4VLX15	64 x 24	6,144	32	48	-	320
XC4VLX25	96 x 28	10,752	48	72	-	448
XC4VLX40	128 x 36	18,432	64	96	-	640
XC4VLX60	128 x 52	26,624	64	160	-	640
XC4VLX80	160 x 56	35,840	80	200	-	768
XC4VLX100	192 x 64	49,152	96	240	-	960
XC4VLX160	192 x 88	67,584	96	288	-	960
XC4VLX200	192 x 116	89,088	96	336	-	960
XC4VSX25	64 x 40	10,240	128	128	-	320
XC4VSX35	96 x 40	15,360	192	192	-	448
XC4VSX55	128 x 48	24,576	512	320	-	640
XC4VFX12	64 x 24	5,472	32	36	1	320
XC4VFX20	64 x 36	8,544	32	68	1	320
XC4VFX40	96 x 52	18,624	48	144	2	448
XC4VFX60	128 x 52	25,280	128	232	2	576
XC4VFX100	160 x 68	42,176	160	376	2	768
XC4VFX140	192 x 84	63,168	192	552	2	896

All three families of the Virtex-4 FPGA utilize a column-based architecture as illustrated in Figure 2.1. Almost every device in the Virtex-4 product line has a different number of columns comprising its device size. All devices have a master center column that facilitates most of the secondary embedded components such as boundary scan, power management, and digital clock manager modules [17]. Moving outward from the center column are columns comprising digital signal processors (DSPs), block RAMs (BRAMs), and programmable logic blocks (PLBs) [18]. The edges of the LX and SX devices comprise the input/output buffer blocks (IOBs) [18]. The FX family also

incorporates one or two, depending on device size, IBM Power PC blocks to the left of the center column and rocket IO gigabit transceivers on the edges with regular IOBs relocated in the eighth column from the edges [17].

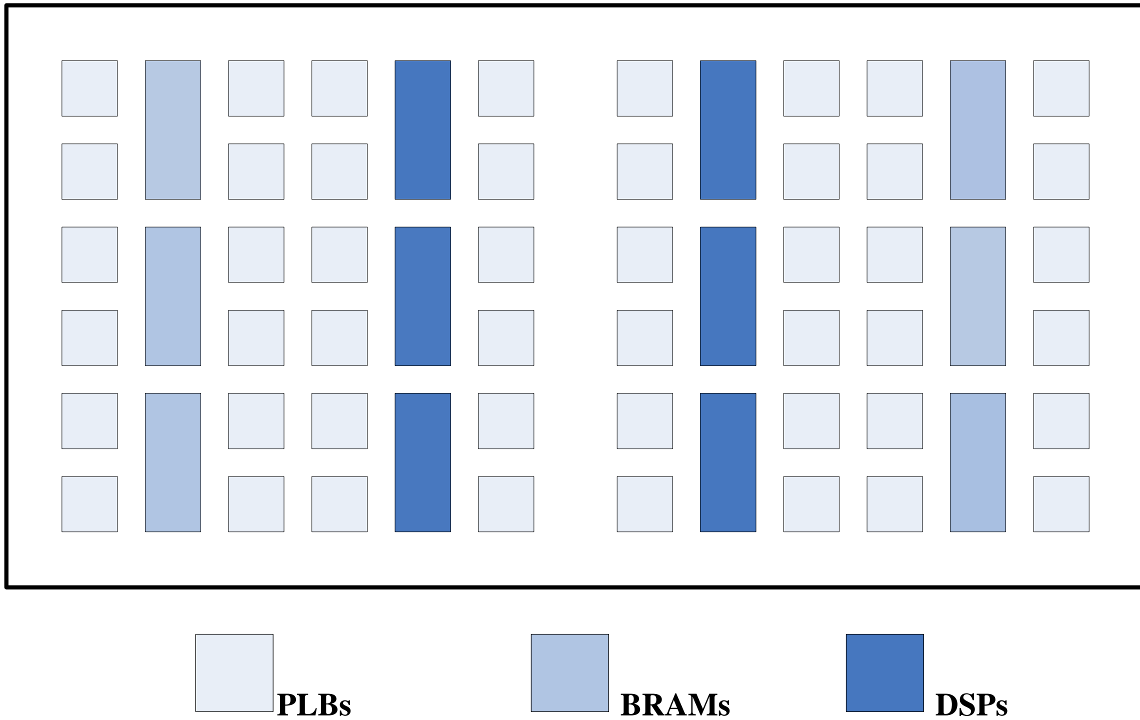


Figure 2.1: Example Virtex-4 Architecture

2.1.1 Virtex-4 Programmable Logic Blocks

The Virtex-4 PLB (Figure 2.2) consists of four SLICES; two SLICELs and two SLICEMs [17]. A SLICEL is made up of two 4-input LUTs, two flip-flops/latches, and some secondary logic gates and multiplexers (Figure 2.3) [18]. The LUTs contain the truth table of the combinational logic functions configured into a SLICE. The flip-flops/latches are used for any sequential logical functions that need to be performed by

the SLICE. The secondary logic gates and multiplexers control the internal signal routing within the SLICE as well as specialized logic functions such as fast carry logic of adders [20]. A SLICEM is a more complex SLICEL. It incorporates all of the internal characteristics of a SLICEL with added functionality to be used as shift registers or small RAMs. In this thesis, PLBs will function as the TPGs and ORAs required to test the programmable interconnect of the FPGA.

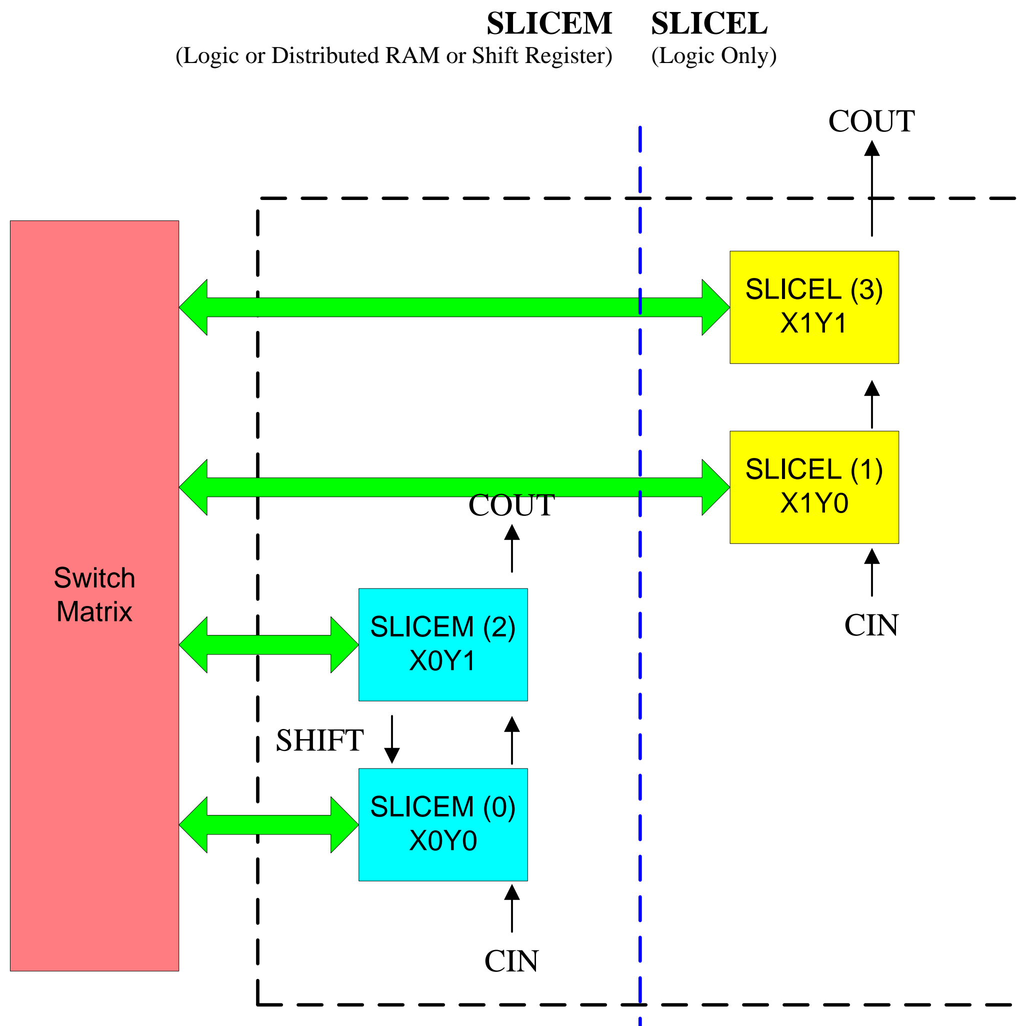


Figure 2.2: Virtex-4 Programmable Logic Block [17]

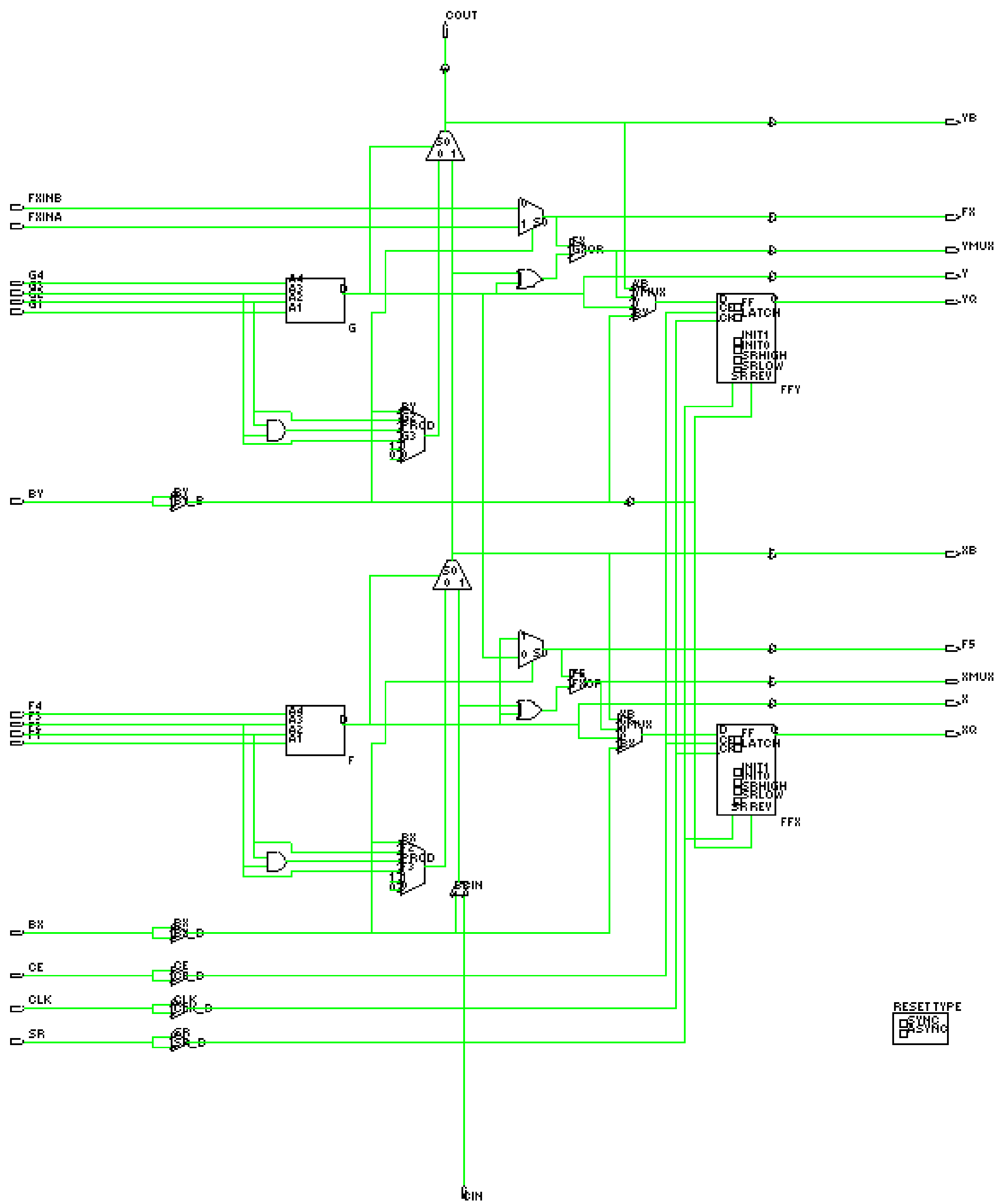


Figure 2.3: Virtex-4 SLICEL [19]

2.2 Virtex-4 Programmable Interconnect

The Virtex-4 FPGA houses an 11-layer metal interconnect [20]. The basic purpose of this interconnect network is to connect the PLBs, IOBs, and embedded components together to allow for data transfer. Typically, 80% of the configuration bits in a given bitstream control the many wire segments and programmable switches that make up the programmable interconnect network [15]. This network is made up of global routing, dedicated routing, local routing, and switch matrices. These resources are made of wire segments and programmable interconnect points (PIPs).

2.2.1 Virtex-4 Programmable Interconnect Switch Matrix

The programmable interconnect switch matrix is the gateway between all global and local routing resources within the FPGA. Every component has one or more switch matrices associated with it [19]. All signals that transfer from one PLB, IOB, or embedded component to another must route through a switch matrix. The switch matrix itself is comprised of a dense mesh of wire segments and multiplexer PIPs. There are a total of 164 multiplexers within a switch matrix, ranging from 1 to 37 inputs [19]. As a whole, a single switch matrix contains 3,312 multiplexer PIPs [19]. These internal segments and PIPs are the connections between the external global routing and the local routing of the associated component [17].

2.2.2 Virtex-4 Global, Local, and Dedicated Routing Resources

The global, local, and dedicated routing resources are the wire segments within the FPGA that are not part of the switch matrix itself. Local routing resources are used to connect PLBs to adjacent PLBs or the global routing resources [15]. Global routing resources are used to connect PLBs to IOBs, non-adjacent PLBs, and other embedded components [15].

Global routing resources can be broken down into three different wire segment types: double lines, hex lines, and long lines (Figure 2.4). The main difference in the three types of wire segments is the distance from where they begin to where they terminate within the FPGA. The biggest similarity between the double and hex lines is that they have three connections into a switch matrix along their span: a beginning (BEG) terminal, a middle (MID) terminal, and an ending (END) terminal [19]. The long lines have a total of five connections into a switch matrix along its span. The double and hex lines source in all four directions of the FPGA from a given switch matrix: north, south, east, and west. As can be seen in Figure 2.5, every direction has 10 BEGs, 10 MIDs, and 10 ENDs for a total of 240 double and hex wire segments per switch matrix [19]. The ten long lines, 5 horizontal and 5 vertical, are bidirectional and source in all four directions.

The double global routing resources span three rows or columns of components, including the starting component, with switch matrix connections at all three components (Figure 2.4) [19]. The hex global routing resources span six rows or columns of components, not including the starting component (BEG), with switch matrix connections

at the third (MID), and sixth (END) components (Figure 2.4) [19]. The long lines span twenty-four rows or columns of components, not including the starting component, with switch matrix connections at every sixth component as shown in Figure 2.4.

