BUILT-IN SELF-TEST FOR INPUT/OUTPUT TILES 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.

Lee W. Lerner

Certificate of Approval:

Charles E. Stroud, Chair
Professor
Electrical and Computer Engineering

Vishwani D. Agrawal
Professor
Electrical and Computer Engineering

Victor P. Nelson
Professor