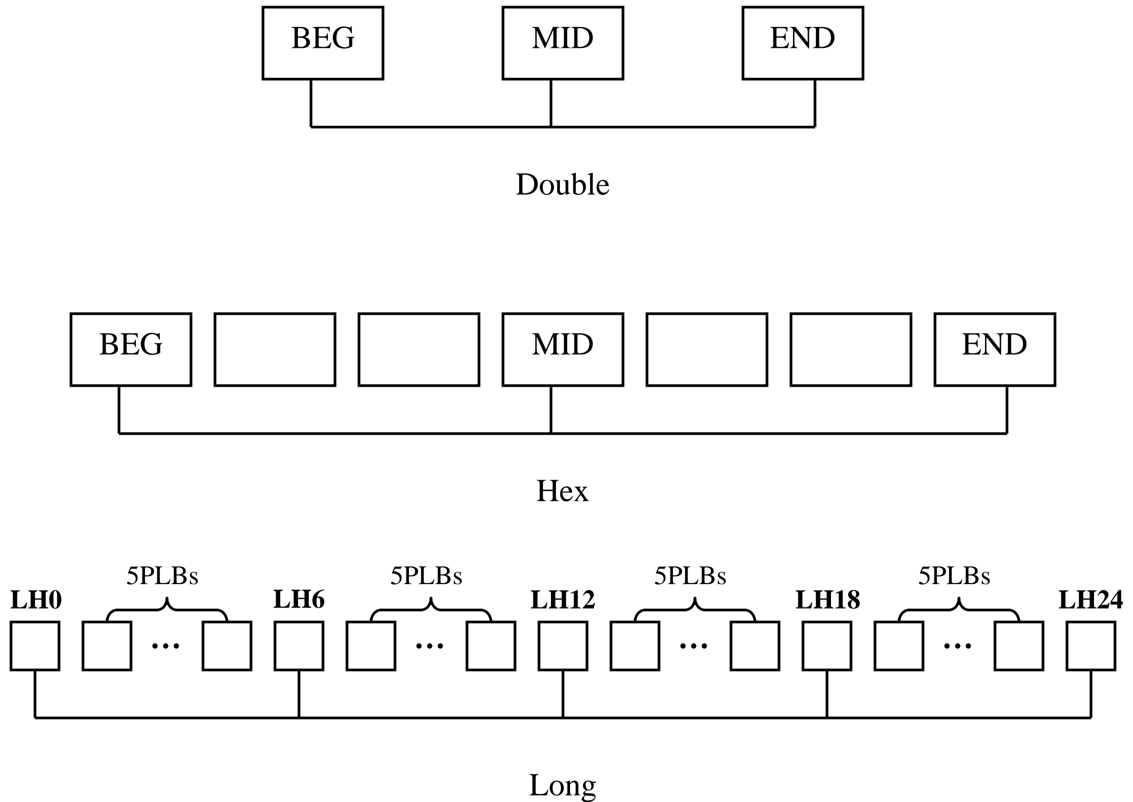
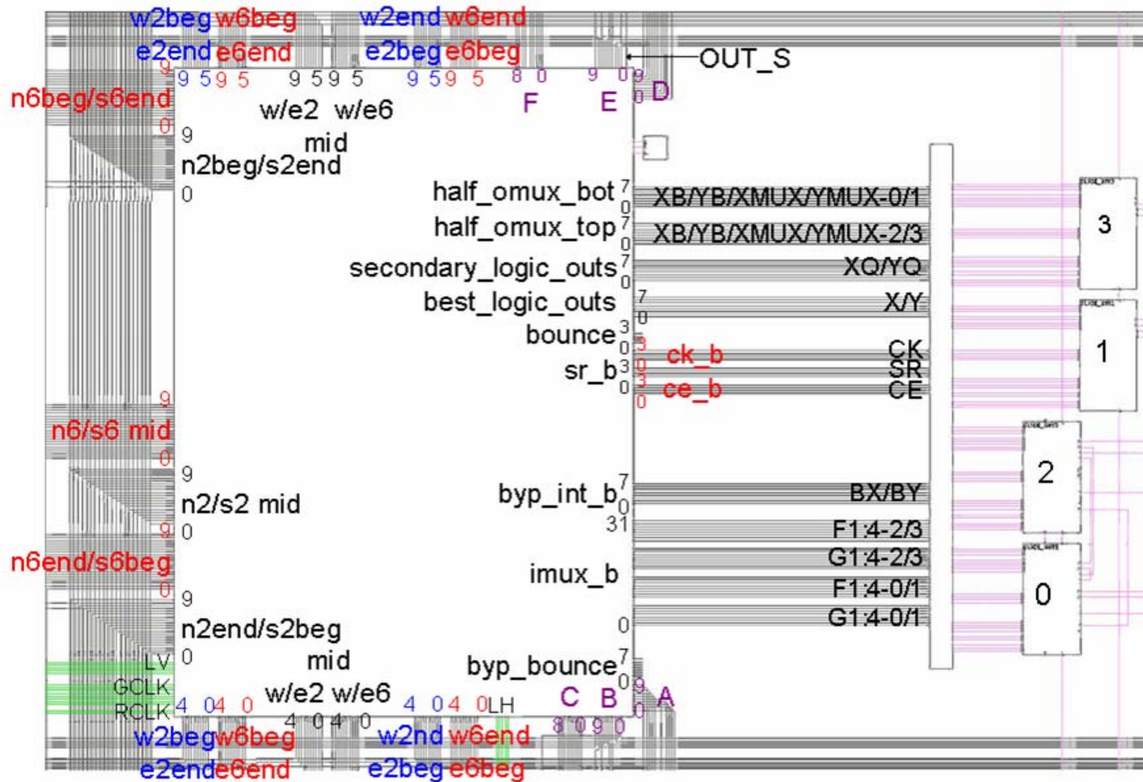


Figure 2.4: Virtex-4 Global Routing Resources

Local routing resources are the lines that connect the switch matrices to the actual PLBs and embedded components (Figure 2.5). The signals coming into the PLBs and embedded components usually connect through the input multiplexers (IMUXs) to their associated pinwire within the SLICE or component [19]. The signals leaving the PLBs and embedded components usually connect to outbound wires and enter the switch matrix to be routed onto the global resources. The flip-flops and latches within a PLB usually

have to connect first to an output multiplexer (OMUX) before it can be routed through the switch matrix to other routing resources [19].



Wire\Port	A	B	C	D	E	F
0	OMUX_NW10	OMUX2	OMUX_N13	OMUX10	OMUX13	OMUX_S2
1	OMUX_N10	OMUX_E2	W2END_N8	OMUX12	OMUX_E13	E6END_S0
2	OMUX_NE12	OMUX7	W2END_N9	OMUX_SW5	OMUX8	E6END_S1
3	OMUX_N12	OMUX_EN8	W6END_N8	OMUX_S5	OMUX_E8	E2END_S0
4	OMUX5	OMUX_E7	W6END_N9	OMUX_SE3	OMUX_ES7	E2END_S1
5	OMUX3	OMUX_W6	N6END_N9	OMUX_S3	OMUX_W9	S2END_S0
6	OMUX_N15	OMUX6	N6END_N8	OMUX15	OMUX9	S2END_S1
7	OMUX_N11	OMUX_WN14	N2END_N9	OMUX11	OMUX_W14	S6END_S0
8	OMUX4	OMUX_W1	N2END_N8	OMUX_S4	OMUX14	S6END_S1
9	OMUX0	OMUX1	-	OMUX_S0	OMUX_WS1	-

Wire\Port	LH (left to right)	LV (bottom to top)	GCLK (bottom to top)	RCLK (bottom to top)
0	LH0	LV0	GCLK0	RCLK0
1	LH18	LV6	GCLK1	RCLK1
2	LH12	LV12	GCLK2	-
3	LH6	LV18	GCLK3	-
4	LH24	LV24	GCLK4	-
5	-	-	GCLK5	-
6	-	-	GCLK6	-
7	-	-	GCLK7	-

Figure 2.5: Global and Local Routing Resources of the PLB

2.3 BIST for FPGAs

BIST for FPGAs usually involves developing test configurations where both test pattern generation and output response analysis completely test all programmable resources within an FPGA. Typically, the testable programmable resources are broken down into two groups: logic and routing resources [15]. These two programmable resource groups can be further broken down into their traditional BIST classification. Programmable logic is usually split into PLBs, IOBs, and specialized cores while programmable routing is split into local and global resources [15]. The work presented in this thesis will concentrate on the global routing resources within the programmable interconnect network of the FPGA.

Since early FPGAs consisted mainly of an array of PLBs and routing resources, the most common and easiest to understand BIST is that for PLBs; also known as logic BIST [15]. The general architecture for logic BIST is that multiple identical TPGs are configured to source test patterns to multiple identically configured PLB blocks under test (BUTs). The output responses are then sent to one or more comparison-based ORAs. The BUTs are repeatedly reconfigured to be tested in various modes of operation until they have been completely tested. The process is then repeated alternating the PLBs that acted as TPGs and ORAs with the PLBs that were BUTs. After all PLBs have been tested as BUTs, the logic BIST is said to be complete.

2.4 BIST for Programmable Interconnect

BIST for programmable interconnect, often referred to as routing BIST, is one of the most complex BISTs for FPGAs. This complexity is due to the sheer number of routing resources that have to be tested versus the number of resources that are tested in other types of BIST. Routing BIST complexity is also higher because over 80% of the configuration bits loaded into the configuration memory of the FPGA control the routing resources [15]. With this much complexity, ensuring the routing resources of an FPGA are fault-free is a difficult task. The most apparent issue is the enormous amount of wire segments and programmable switches that have to be tested. This in turn makes the total number of configurations required to test an entire FPGA quite large. For this reason, the main overall goal of developing a routing BIST approach is to maximize the number of WUTs per test configuration, thereby minimizing the total number of configurations required to test the FPGA.

One of the main focuses of the work presented in this thesis is the analysis and evaluation of previously implemented and newly proposed routing BIST approaches to be able to determine the most feasible approach to be implemented for the Xilinx Virtex-4 FPGA architecture. The criterion for this analysis is that every approach be modeled and simulated to acquire its gate-level stuck-at and bridging fault coverage. The data is then evaluated against other BIST approach data to determine which approaches have the highest fault coverage along with the most WUTs per configuration.

2.4.1 Routing BIST Fault Models

The typical fault models used in FPGA interconnect testing include wire segments stuck-at 0 and 1, shorted wires, and open wires [2]. Stuck-open (stuck-off) and stuck-closed (stuck-on) PIP faults are also included because they correspond to stuck-at faults in the configuration memory bits that control the PIPs [2]. For the work presented in this thesis, only hard faults are considered, therefore delay faults are not part of this analysis. Delay faults are faults that allow the circuit to function normally but at a reduced clock rate [3].

The stuck-at 0 and 1 fault models are simply faults in which one end of the wire segments behaves as if disconnected and tied low to Vss or high to Vdd, respectively (Figure 2.6 a,b) [2]. Shorted wires are a characteristic of bridging faults. These fault models most commonly include dominant (DOM), dominant-AND (DAND), and dominant-OR (DOR) fault models (Figure 2.7 a,b,c) [2]. A DOM bridging fault is when two adjacent wires incur a low resistance short and the stronger driving gate dominates the short for all logic values. DAND and DOR bridging faults occur when two adjacent wires incur a low resistance short and the stronger driving gate dominates the short for only one logic value thus resembling an AND and OR logic gate connection, respectively. Since the number of bridging faults possible in an FPGA, if exhaustively tested, is exponential, most bridging fault testing is mainly focused on WUTs that are adjacent to each other [7].

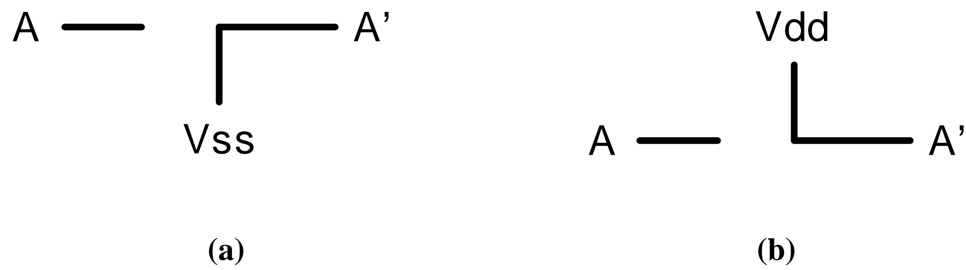
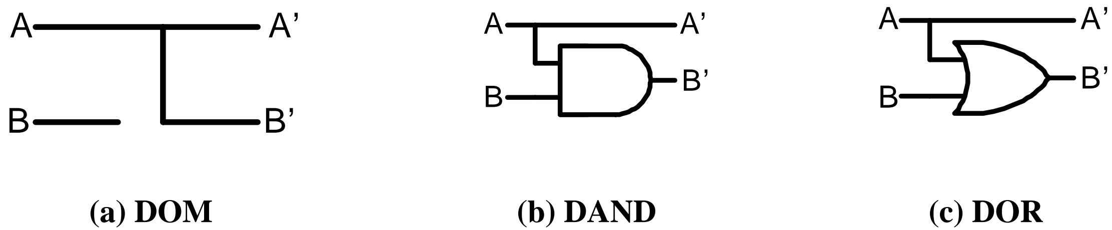


Figure 2.6: Stuck-at 0 and 1 Fault Models



(a) DOM **(b) DAND** **(c) DOR**

Figure 2.7: DOM, DAND, and DOR Fault Models

Detecting these hard faults in FPGA programmable interconnect is straightforward [5]. Every wire segment and PIP put under test must be able to transmit both a 0 and a 1. Every pair of wire segments that could possibly short must be able to transmit both (0, 1) and (1, 0) pairs. By applying both logic values to one end of a wire segment and observing those values on the other end, any stuck-at fault that exists on any wire segment to which the test is applied can be detected. This method also detects any wire open or stuck-open faults for any activated PIP along the wire segment under test. Applying opposite logic values at the ends of wire segments connected by a deactivated PIP while monitoring both sides of the PIP will reveal a stuck-closed fault for that PIP. Testing the MUX PIPs requires a separate test configuration for every one of its inputs [5]. The selected input gets both logical 0 and 1 values applied to it while the non-

selected inputs get the opposite values. This method detects if a stuck-open fault is in the PIP that connects the selected input with the output of the MUX PIP. It also detects if stuck-closed faults are present on the PIPs that are associated with the non-selected inputs.

2.4.2 Previous Routing BIST Approaches

Throughout the past work in routing BIST, several assumptions have been made to achieve adequate fault coverage for the target FPGA architectures. Since the TPGs and ORAs are constructed from PLBs, and routing resources are needed to connect the logic that forms them, the biggest assumption has been that the routing resources used are fault-free when testing PLBs and that the PLBs used are fault-free when testing routing resources [22]. In consideration of possible bridging fault sites, knowledge of physical positions of wire segments was assumed. These assumptions benefit in making routing BIST easier to comprehend, but fall to the question of how accurate is routing BIST interconnect testing.

One important issue to consider in previous routing BIST approaches is the way in which ORA results are retrieved. There are many methods for retrieving the test results from the ORAs [2], but only two were widely used in previous routing BIST approaches. A comparison-based ORA with integrated scan chain was used in ORCA [5] interconnect testing. The scan chain was used to shift the test results out of the ORAs at the end of the BIST cycle. While this method of result retrieval was very effective and fast, its major drawback was the requirement of additional logic for the multiplexer and

routing resources to construct and operate the scan chain. The increase in resources needed for the integrated scan chain within the ORA decreased the number of routing resources that could be tested in a given configuration, thereby increasing the number of test configurations required for complete testing. The only time the integrated scan chain ORA did not increase the number of test configurations was in the case of the Cypress Delta39K where a built-in scan chain was already associated with the flip-flops [7], and in the case of Atmel [6] where the shift register was constructed via dynamic partial reconfiguration at the end of the BIST sequence.

The second method used for routing BIST results retrieval is configuration memory readback. This approach was used for the Xilinx 4000 and Spartan series FPGAs [8]. Configuration memory readback in essence requires that the entire configuration memory be read for every BIST configuration to extract the results from the flip-flops within the ORAs. The drawback of configuration memory readback is that it basically doubles testing time since reading back the memory takes roughly the same amount of time as writing a configuration to it. The advantage comes in that the comparison-based ORAs no longer need a scan chain and use less logic and routing resources thereby freeing these resources to be tested in fewer configurations.

The growing trend in routing BIST results retrieval for new FPGAs is partial configuration memory readback [9]. This capability allows for only the portions of the configuration memory that contain the ORA results to be read. This method of reconfiguration and results retrieval has been proven to reduce download time by a factor of 4 and readback by a factor of 2 [9]. The work presented in this thesis uses this method of ORA results retrieval for the Xilinx Virtex-4 FPGA.

2.4.2.1 Comparison-Based Counter Approaches

The first routing BIST approach for FPGA interconnect testing used counter-based TPGs to generate the test patterns. The patterns were transmitted along the WUTs and compared by the comparison-based ORAs [1]. Many devices used this approach [1] [8] [9] [10]. The architecture of the device being tested was usually the deciding factor in how many and what size of counters would compose the TPGs. The typical composition of the counter-based TPGs would either be single or dual 2 or 4-bit counters. The single 2-bit counter TPG worked by using the current and next state of the counter as the test pattern sequence to provide four signals with opposite logic values between any pair of signals (Figure 2.8) [8]. A total of eight wires were under test using the single 2-bit counter TPG because the four signals were fanned out to the WUTs before routing to the ORAs.

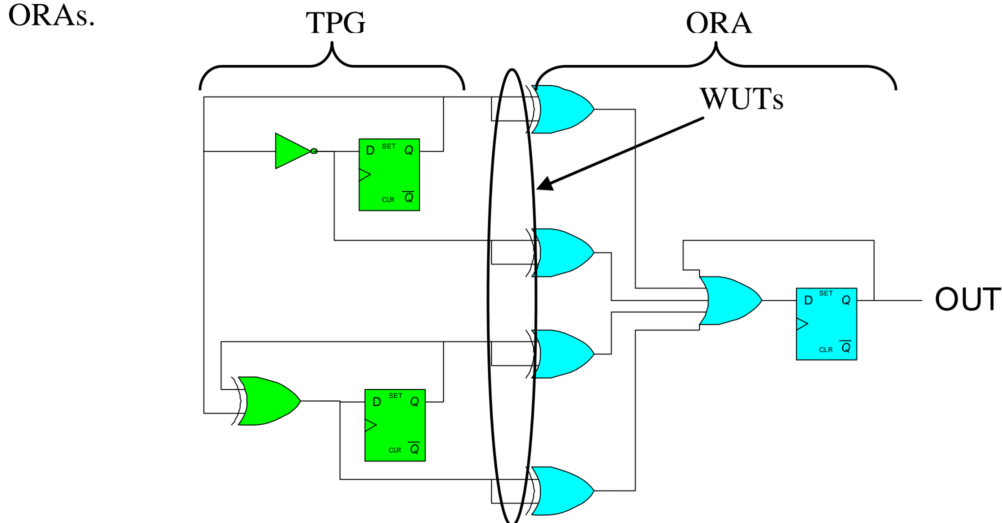


Figure 2.8: Single 2-Bit Counter Approach

The dual 2-bit counter TPG approach (Figure 2.9) was also used in FPGAs [8], and worked off the same principal as the single 2-bit counter TPG approach. The current and next states served as the test pattern sequence. The only difference was that the four signals did not have to be fanned out to the eight WUTs since the dual counters provided the eight sources independently.

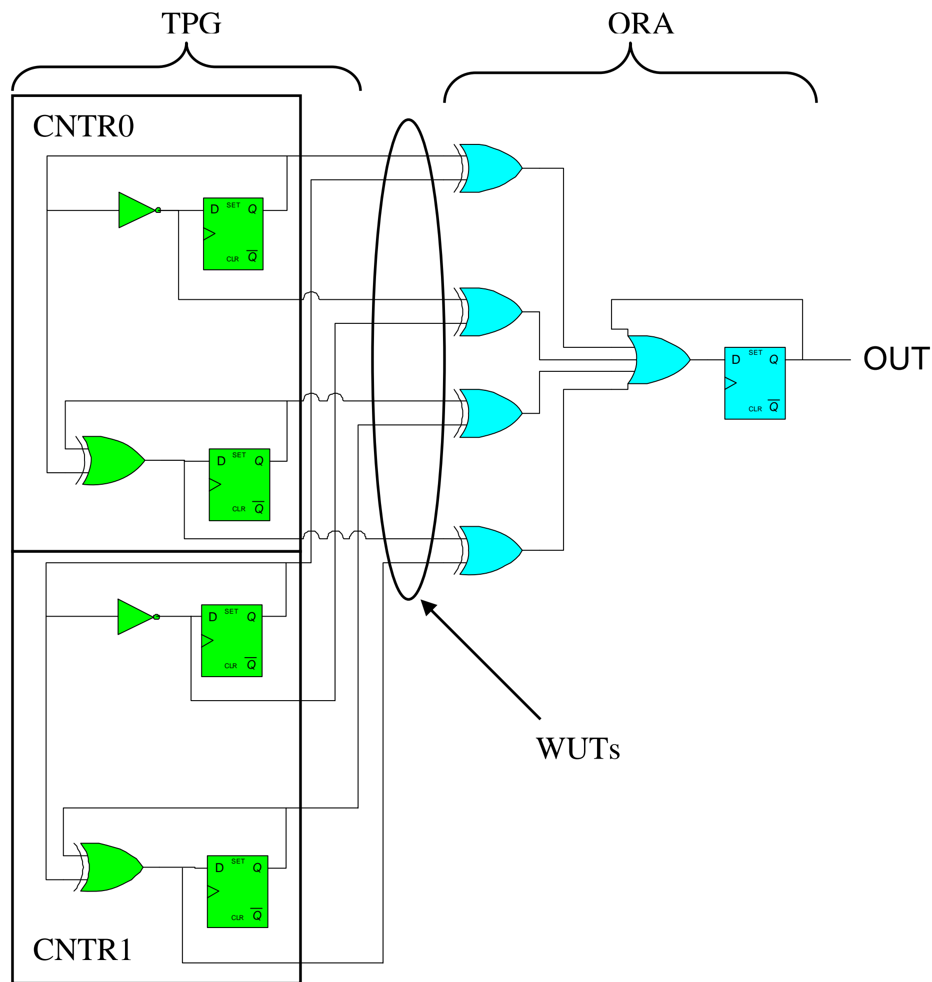


Figure 2.9: Dual 2-Bit Counter Approach

Single 4-bit counter TPGs were used in FPGAs where the PLBs contained four flip-flops [1] [9] (Figure 2.10). They worked by using only the current state of the counter as the test pattern sequence to provide the four needed signals with opposite logic values between any pair of signals. Similar to the single 2-bit counter TPG approach, the single 4-bit counter also had its signals fanned out to the WUTs before routing to the ORAs to test a total of eight wires under test.

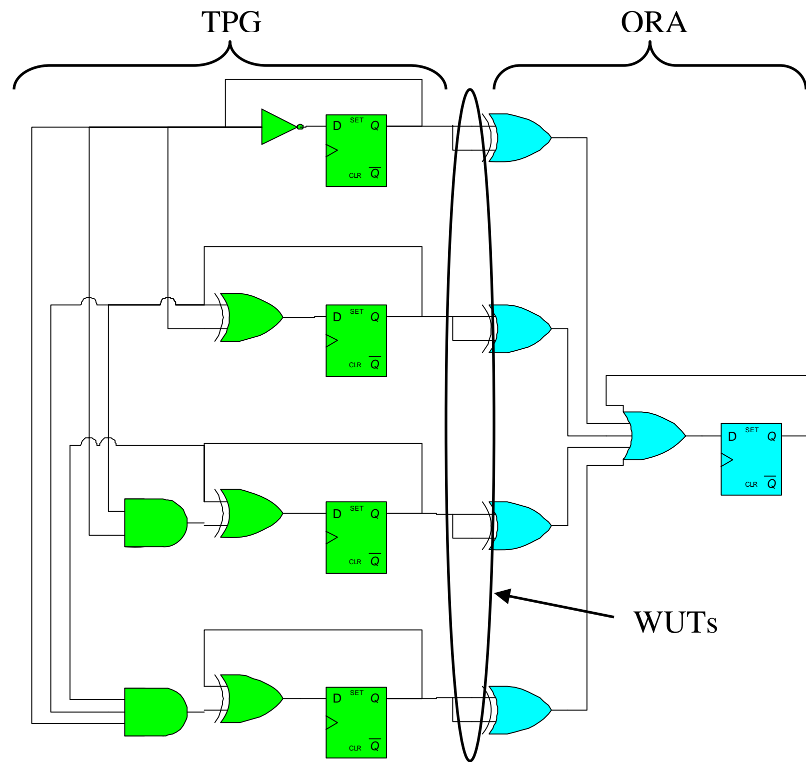


Figure 2.10: Single 4-Bit Counter Approach

The dual 4-bit counter TPG approach (Figure 2.11) was also used in larger FPGAs [1] [9], and worked off the same principal as the single 4-bit counter TPG approach. Only the current state of the counter was used as the test pattern sequence. The difference

between the 4-bit counters was the same as the 2-bit counters; the test patterns were not fanned out to the WUTs.

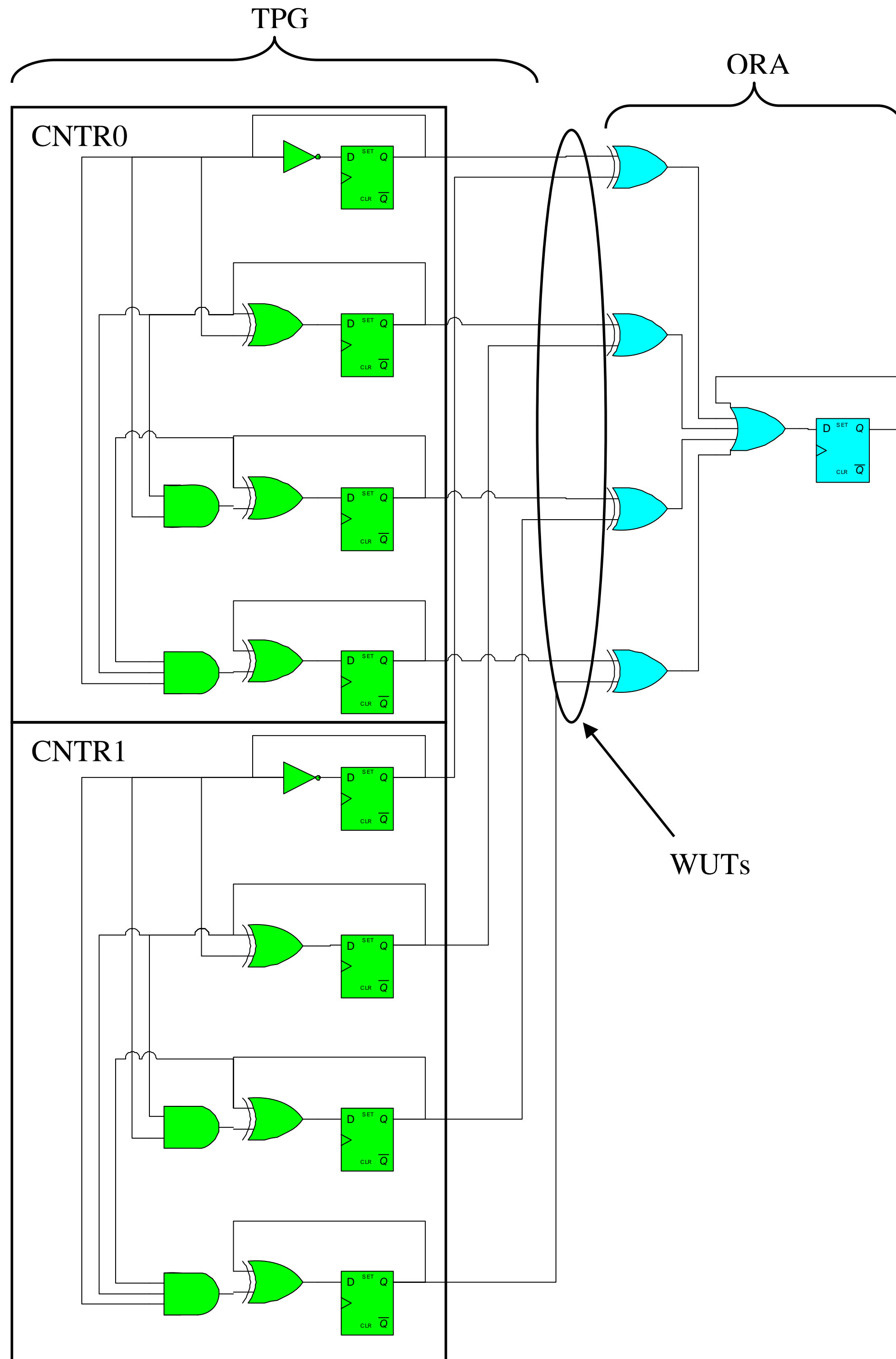


Figure 2.11: Dual 4-Bit Counter Approach

2.4.2.2 Parity-Based Approaches

Another routing BIST approach for FPGA programmable interconnect testing was a parity-based approach. It was proposed for the Xilinx 4000 series FPGAs in [11] (Figure 2.12). The idea behind this approach was that an *N*-bit counter TPG would exhaustively source test patterns across *N* wires under test that were connected to ORAs. The TPG would also generate a parity bit that would be sent to the ORAs as well. The ORAs would then perform a parity check function based on the values on the WUTs thereby detecting any faults for mismatched parity. The big disadvantage of this approach was that it assumed that the parity bit was being transmitted over fault-free routing resources.

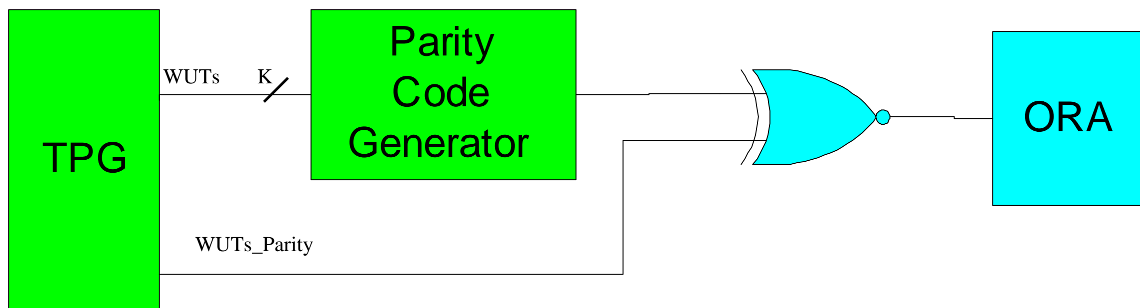


Figure 2.12: Original Parity-Based Approach [11]

This assumption was not realistic so the approach was modified in [6]. The approach would now send the parity over a WUT for a total of *N*+1 WUTs (Figure 2.13). This modification also allowed for greater flexibility in using different types of TPGs and ORAs. The parity approach really came into use in testing FPGAs that had small PLBs.

This new parity approach was used in the testing of Atmel AT40K and AT94K series of FPGAs [6]. In those tests, a 2-bit up counter, initialized to all 0s and generating even parity, was combined with a 2-bit down counter, initialized to all 1s and generating odd parity, to facilitate the testing of six wires at once per TPG/ORA combination. The count and parity signals were routed to opposite sides of deactivated PIPs to detect the stuck-closed faults within those PIPs. The opposite logic values produced between any pair of the six signals detected bridging faults between adjacent wire segments as well. This approach will be referred to as "dual parity" in this thesis.

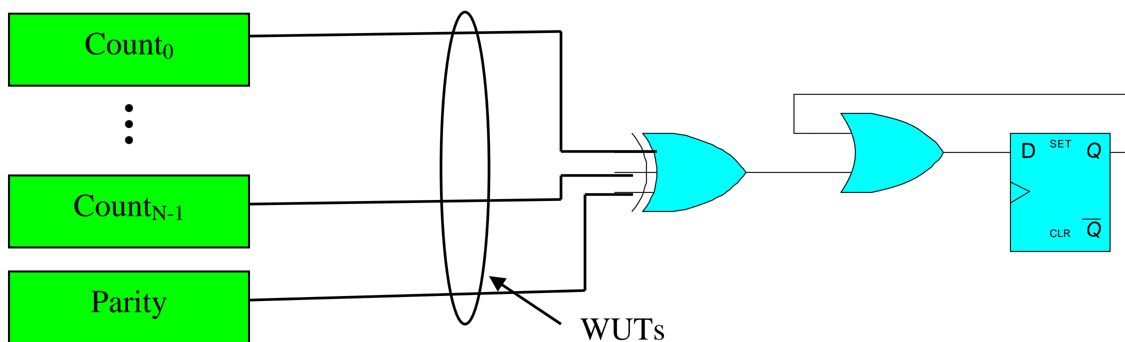


Figure 2.13: Modified Parity-Based Approach

2.4.3 Previous Routing BIST Approach Assumptions

All of the previous routing BIST approaches shared several common assumptions to be able to justify test validity [15]. One of these assumptions was that the logic resources used to create the TPGs and ORAs had been previously tested and found to be falt-free. The problem with this assumption is that in testing the logic resources for

faults, faulty routing resources could be used to connect the logic components. This in turn required that the same assumption be made in reference to the routing resources being used. A second assumption made in all of these routing BIST approaches was that the feedback routing used by the counter TPGs was fault-free. A third assumption, especially with older FPGAs, was that knowledge of the relative position of wire segments to be tested was present and therefore guaranteed that the same signal was not transmitted over adjacent wire segments. The first and second assumptions have to be used for the single counter TPGs of the comparison and parity-based routing BIST approaches. These assumptions can be alleviated depending on the method and procedure of ORA results retrieval. For instance, a logic fault that would prevent a single counter TPG from counting will go undetected along with any faults it would otherwise detect in the WUTs. However, by using configuration memory readback to retrieve the ORA test results and the current state of the TPGs after completing the entire BIST sequence and advancing past the starting state, any logical fault that could have occurred will be detected. Logical faults are not a problem in the case of using dual counters with the comparison-based approach because the TPG signals are not being fanned out to the WUTs so the ORAs will catch any differences in state between the two counters. The second and third assumptions were tolerable for the targeted FPGAs under test because they were considered small devices by today's standards, and they all had dedicated feedback routing. The number of wire segments and PIPs per PLB were very low. The number of wire segments would range from 42 to 77 segments per PLB [15]. The number of PIPs ranged from 128 to 206 per PLB [15]. Since these devices were relatively small, information about the positions of the routing resources could be

obtained from the device datasheets and physical design graphical editors. In contrast, today's FPGAs do not have dedicated feedback routing resources. Many have around 406 wire segments and 4,100 PIPs per PLB [15]. Therefore, it is imperative that new routing BIST approaches be developed to eliminate the typical testing assumptions so future, more complex FPGAs will be testable.

2.5 Thesis Restatement

The goal of this thesis is to develop and implement a BIST architecture that will perform BIST on the programmable interconnect of the Xilinx Virtex-4 FPGA. This will be achieved by performing a comparative analysis of prior routing BIST approaches. These previous approaches were at a disadvantage on the validity of their testing because of the assumptions that had to be made to achieve adequate fault coverage. The work presented in this thesis will propose new approaches that minimize these assumptions while still maximizing the number of WUTs per test configuration and achieving high fault coverage. Chapter 3 will introduce the new approaches that were proposed for Virtex-4 routing BIST. It will also present the simulations and fault coverage of some of the approaches discussed in this thesis along with the comparative analysis and evaluation of such data. Chapter 4 will show the chosen routing BIST approach for Virtex-4 global interconnect testing. It will also present the actual implementation and application of the global routing BIST for Virtex-4. Chapter 5 will conclude with a summary of the work presented and suggestions for work to be done in the future.

Chapter 3

Routing BIST Analysis

New approaches for BIST of programmable interconnect in FPGAs will be proposed in this chapter. Some of the previous and new routing BIST approaches were modeled, and simulated, to determine gate-level stuck-at and bridging fault coverage, with the Auburn University Simulator (AUSIM) [23]. This information is analyzed and discussed in this chapter to determine the best approach for Virtex-4 routing BIST implementation.

3.1 Parity-Based Routing BIST Approaches

The original parity-based routing BIST approach was proposed in [11], and later modified in [6] to allow the wire being used for parity transmission to also be a WUT. With the growing complexity of FPGAs, the assumptions that were used in testing the programmable interconnect utilizing these approaches are quickly becoming unrealistic. The most impractical assumption in the case of recent FPGAs was that the routing resources used for the feedback of the TPGs were assumed to be fault free. For example, since Virtex-4 FPGAs have no dedicated feedback routing resources, those which are

assumed to be fault-free are in essence part of the same wires that are to be tested. For that reason, the parity-based routing BIST approach had to be rethought to better test these new complex FPGAs.

3.1.1 Cross-Coupled Parity-Based Approach

With the absence of dedicated feedback routing resources in Virtex 4 FPGAs, a fault that inhibits the count sequence could go undetected if the parity bit remains the correct value. A work around for this issue is to read the current state of the TPGs along with the ORA results, using partial configuration memory readback. However, by cross-coupling the parity lines to the ORAs, the need to read the current state of the TPGs can be totally eliminated since a fault in the TPGs will be detected by the ORAs, as will be shown in this chapter.

The cross-coupled parity-based routing BIST approach is similar to the approach used in the Atmel series FPGA programmable interconnect testing [6]. It consists of one 2-bit up-counter initialized to all 0s and one 2-bit down-counter initialized to all 1s driving two ORAs each (Figure 3.1). The next state of the most significant bit of both counters is the parity and is cross-coupled to the ORAs that analyze the count values of the other counter. This cross-coupling of the parity bit allows for any fault that would hinder the count of one of the counters to be detected by the ORAs, thereby eliminating the assumption of fault-free feedback routing resources within the construction of the TPG. This added detection of faults within the feedback of the counters of the TPG also allows for those resources being used as feedback to be considered as WUTs. This set of

feedback WUTs includes both the feedback path of each counter to its driving LUT along with the path to the next bit of the respective counter. The feedback WUTs give three WUTs per counter within the construction of the TPGs. This along with the six other WUTs being driven by the TPGs allows the cross-coupled parity-based routing BIST approach to test 12 wire segments within a single test configuration. Since two ORAs will fit into a single Virtex-4 slice, the TPG signals could be fanned out to drive six more WUTs, bringing the total number of WUTs possible to 18.

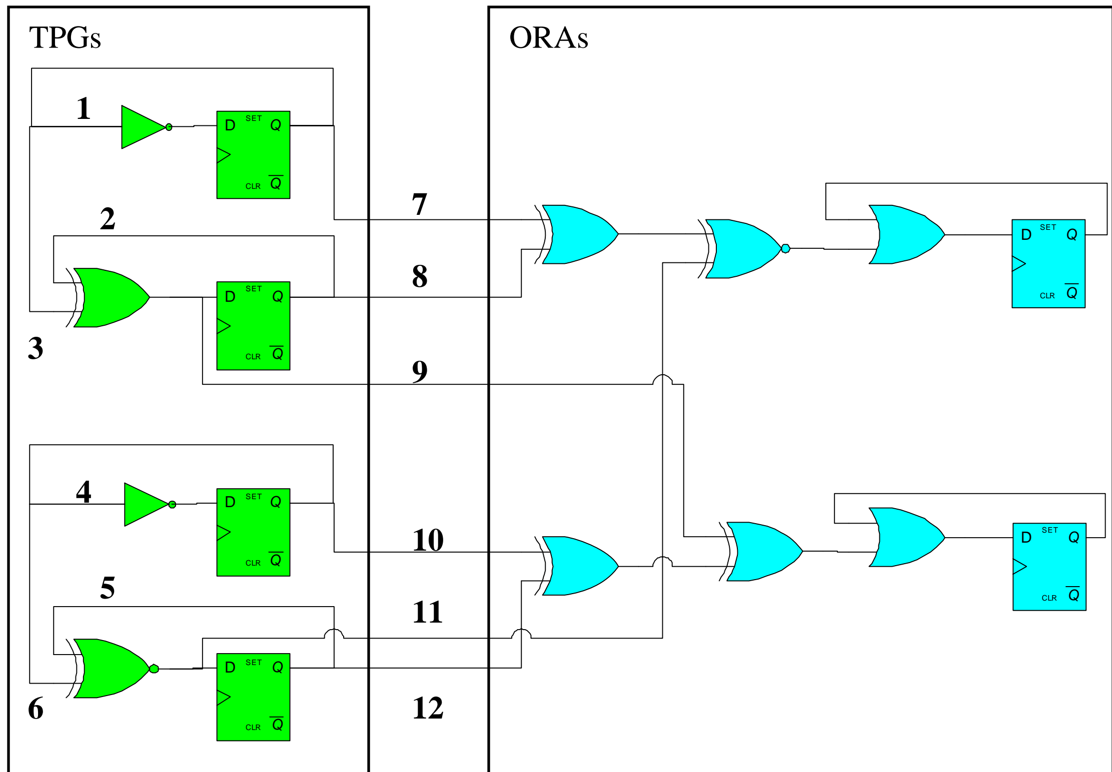


Figure 3.1: Cross-Coupled Parity Approach with Numbered WUTs

3.2 Linear Feedback Shift Register Based TPG/ORA Combination Approach

In modeling the parity-based approaches, it was observed that the feedback routing resources of counter-based TPGs can be considered wires under test. One idea is that the counter itself can be considered as both a TPG and its own ORA. At the end of the designated BIST sequence, typically after cycling through the complete count sequence and continuing to a different count value from the starting value, the current state of the counter can be read via partial configuration memory readback. The counter must then be clocked several more times, and the current state read again to determine the pass or fail status of the test. To eliminate the need to read the configuration memory twice to retrieve BIST sequence results, a primitive polynomial linear feedback shift register (LFSR) can be used for the basis of a TPG. The thought is that by reducing the number of configuration memory readbacks, total testing time can be reduced.

3.2.1 4-bit LFSR with Internal Feedback

An example LFSR-based TPG/ORA combination approach is a 4-bit primitive polynomial LFSR with internal feedback (Figure 3.2). It is clocked for an arbitrary number of clock cycles before the BIST results are read back. The main consideration to keep in mind is that the number of clock cycles has to be at least enough to let every bit of the LFSR toggle its value. This approach yields five WUTs. However, since two LFSRs can be placed in a single Virtex-4 PLB the total number of WUTs would be ten as long as the shift register mode of the LUT is not used to construct the LFSR.

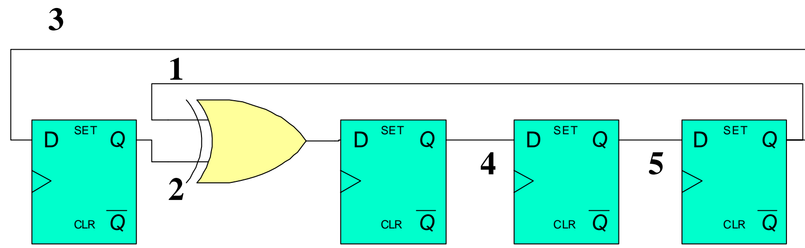


Figure 3.2: Internal Feedback 4-bit LFSR with Numbered WUTs

3.2.2 4-bit LFSR with External Feedback

It is possible to modify the approach to a 4-bit primitive polynomial LFSR with external feedback (Figure 3.3). Again, the LFSR has to be run enough clock cycles to allow all bits to have toggled before BIST sequence results can be retrieved. With external feedback, the total number of wires under test is still five with the potential of ten in a single Virtex-4 PLB as long as the shift register does not use dedicated routing.

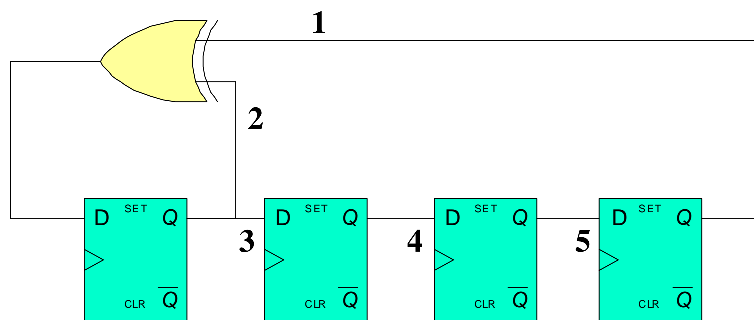


Figure 3.3: External Feedback 4-bit LFSR with Numbered WUTs

3.2.3 8-bit LFSR with Internal Feedback

To increase the number of WUTs one can increase the size of the LFSR to an 8-bit primitive polynomial LFSR with internal feedback (Figure 3.4). The thought is that by increasing the size of the LFSR that the number of feedback paths will increase thereby increasing both WUTs and fault coverage. While this approach does increase the number of wires being tested to 11 for a single LFSR, it requires a full PLB and as a result is no different than the dual 4-bit LFSR.

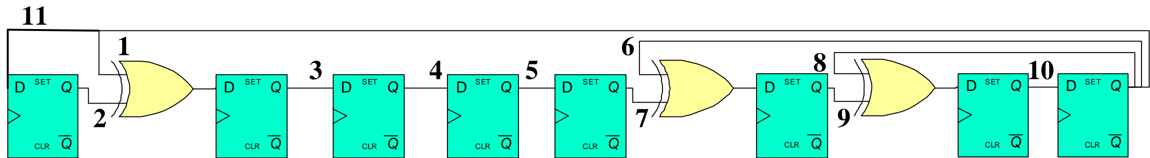


Figure 3.4: Internal Feedback 8-bit LFSR with Numbered WUTs

3.2.4 8-bit LFSR with External Feedback

It is possible to modify the approach to an 8-bit primitive polynomial LFSR with external feedback (Figure 3.5). The number of wires under test remains 11. This is because the polynomial has not changed even though the routes have.

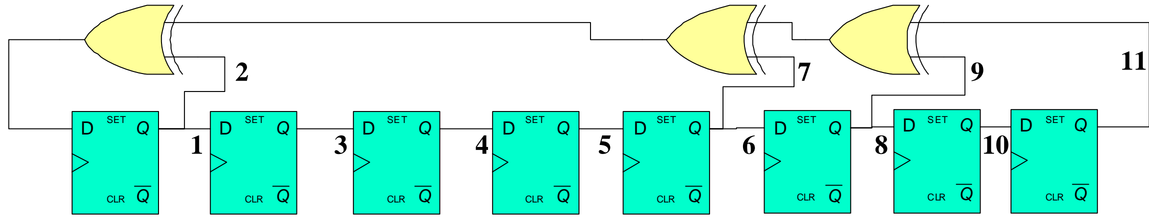


Figure 3.5: External Feedback 8-bit LFSR with Numbered WUTs

3.3 Cellular Automata Register TPG/ORA Combination Approach

A cellular automata register (CAR) TPG/ORA combination approach was investagated. Much like the LFSR, the CAR utilizes pseudo-random test pattern generation as the basis of its BIST sequence. Instead of polynomials that govern the architecture of LFSRs, the architecture of CARs is governed by rules. The most widely used rules in the construction of CARs are rules 90 and 150 as described in Table 3.1.

Table 3.1: CAR Rules [2]

Rule	Bit 1 (LSB)	i^{th} Bit	Bit N (MSB)
150	$Q_1^+ = Q1 \oplus Q2$	$Q_i^+ = Qi \oplus Q_{i-1} \oplus Q_{i+1}$	$Q_N^+ = Q_N \oplus Q_{N-1}$
90	$Q_1^+ = Q2$	$Q_i^+ = Q_{i-1} \oplus Q_{i+1}$	$Q_N^+ = Q_{N-1}$

As can be seen in the table, rule 150 allows for three wires under test while rule 90 allows for two. With that in mind, the best way to maximize WUTs is to maximize the use of 150 rules in the construction of the CAR.

3.3.1 4-bit CAR with all 150 Rules

Consider, for example, a 4-bit CAR with all 150 rules (Figure 3.6). In theory this approach allows the testing of 24 wires in a single Virtex 4 PLB; the highest thus far. Due to the nature of CARs, this approach only produces two unique test patterns before repeating, depending on the initialization vector. It was also observed that several initialization vectors would lock up and never cycle. By having only two patterns, dual readback would be required to achieve high fault coverage. Even though the two patterns reflected different values, there is always one bit of the CAR that would never toggle. The location of this bit changes depending on the initialization vector.

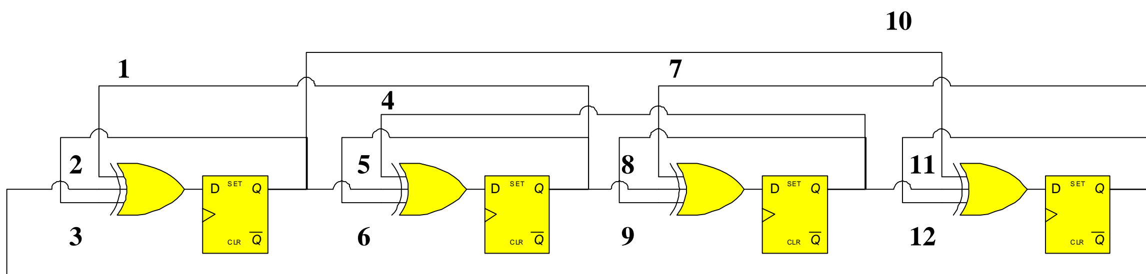


Figure 3.6: 4-bit CAR with all 150 Rules with Numbered WUTs

3.3.2 8-bit Cyclic Boundary CAR

To account for the lack of unique test patterns being generated by the 4-bit CAR approaches, an 8-bit CAR with cyclic boundary conditions was investigated (Figure 3.7). This approach will not ensure all possible test patterns, but enough to possibly ensure high fault coverage. As can be seen in the figure, 150 rules were used on the ends of the CAR to account for the cyclic boundary conditions while still giving a total of 21 WUTs.

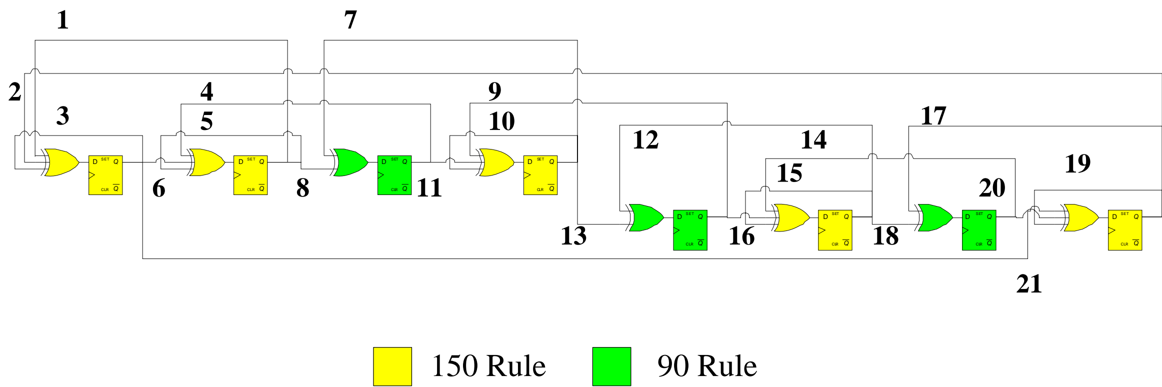


Figure 3.7: 8-bit Cyclic Boundary CAR with Numbered WUTs

3.3.3 8-bit Maximal Length Sequence CAR

The only way to achieve all possible test pattern combinations with a CAR is to assume null boundary conditions [2]. With that in mind, the 8-bit cyclic boundary CAR was modified into an 8-bit maximal length CAR with null boundary conditions (Figure 3.8). This modification gives a 2^N-1 test vector set to be applied to the WUTs. This modification also reduces the total number of WUTs from 21 to 19.

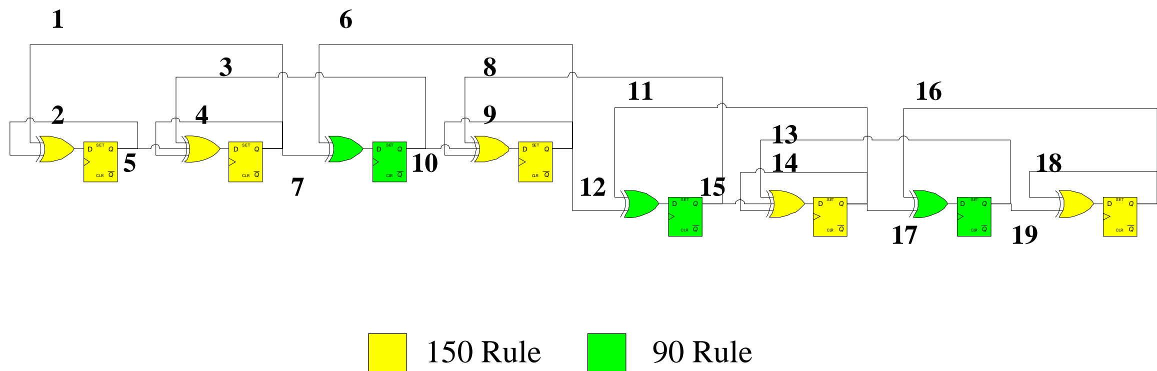


Figure 3.8: 8-bit Maximal Length Sequence CAR with Numbered WUTs

3.4 Routing BIST Approach Fault Simulations

Some of the routing BIST approaches mentioned in this thesis were modeled and simulated to analyze their gate-level stuck-at and bridging fault coverage. These simulations not only provide insight into the total fault coverage obtainable by the approach, but also what wires can be considered under test and whether or not the assumption of fault-free logic resources is valid.

The simulations were conducted by first writing a model of some of the routing BIST approaches in the Auburn Simulation Language (ASL). The ASL models were then used by two of the AUSIM commands, FLTGEN and BFTGEN, to generate the gate-level stuck-at and bridging fault lists respectively [23]. Once the fault lists were generated, the ASL models were simulated by the command, SIMUL8, to obtain fault-free circuit operation output results [23]. Using these output results and the fault lists, the FLTSIM and BFTSIM commands were used to perform the gate-level stuck-at and

bridging fault simulations respectively [23]. The fault coverage results were collected and are analyzed in the following sections. It should be noted that stuck-at refers to gate-level stuck-at fault coverage in all results presented. It should also be noted that the XOR gates used in the models were gate-level subcircuits consisting of an AND and two NOR gates.

3.4.1 Parity-Based Approaches

The fault simulation results for some of the parity-based routing BIST approaches are shown in Figure 3.9. If constructed to utilize an entire Virtex-4 PLB, then the dual parity [6] approach would be able to drive two ORAs, thereby doubling the number of WUTs to 12. Under the assumption that the feedback of the counters could also be considered WUTs, then the dual parity [6] approach would be able to test 18 wires in a single Virtex-4 PLB. However, the increase in overall fault coverage of the cross-coupled parity approach over the dual parity [6] approach is due to the fact that any fault within the counter TPG that can hinder the count, but still provide correct parity, will be detected by cross-coupling the parity bits. Even with the same number of WUTs, the dual parity [6] approach could never match the gate-level stuck-at fault coverage of the cross-coupled parity approach.

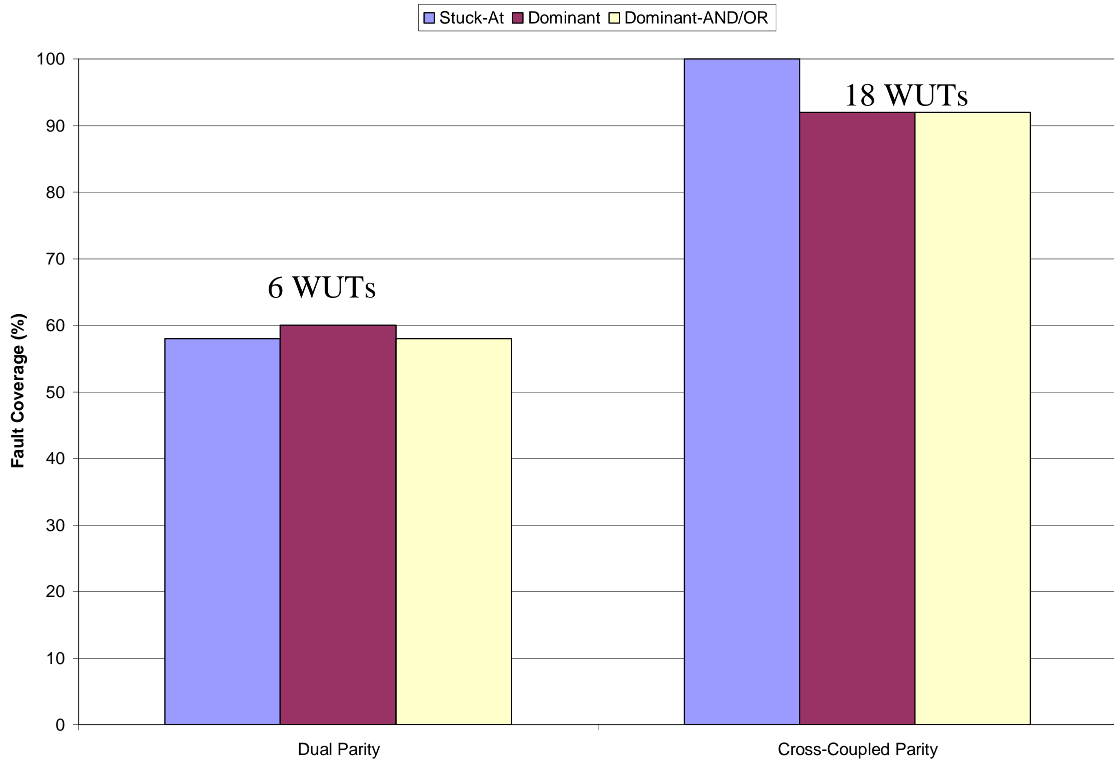


Figure 3.9: Parity-Based Approach Results

3.4.2 Counter-Based Approaches

The fault simulation results for counter-based approaches with comparison-based ORAs are shown in Figure 3.10. The results labeled stuck-at fault coverage with ORAs were those with all of the gate-level stuck-at faults generated from a model which included the comparison-based ORA. The stuck-at fault coverage without ORAs was obtained by removing all gate-level faults in the fault list that are associated with the comparison-based ORA. As can be seen in the figure, the single counter approach was unable to detect any stuck-at faults in the TPG. This was because all the faults detected by the single counter approach were within the ORA itself. Without the ORA faults, the

dual counter approach was able to achieve 100% stuck-at fault coverage. This observation is what first led to the investigation of TPG/ORA combination approaches.

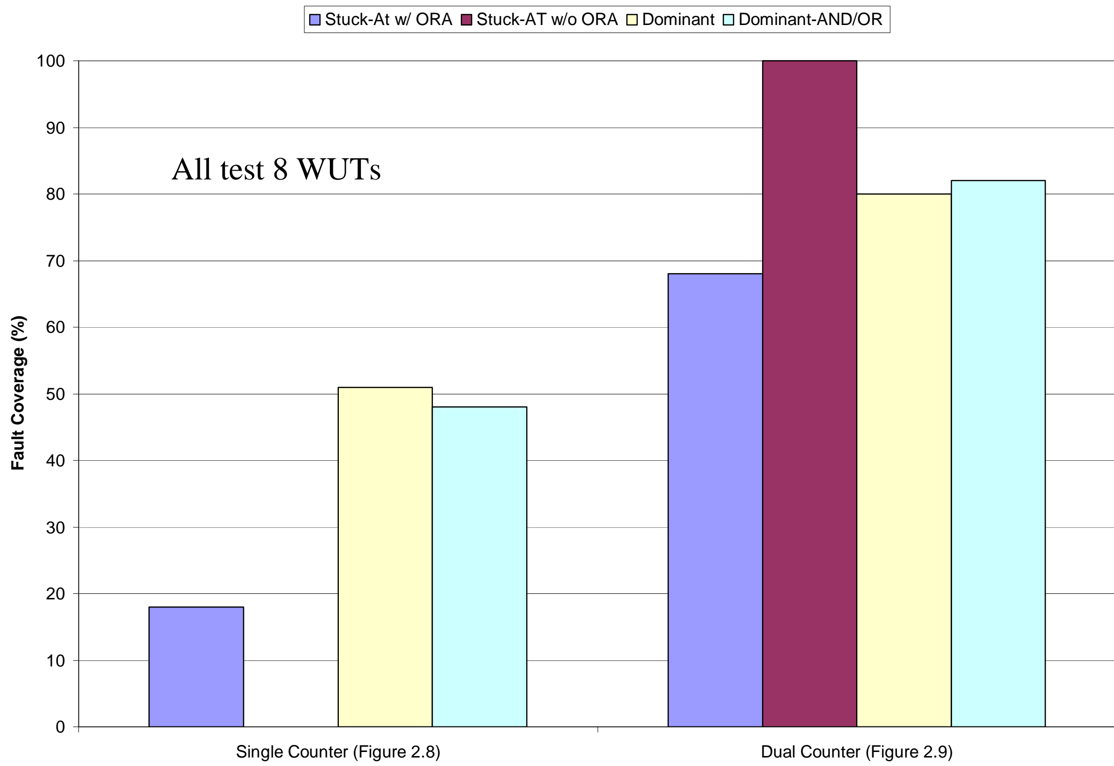


Figure 3.10: Counter-Based Approach Results

3.4.3 LFSR TPG/ORA Combination Approachs

The fault simulation results for LFSR TPG/ORA combination approaches with one configuration memory readback are given in Figure 3.11. All four approaches have adequate fault coverage, but are lacking in the number of total wires under test in a single PLB. As can be seen in the figure, the dominant bridging fault coverage for the 4-bit

external LFSR was lower due to the external XOR feedback gate. This caused the dominant faults for the LSB of the LFSR to only be potentially detected.

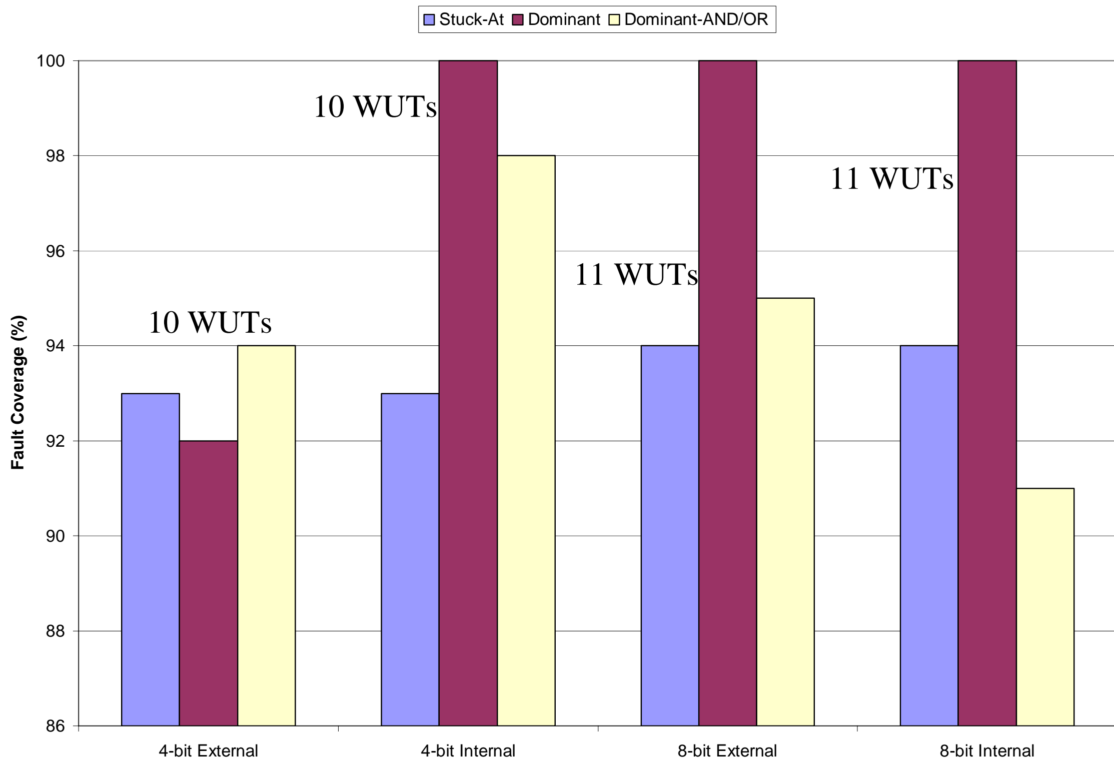


Figure 3.11: LFSR TPG/ORA Combination Approach Results

3.4.4 CAR Simulation Results

The fault simulation results for the CAR approaches can be seen in Figure 3.12. The greatest advantage of the CAR approaches is the total number of wires under test in a single PLB. The 8-bit cyclic CAR did not produce enough test patterns to allow for adequate fault coverage using a single BIST result readback. Reading the configuration

memory twice, however, was able to give this approach 100% gate-level stuck-at and slightly higher dominant and dominant-AND/OR fault coverage.

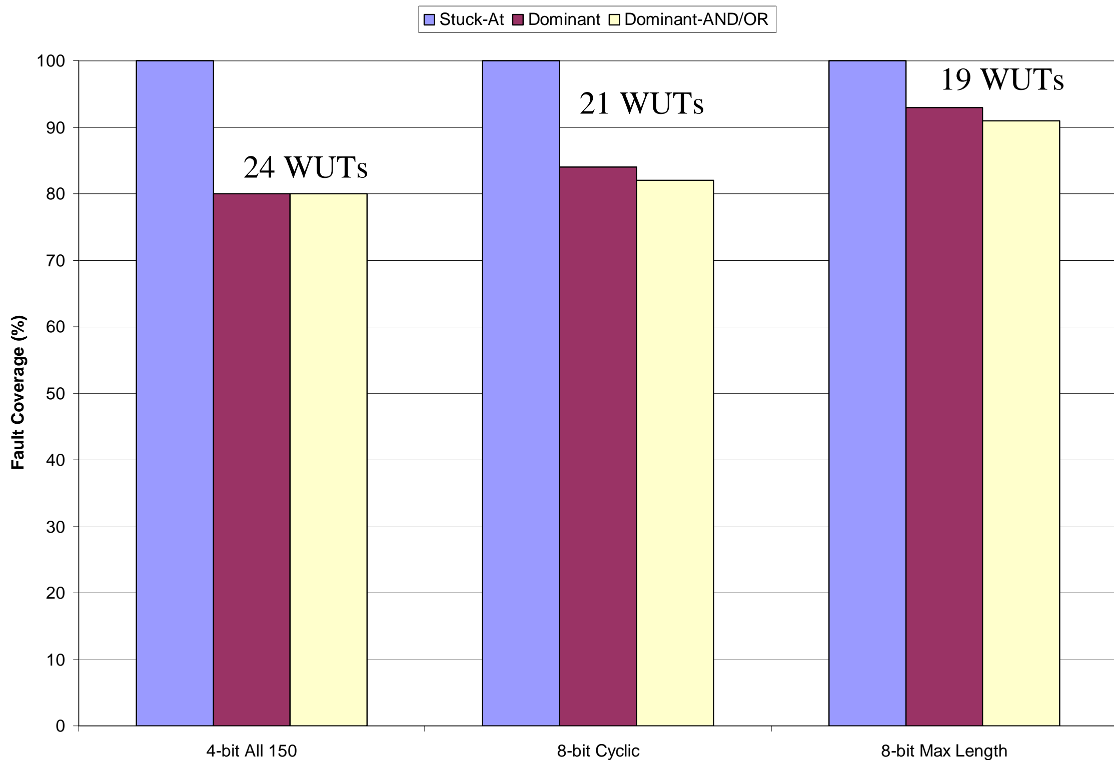


Figure 3.12: CAR Approach Results

Of all the CAR approaches, the 8-bit maximal length sequence approach had the best results. One thing to mention about this approach was its ability to achieve 100% fault coverage from a single configuration memory readback in only 18 clock cycles. It was also observed that the ending vector could be used as the initialization vector for a second phase of fault simulations for another 18 clock cycles, and still achieve the high fault coverage. The thought was that this ability could be used for quick dynamic partial reconfiguration of the Virtex-4 to minimize testing time.

3.5 Summary

Looking over the analysis of these approaches, it is obvious that every approach has its strengths and weaknesses. Consideration has to be given to total fault coverage, number of possible WUTs, and number of BIST results readbacks. The LFSR approaches had the least number of WUTs, but very high stuck-at and bridging fault coverage. However, the high fault coverage was only obtained by reading the configuration memory twice. The CAR approaches had the highest number of WUTs and fairly decent fault coverage. All of the CAR approaches had to read back the configuration memory twice, except the 8-bit maximal length CAR. Looking at the parity approaches, the cross-coupled parity approach dominated the dual parity approach on all respects. It had a high number of WUTs and excellent fault coverage while only having to retrieve the BIST results once. Of all the approaches both the cross-coupled parity and 8-bit maximal length CAR would be good approaches to implement for Virtex-4 routing BIST.

Chapter 4

Virtex-4 Routing BIST Implementation

The implementation of the cross-coupled parity BIST approach for programmable interconnect for the Xilinx Virtex-4 FPGA will be presented in this chapter. Specific focus will be placed on the testing of the global interconnect double lines for both PLB and non-PLB columns.

4.1 Virtex-4 Routing BIST Approach

The cross-coupled parity and 8-bit maximal length CAR were the two approaches that appeared to be the best candidates for Virtex-4 routing BIST implementation. Upon further investigation, it was observed that the limitations of the Virtex-4 architecture would not allow the 8-bit maximal length CAR to be constructed within a single PLB. The cause of this issue was mainly the amount of feedback connections the CAR required. Most flip-flop feedback has to enter the switch matrix through OMUXs. The Virtex-4 FPGA has a limited number of OMUXs per switch matrix, therefore the 8-bit CAR could not be constructed within a single PLB. To test the global routing resources, every bit of the CAR would have to be placed in a separate PLB. The complexity of

ensuring proper CAR construction and uniform replication throughout the entire array is too much. For those reasons, cross-coupled parity became the implementation approach for Virtex-4 routing BIST.

The approach itself was tailored to the Virtex-4 architecture (Figure 4.1). Two slices were populated with an even and odd parity generating TPG while the other two slices were populated with both an even and odd parity-based ORA in each slice. This adaptation allowed for full utilization of a single PLB, and produced a total of 6 test signals to be applied to the double lines. The six signals were fanned out to drive two ORAs putting 12 wires under test. By including the 6 feedback WUTs, the Virtex-4 routing BIST implementation tested a total of 18 wires.

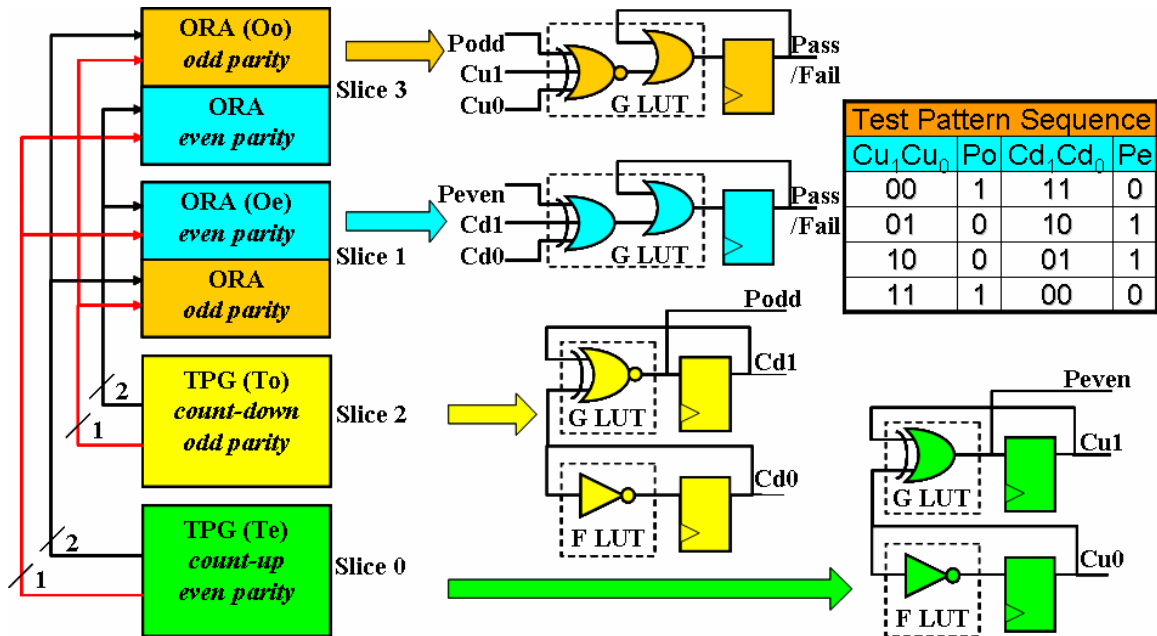


Figure 4.1: Cross-Coupled Parity Implementation in Virtex-4

To ensure opposite logic values on adjacent WUTs for bridging fault testing, the TPGs are alternated every other row or column depending on the direction of wires being tested. Implementing cross-coupled parity routing BIST for global routing double lines is presented in the following sections.

4.2 PLB Column Double Lines

There are 30 wire segments for each direction (north, south, east, and west) of the global routing double lines per switch matrix in the Virtex-4 FPGA. These segments are divided into three groups of terminals (BEG, MID, and END) for every switch matrix with 10 terminals per group. To ensure adequate fault coverage, all 10 terminals must be tested for each group of a switch matrix. The test configurations are separated into groups according to the directions they test.

The north double lines were the first direction configurations developed using the cross-coupled parity approach (Figure 4.2). This is accomplished by populating two slices of a PLB with the 2-bit up and down counters throughout the entire array. The remaining two slices of a PLB are then populated with even and odd parity-based ORAs throughout the entire array. The six test signals, four count bits and two parity bits, were routed onto six north double line segments via the BEG terminals. These signals are then routed through the north double lines at the MID terminals of the northerly adjacent switch matrix. Within the switch matrix, the test signals are routed to their respective parity-based ORA. This process is repeated at the END terminals of the double lines.

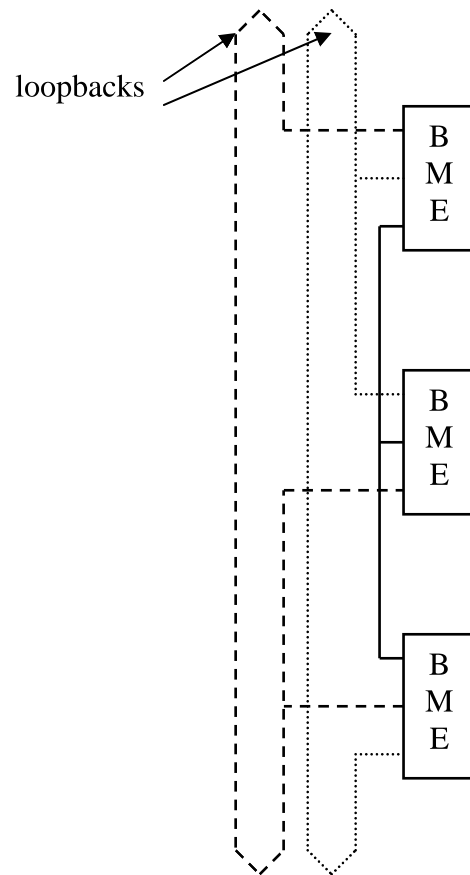


Figure 4.2: Example North Double Line Implementation

When the signals reach the edge of the array, edge loopback segments are utilized to route the signals onto their corresponding south direction line. They are routed back to the opposite edge of the array using south double lines. When an END terminal is reached, it is reconnected to the BEG terminal within the switch matrix to continue down the array. Edge loopback segments are also utilized at the southern edge of the array to route the test signals back onto their original northbound double lines. This allows the final MID and END ORA terminals to be routed, thereby completing the circular test architecture. One point to note is that there are only six test signals, yet 10 double lines

need to be tested. Due to the limited supply of output multiplexers within a switch matrix, the six test signals can not be fanned out to drive all 10 double lines at once. This therefore requires two test configurations be used to test all 10 lines (Table 4.1 and 4.2).

In the table the slice numbers are paired with their respective TPG or ORA (Te, To, Oe, and Oo). The first column shows which line number is being tested. The BEG column shows which test signal is connected to the BEG terminal via its respective slice output along with any required PIPs to complete the first connection in the third column. The MID and END columns show which ORA LUT input in the respective slice is connected to the MID and END terminals, along with any PIPs needed to complete the connection in the fifth and seventh columns. The feedback table shows the output feedback to LUT input paths the TPG and ORA require.

Table 4.1: North Double Line Configuration 1

Oe(0) Oo(1) Te(2) To(3)

Wire	BEG	PIP	MID	PIP	END	PIP
3	X3(CU0)		F3-1		G3-0	
4	X2(CD0)		G3-1		F3-0	
5	Y2(peven)		G2-1		F2-0	
6	YQ2(CU1)	OMUX9 G3-2	F4-1	Byp_Int_B1	G2-0	
7	YQ3(CD1)	OMUX11 W2BEG6 G3-3	G1-1		F1-0	
8	Y3(podd)		F1-1		G1-0	

Feedback (TPG)
XQ2 >> OMUX6 >> F2-2
XQ2 >> OMUX6 >> Bounce3 >> G4-2
XQ3 >> OMUX13 >> F4-3
XQ3 >> OMUX13 >> Byp_Int _B7 >> G2-3
Feedback (ORA)
XQ0 >> OMUX2 >> F4-0
YQ0 >> OMUX4 >> W2BEG2 >> Byp_Int_B4 >> G4-0
XQ1 >> OMUX15 >> E2BEG9 >> Byp_Int_B5 >> F2-1
YQ1 >> OMUX5 >> Bounce0 >> G4-1

Table 4.2: North Double Line Configuration 2

Oe(0) Oo(1) Te(2) To(3)

Wire	BEG	PIP	MID	PIP	END	PIP
0	Y2(peven)	OMUX0	F3-1	Byp_Int_B2	G4-0	
1	X2(CD0)		F4-1		G2-0	Byp_Int_B0
2	YQ2(CU1)	OMUX4 W2BEG2 G2-2	G3-1		F3-0	
3	X3(CU0)		G4-1	Byp_Int_B4	F2-0	E2BEG4 Bounce2
8	Y3(podd)		G1-1		F1-0	
9	YQ3(CD1)	OMUX15 E2BEG9 G4-3	F1-1		G1-0	

Feedback (TPG)
XQ2 >> OMUX6 >> F2-2
XQ2 >> OMUX6 >> Bounce3 >> G4-2
XQ3 >> OMUX13 >> F4-3
XQ3 >> OMUX13 >> Byp_Int _B7 >> G2-3
Feedback (ORA)
XQ0 >> OMUX2 >> F4-0
YQ0 >> OMUX5 >> Bounce1 >> G3-0
XQ1 >> OMUX11 >> W2BEG6 >> F2-1
YQ1 >> OMUX9 >> G2-1

The south double lines are similar to the north double lines. They too require two test configurations to test all 10 lines (Table 4.3 and 4.4). Like the north double lines, the test signals are first routed onto the wire segments via the BEG terminals to be sourced to the parity-based ORAs via the MID and END terminals along the wire segments. At the southern edge of the array the test signals are routed onto the corresponding north double lines using the edge loopback segments. When the signals have traveled back to the northern edge of the array they are routed back onto the original south double lines to complete the circular MID and END terminal connections.

Table 4.3: South Double Line Configuration 1

Oe(0) Oo(1) Te(2) To(3)

Wire	BEG	PIP	MID	PIP	END	PIP
2	YQ2(CU1)	OMUX4 G1-2	F4-1		G4-0	
3	XQ2(CU0)	OMUX6 F2-2 G2-2	F3-1		G3-0	
4	X2(CD0)		G3-1		F3-0	
5	Y2(peven)		G2-1		F4-0	E2BEG3 Byp_Int_B4
6	Y3(podd)		F2-1		G2-0	
7	YQ3(CD1)	OMUX11 W2BEG6 G3-3	G1-1	Byp_Int_B3	F2-0	

Feedback (TPG)
XQ3(CD0) >> OMUX9 >> Byp_Int_Bounce1 >> F1-3 >> G1-3
Feedback (ORA)
XQ1 >> OMUX13 >> F1-1
YQ1 >> OMUX2 >> G4-1
XQ0 >> OMUX5 >> Bounce1 >> Byp_Int_Bounce6 >> F1-0
YQ0 >> OMUX15 >> E2BEG9 >> G1-0

Table 4.4: South Double Line Configuration 2

Oe(0) Oo(1) Te(2) To(3)

Wire	BEG	PIP	MID	PIP	END	PIP
0	XQ2(CU0)	OMUX0 G1-2 F1-2	G4-1		F4-0	
1	XQ3(CD0)	OMUX2 F1-3 G1-3	F4-1		G4-0	
2	YQ2(CU1)	OMUX4 S2BEG2 Byp_Int_B0 G3-2	G3-1	W2BEG2	F3-0	Byp_Int_B2
7	Y2(peven)		F2-1		G2-0	
8	Y3(podd)		G1-1		F1-0	
9	YQ3(CD1)	OMUX15 E2BEG9 G4-3	F1-1		G1-0	

Feedback (ORA)
XQ1 >> OMUX6 >> F3-1
YQ1 >> OMUX9 >> G2-1
XQ0 >> OMUX11 >> W2BEG6 >> F2-0
YQ0 >> OMUX5 >> Bounce1 >> G3-0

In testing the east double lines, the non-PLB columns must be taken into account (Figure 4.3). This is because the east double lines are row based instead of column based. Like the north and south double line configurations, the east double line test signals are routed onto the wire segments via the BEG terminals to source parity-based ORAs via the MID and END terminals along the wire segments. At the eastern edge of the array, the signals use loopback segments to travel back across the array on the west double lines. When they reach the western edge of the array they route back onto their respective east double lines and finish the MID and END terminal connections, thereby completing the circle. The main difference is dealing with the non-PLB columns. The

non-PLB column switch matrix MID terminal connections are not used since there are no PLB slices to connect them to. The END terminals in non-PLB columns, however, are reconnected to the BEG terminals so that the next adjacent PLBs can be sourced with test patterns. By doing this, the non-PLB columns are in essence emulating the behavior of a TPG thereby providing the proper parity and count signals required by the adjacent PLBs. Again, two configurations were required to test all 10 lines (Table 4.5 and 4.6).

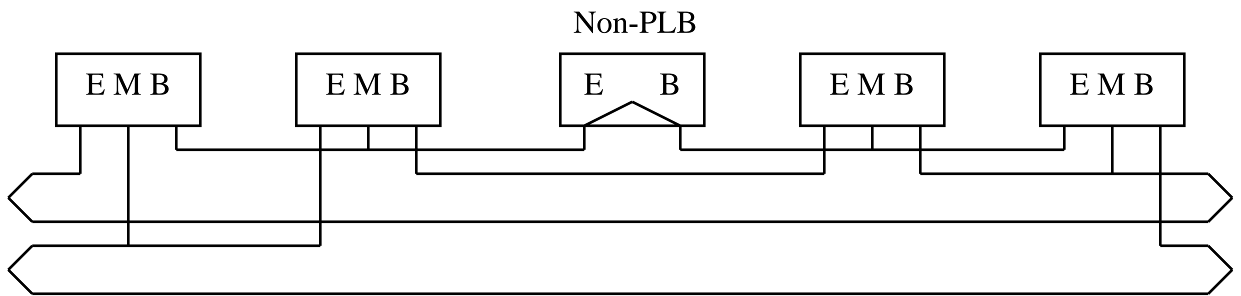


Figure 4.3: Example East Double Line Implementation

Table 4.5: East Double Line Configuration 1

Oe(0) To(1) Te(2) Oo(3)

Wire	BEG	PIP	MID	PIP	END	PIP
1	Y1(podd)		F2-3	Byp_Int_Bounce2	G1-0	Byp_Int_Bounce0 Byp_Int_Bounce3
2	YQ2(CU1)	OMUX6 G2-2	F1-3		G4-0	
3	X2(CD0)		G2-3		F4-0	Byp_Int_Bounce4
4	Y2(peven)		G4-3	Bounce3	F3-0	
5	X1(CU0)		F3-3		G2-0	
6	YQ1(CD1)	OMUX11 N2BEG7 G1-1	G3-3		F2-0	

Feedback (TPG)
XQ1(CD0) >> OMUX9 >> F2-1
XQ1(CD0) >> OMUX9 >> G2-1
XQ2(CU0) >> OMUX2 >> F1-2
XQ2(CU0) >> OMUX2 >> G1-2
Feedback (ORA)
XQ3 >> OMUX13 >> F4-3
YQ3 >> OMUX0 >> S2BEG0 >> G1-3
XQ0 >> OMUX15 >> N2BEG9 >> F1-0
YQ0 >> OMUX5 >> Bounce1 >> G3-0

Table 4.6: East Double Line Configuration 2

Te(0) To(1) Oe(2) Oo(3)

Wire	BEG	PIP	MID	PIP	END	PIP
0	Y0(peven)		F1-3		G1-2	
1	Y1(podd)		G1-3		F1-2	
6	YQ0(CU1)	OMUX9 G2-0	G3-3		F4-2	Byp_Int_Bounce3
7	X1(CU0)		G4-3	N2BEG7	F3-2	
8	X0(CD0)		F3-3	Byp_Int_Bounce5	G4-2	
9	YQ1(CD1)	OMUX15 E2BEG9 G1-1	F4-3		G2-2	Byp_Int_Bounce7

Feedback (TPG)
XQ0(CU0) >> OMUX13 >> F1-0
XQ0(CU0) >> OMUX13 >> G1-0
XQ1(CD1) >> OMUX5 >> Bounce0 >> F4-1
XQ1(CD1) >> OMUX5 >> Bounce0 >> G4-1
Feedback (ORA)
XQ3 >> OMUX6 >> F2-3
YQ3 >> OMUX4 >> W2BEG2 >> G2-3
XQ2 >> OMUX2 >> Byp_Int_Bounce2 >> F2-2
YQ2 >> OMUX0 >> S2BEG0 >> Byp_Int_Bounce0 >> G3-2

The west double lines are similar to the east double lines, where careful attention is paid to the non-PLB columns. The count and parity test signals were routed onto wire segments via BEG terminals and monitored at MID and END terminals by their corresponding parity-based ORAs. At the western edge of the array, the test signals are routed onto their respective east double lines using loopback segments. As they travel back to the eastern edge, END to BEG connections are made where needed to continue the route. At the eastern edge of the array, they are then routed back onto their corresponding west double lines to complete the final MID and END ORA connections. When considering the non-PLB columns, the END terminals are connected to the BEG terminals so that the two westerly adjacent PLB columns are sourced with the proper test

signals. Again, the non-PLB columns MID terminals are not used. The west double lines also require two test configurations to test all 10 lines (Table 4.7 and 4.8).

Table 4.7: West Double Line Configuration 1

Oe(0) Te(1) To(2) Oo(3)

Wire	BEG	PIP	MID	PIP	END	PIP
3	X2(CU0)		G2-3		F3-0	
4	Y1(peven)		F2-3		G3-0	
5	X1(CD0)		F3-3		G2-0	
6	YQ1(CU1)	OMUX11 W2BEG6 G2-1	G3-3		F2-0	
7	Y2(podd)		G4-3		F1-0	
8	YQ2(CD1)	OMUX13 G4-2	F4-3		G1-0	

Feedback (TPG)
XQ1 >> OMUX2 >> F4-1
XQ1 >> OMUX2 >> G4-1
XQ2 >> OMUX9 >> F3-2
XQ2 >> OMUX9 >> G3-2
Feedback (ORA)
XQ0 >> OMUX0 >> S2BEG0 >> F4-0
YQ0 >> OMUX4 >> S2BEG2 >> G4-0
XQ3 >> OMUX5 >> Bounce0 >> F1-3
YQ3 >> OMUX6 >> Byp_Int_B4 >> G1-3

Table 4.8: West Double Line Configuration 2

Te(0) Oe(1) To(2) Oo(3)

Wire	BEG	PIP	MID	PIP	END	PIP
0	Y0(peven)		F1-3		G4-1	
1	YQ0(CU1)	OMUX2 G4-0	G3-3	Byp_Int_B0	F4-1	
2	XQ2(CD0)	OMUX6 F2-2 G2-2	F2-3		G2-1	N2BEG4 Bounce2
3	X2(CU0)		G2-3		F3-1	
8	YQ2(CD1)	OMUX13 G4-2	F3-3	Byp_Int_B5	G1-1	
9	Y2(podd)	OMUX15	G4-3		F1-1	

Feedback (TPG)
XQ0 >> OMUX9 >> F2-0
XQ0 >> OMUX9 >> G2-0
Feedback (ORA)
XQ1 >> OMUX11 >> W2BEG6 >> F2-1
YQ1 >> OMUX5 >> Bounce1 >> G3-1
XQ3 >> OMUX0 >> S2BEG0 >> Byp_Int_B2 >> Byp_Int_B6 >> F4-3
YQ3 >> OMUX4 >> S2BEG2 >> G1-3

4.3 Non-PLB Column Double Lines

Testing the non-PLB column double lines is more involved than the PLB column double lines. This is due to having to place the TPGs and ORAs in surrounding PLB columns, and routing the test signals to the desired lines to be tested. More routing resources are required to realize the BIST architecture for these wires. Help is given in that the entire array does not have to be populated with the design. Otherwise, routing congestion and conflicts would surely occur.

One advantage the non-PLB column double line configurations have over the PLB column double line configurations is that two sets of directional wires are tested in one configuration instead of only one. Since the non-PLB column BEG and END terminals were used in testing east and west double lines, they are already considered tested. By using the non-PLB column east and west MID terminals as gateways into the non-PLB column switch matrices to source the north and south double lines, they too can be considered tested, thereby completing the east and west wire testing of non-PLB columns.

In theory, this advantage should reduce the number of test configurations for non-PLB column double lines by eliminating the configurations needed for two directions. This was not the case, however, due to the connections of wire segments between different directions. When leaving the non-PLB column switch matrix to connect with the ORA, the test signal must be routed onto its respective east or west wire segment via the BEG terminal. The north and south MID terminal connections have no problem with this routing. However, the END terminal connections shift and connect to the next wire in the group (eg. N2END0 connects to E2BEG1) (Figure 4.4). This shift causes routing conflicts in the adjacent rows. For that reason, MID and END terminal testing must be separated into their own configurations thus requiring four configurations for each direction; two for MID and two for END terminal testing.

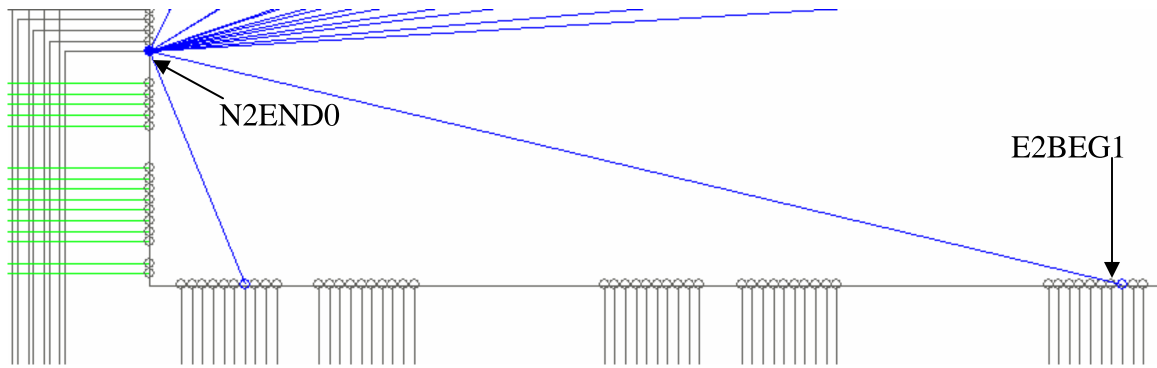


Figure 4.4: END Terminal Connection Shift

Another disadvantage of non-PLB column double line testing is that not all segments have the ability to connect with east and west lines to leave the non-PLB column and connect to their ORA. This problem occurs on the N2END9, S2END0, and S2END1 terminals (Figure 4.5). The only feasible way to test these segments is to develop an additional configuration. The signals are routed onto the desired wire segments and routed to the edges of the array. At the edges, they use the loopback segments to route onto the corresponding opposite direction double lines. The signals are then routed to their respective ORA.

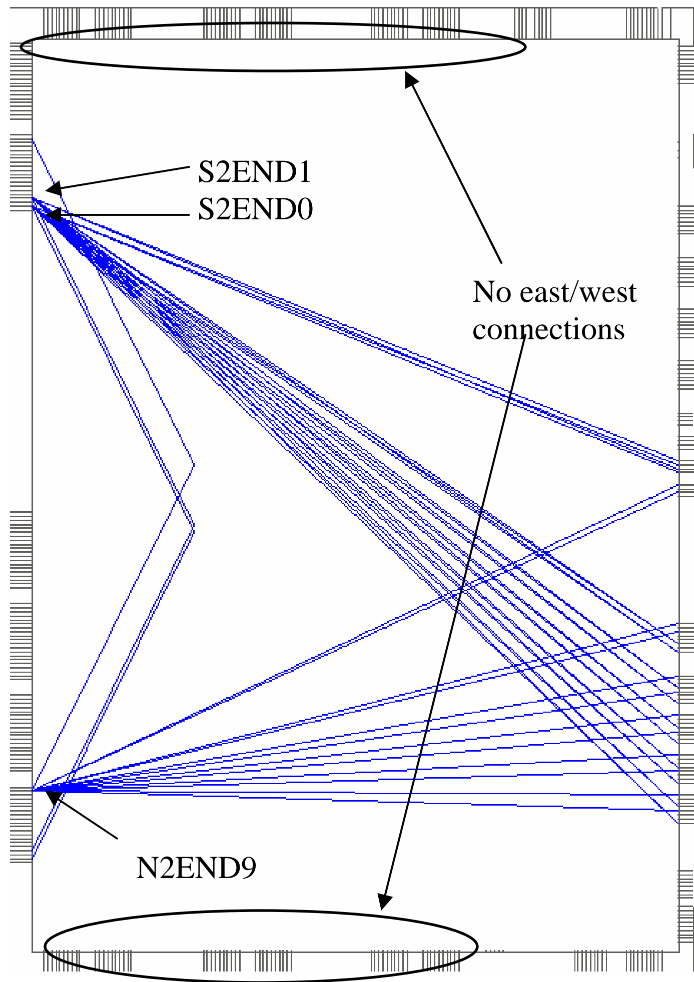


Figure 4.5: Non-Connecting Terminals

The north non-PLB double lines BIST configurations also test the remaining untested east non-PLB double lines MID terminals. TPGs are placed in the left adjacent column to the non-PLB column while ORAs are placed in the right adjacent column (Figure 4.6).

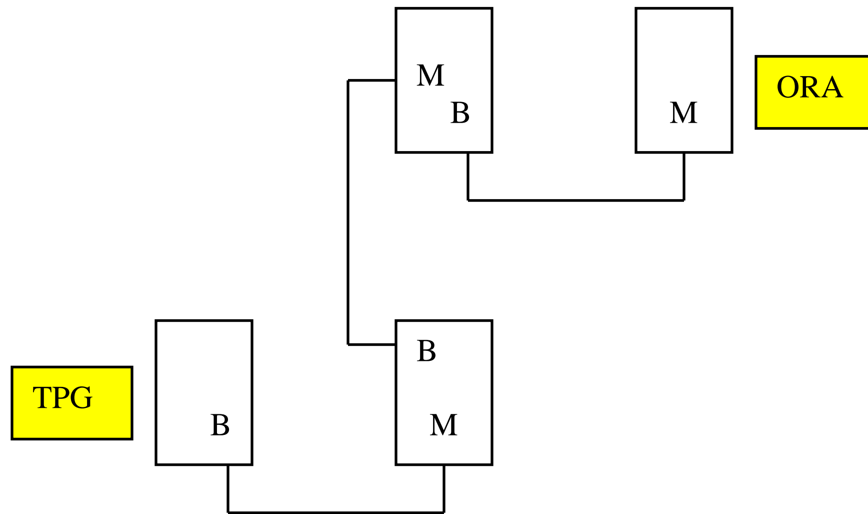


Figure 4.6: Example North Non-PLB Double Line Implementation

This process is repeated throughout the entire array except for the eastern and western edges. On the edges, the TPGs and ORAs are placed in the same column. The test signals are routed onto the east wire segments via the BEG terminals. At the non-PLB column switch matrix, they are routed from the east MID terminals and onto the north wire segments via the BEG terminals. Depending on the configuration, the test signals are rerouted back onto the east wire segments via the BEG terminals from either the north MID or north END terminals. They are then routed to the adjacent PLB column and connect to the parity-based ORA via the MID terminals. On the east edge of the array, instead of rerouting back onto east wire segments to be connected to the ORA, the signals are routed onto west wire segments via the BEG terminals from the north MID and END terminals. On the west edge of the array, the test signals are first routed on the west wire segments via BEG terminals. The loopback segments are then used to route the signals onto their corresponding east lines where they were connected to the non-PLB column

switch matrix from the east MID terminal (Figure 4.7). They then follow the same pattern as the other routes running north then east.

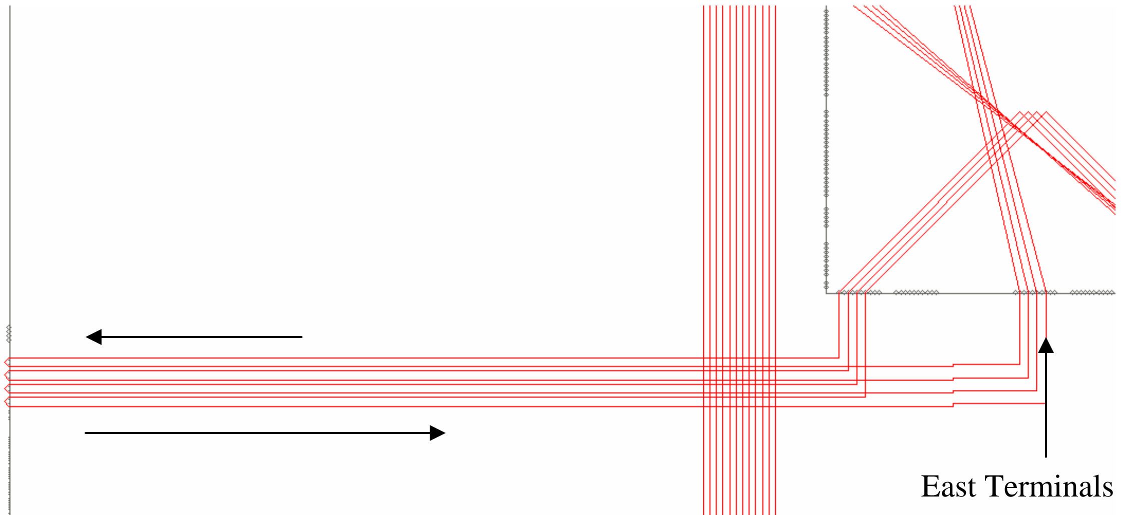


Figure 4.7: Loopbacks Used to Ensure East MID Testing

The south non-PLB column double lines are similar to the north configurations, and also test a second direction; the west MID terminals (Figure 4.8). The TPGs are placed in the right adjacent column to the non-PLB column while ORAs are placed in the left adjacent column. The process is repeated throughout the entire array except for the eastern and western edges. Like the north configurations, the edge TPGs and ORAs are placed in the same column. The test signals are routed onto the west wire segments via BEG terminals to connect to the non-PLB column switch matrix. There they are rerouted onto south wire segments. Depending on the configuration, the test signals are rerouted back onto the west wire segments of the non-PLB column from either the south MID or south END terminals. They are then connected to their respective ORA by the west MID terminals of the adjacent PLB column.

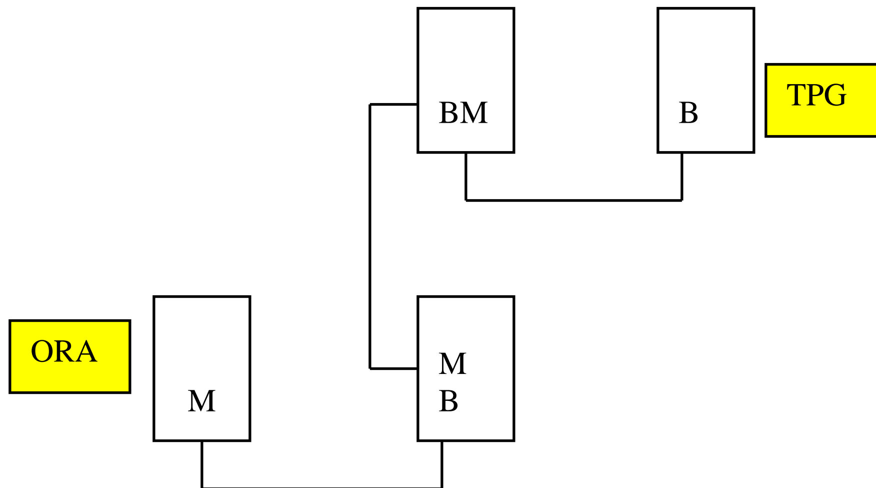


Figure 4.8: Example South Non-PLB Double Line Implementation

On the eastern edge of the array, the signals first have to be routed on the east wire segments to the loopback connections at the edge. There they connect to their corresponding west lines where they are then connected to the non-PLB column switch matrix by the west wire segments via MID terminals (Figure 4.9). From there they follow the same path as the other routes running south then west. On the western edge of the array, the signals follow the same paths as the other routes except where they are rerouted to the ORAs. At that point they are routed onto east wire segments to connect to their respective ORA by east MID terminals. Like the north and east configurations, the south and west configurations requires four configurations to test all 10 lines. This brings the total configuration count to nine for non-PLB column wires.

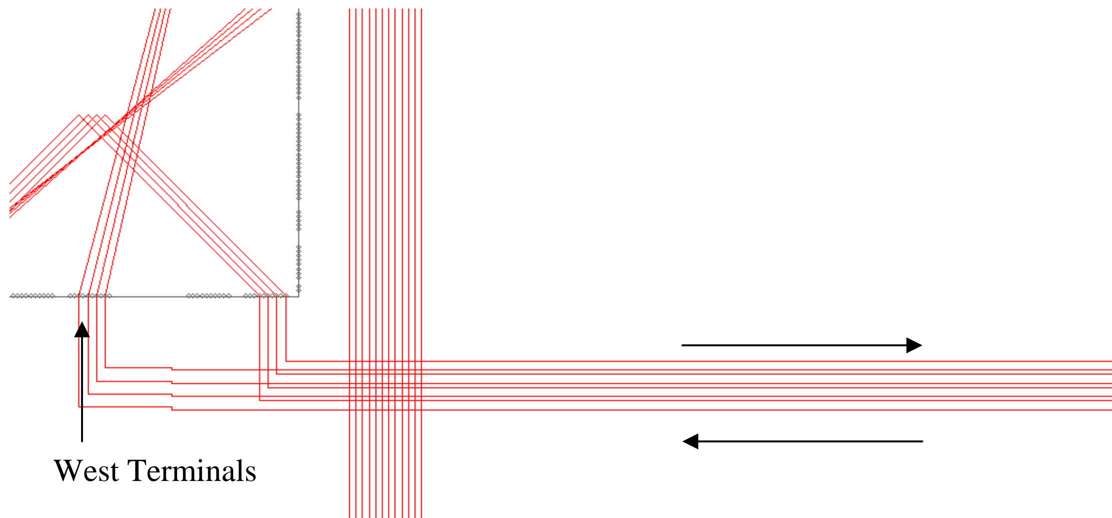


Figure 4.9: Loopbacks Used to Ensure West MID Testing

4.4 Virtex-4 FX Device Configurations

The Virtex-4 FX devices contain one or two Power PC modules in the left half of the FPGA. In developing BIST configurations for the double line segments careful attention must be paid to the routing around these modules. Like the edges of the array, the edges of the Power PCs also have loopback connections that can be used by the double lines to change direction of the routing path. Every Power PC row and column has a set of these loopback connections. Since the configurations for the FX device double lines are relatively the same as the other Virtex-4 devices, a post process algorithm was developed to modify a dummy configuration with the needed changes to be compatible with the FX devices (Figure 4.10). The algorithm takes a completely routed array with no consideration for the Power PC locations and strips out all of the routing where the Power PCs are located. It then makes the appropriate connections and

modifications to complete the routing BIST architecture in the loopback connections around the Power PC.

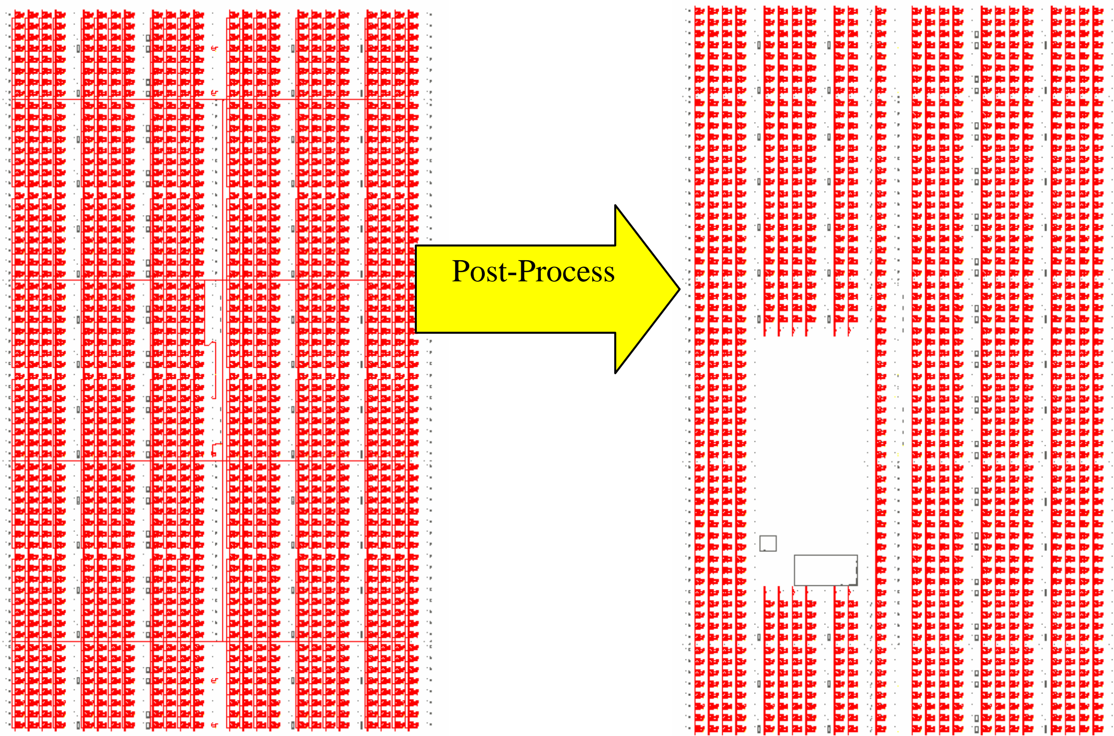


Figure 4.10: FX Configuration Generation Process

A few special considerations must be made for the FX device configurations based on the direction being tested. For the north and south PLB column double lines, the entire column on the right edge of the Power PCs must be treated like a non-PLB column. This is because that particular column contains both PLB and non-PLB sections (the non-PLB sections being alongside the Power PCs). The column will be tested in the non-PLB column tests. For the non-PLB column configurations, the top and bottom rows in the Power PC area must be included. Aside from the small modifications to the

existing configurations by the post process algorithm, no additional configurations must be developed specifically for FX devices.

4.5 Virtex-4 Routing BIST Results

The implementation of the cross-coupled parity routing BIST approach in the Virtex-4 FPGA was presented in this chapter (Table 4.9). The adaptation of the approach to the Virtex-4 architecture was also shown. Specific focus on the development of the PLB and non-PLB column double lines BIST configurations was also presented. It was shown how the consideration for the Power PCs in the FX devices was handled.

Table 4.9: Summary of Virtex-4 Routing BIST Global Double Line Configs

Virtex-4 Routing BIST Configurations	
Direction	Number Of Configs
North	2
South	2
East	2
West	2
North/East Non-PLB	4
South/West Non-PLB	4
Additional Non-PLB	1
	17 Total Configs

CHAPTER 5

SUMMARY AND CONCLUSIONS

5.1 Summary of Routing BIST Approaches

The work in this thesis presented an analysis and evaluation of previous routing BIST approaches. New approaches were proposed in an effort to eliminate assumptions that were made with the previous approaches to obtain high fault coverage. Some of the approaches were modeled and simulated to obtain their gate-level stuck-at as well as dominant and dominant-AND/OR bridging fault coverage. Emphasis was not only on fault coverage, but also on the number of wires that could be tested in a given configuration. Counter-based approaches that used a comparison-based ORA had a decent number of WUTs, but suffered in overall fault coverage. A number of different LFSR TPG/ORA combination approaches were simulated. Even though they were able to obtain high fault coverage, the number of WUTs was low and the configuration memory had to be read twice. Several CAR approaches were also simulated as TPG/ORA combinations. The CARs had many more feedback paths, which increased the total number of WUTs. However, a few of the CARs could not produce the amount of test patterns needed to achieve good fault coverage. The 8-bit maximal length

sequence CAR was found to be one of the best approaches, as it was able to achieve very high fault coverage while only needing to read the configuration memory once. It also had one of the highest numbers of WUTs. Its drawback, however, was that its construction could not be contained in a single Virtex-4 PLB due to the limitations in the switch matrices by the OMUXs. In the end, the cross-coupled parity approach was the one chosen for Virtex-4 implementation.

5.2 Summary of Virtex-4 Routing BIST

The work in this thesis developed and implemented a routing BIST architecture for the Virtex-4 FPGA using the cross-coupled parity approach. Specific focus is given to the global double line routing resources of PLB and non-PLB columns. The double lines are characterized by the direction they source: north, south, east, or west. These lines span three rows or columns, depending on direction, in the array and have connection terminals at every switch matrix in their span. Every switch matrix has 30 double line wire segments per direction. The 30 segments are organized into three groups of terminals: BEG, MID, and END. Each group contains 10 wire segments to and from the switch matrix. To ensure adequate testing, all 10 terminals must be tested for each group of terminals and wire segments associated with a switch matrix.

The first routing BIST configurations developed for Virtex-4 were for the columns containing PLBs. The limitation of the Virtex-4 architecture in regards to OMUXs in the switch matrix required that only 6 global lines be tested per BIST configuration. Therefore, each direction required two BIST configurations to fully test

all 10 wire segments and terminals for each group. One advantage of these BIST configurations was that the east and west BEG and END terminals for non-PLB columns were also tested during the east and west direction configurations.

The non-PLB column BIST configurations were more involved than those of the columns containing PLBs. These configurations were able to test two directions at the same time: north/east and south/west. The ability to test two directions simultaneously was thought to be an advantage that would reduce the number of configurations to test the FPGA. However, it was observed that the terminals of the non-PLB columns had to be tested independently due to routing conflicts. Therefore, the non-PLB columns required four BIST configurations for both directions to fully test all of the lines.

Three of the lines that needed to be tested in the non-PLB columns were not able to change directions within a switch matrix. To handle this issue, an additional BIST configuration had to be developed to route the test signals on these wires to the edges of the array and use the loopback connections to route them to their respective ORAs.

One challenge faced in developing the routing BIST configurations for the global double lines was accommodating the area taken up by the Power PCs in the FX devices. To deal with that issue, a post process algorithm was developed that stripped the routing out of the area where a Power PC would be located and reroute any signals that needed to complete the configuration.

5.3 Future Work

There is a lot of work left to do in the area of routing BIST for Virtex-4. Configurations must be developed to test hex and long lines for both PLB and non-PLB columns. Lines that branch from END terminals on global double lines need to be verified and tested. The lines connecting the switch matrices around the Power PCs in FX devices to the Power PCs themselves need tests to be developed as well.

The work presented in this thesis could easily be applied to next generation FPGAs. The cross-coupled parity approach can be modified to utilize the full potential of any architecture. For example, since most newer FPGAs have routing resources based on buffered multiplexer PIPs to prevent signal degradation, the cross-coupled parity test signals could be fanned out to source as many wires as the architecture would allow.

BIBLIOGRAPHY

[1] C. Stroud, S. Wijesuriya, C. Hamilton, and M. Abramovici, "Built-In Self-Test of FPGA Interconnect," *Proc. IEEE International Test Conf.*, pp. 404-411, 1998.

[2] C. Stroud, A Designer's Guide to Built-In Self-Test, Boston: Springer, 2002.

[3] J. Smith, T. Xia, C. Stroud, "An Automated BIST Architecture for Testing and Diagnosing FPGA Interconnect Faults," *J. Electronic Testing: Theory & Applications*, vol. 22, no. 4, pp. 239-253, 2006.

[4] M. Abramovici and C. Stroud, "BIST-Based Test and Diagnosis of FPGA Logic Blocks," *IEEE Trans. on VLSI Systems*, vol. 9, no. 1, pp. 159-172, 2001.

[5] C. Stroud, J. Nall, M. Lashinsky, and M. Abramovici, "BIST-Based Diagnosis of FPGA Interconnect," *Proc. IEEE International Test Conf.*, pp. 618-627, 2002.

[6] J. Sunwoo and C. Stroud, "Built-In Self-Test of Configurable Cores in SoCs Using Embedded Processor Dynamic Reconfiguration," *Proc. International System-on-Chip Design Conf.*, pp. 174-177, 2005.

[7] C. Stroud, J. Bailey, J. Emmert, D. Nickolic, and K. Chhor, "Bridging Fault Extraction from Physical Design Data for Manufacturing Test Development," *Proc. IEEE International Test Conf.*, pp. 760-769, 2000.

[8] C. Stroud, K. Leach, and T. Slaughter, "BIST for Xilinx 4000 and Spartan Series FPGAs: A Case Study," *Proc. IEEE International Test Conf.*, pp. 1258-1267, 2003.

[9] S. Dhingra, S. Garimella, A. Newalkar, and C. Stroud, "Built-In Self-Test for Virtex and Spartan II FPGAs Using Partial Reconfiguration," *Proc. IEEE North Atlantic Test Workshop*, pp. 7-14, 2005.

[10] I. Harris and R. Tessier, "Testing and Diagnosis of Interconnect Faults in Cluster-Based FPGA Architectures," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 21, no. 11, pp. 1337-1343, 2002.

[11] X. Sun, J. Xu, B. Chan, and P. Trouborst, "Novel Technique for Built-In Self-Test of FPGA Interconnects," *Proc. IEEE International Test Conf.*, pp. 795-803, 2000.

[12] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "Architecture of Field-Programmable Gate Arrays," *Proceedings of the IEEE*, vol. 81, no. 7, pp. 1013-1029, 1993.

[13] G. Moore, "Cramming More Components onto Integrated Circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82-85, 1998.

[14] B. Dixon and C. Stroud, "Analysis and Evaluation of Routing BIST Approaches for FPGAs," *Proc. IEEE North Atlantic Test Workshop*, pp. 85-91, 2007.

[15] L. Wang, C. Stroud, and N. Touba, System On Chip Test Architectures: Nanometer Design for Testability, Amsterdam: Elsevier, 2007.

[16] "Xilinx Delivers Virtex-4 FPGAs", Xilinx Press Release #0480, www.xilinx.com.

[17] "Virtex-4 User Guide", User Guide UG070, Xilinx, Inc., 2005.

[18] S. Dhingra, D. Milton, and C. Stroud, "BIST for Logic and Memory Resources in Virtex-4 FPGAs," *Proc. IEEE North Atlantic Test Workshop*, pp. 19-27, 2006.

[19] "Xilinx Corp FPGA Editor Software Manual", Software Manual 9.2i, Xilinx, Inc., 2005.

[20] "Virtex 4 Family Overview", Datasheet DS112, Xilinx, Inc., 2007.

[21] I. Kuon and J. Rose, "Measuring the Gap between FPGAs and ASICs," *ACM International Symp. on FPGAs*, pp. 21-30, 2006.

[22] V. Suthar and S. Dutt, "Mixed PLB and Interconnect BIST for FPGAs Without Fault-Free Assumptions," *Proc. IEEE VLSI Test Symp.*, pp. 36-43, 2006.

[23] C. Stroud, "AUSIM : Auburn University SIMulator – Version L2.2," 2004.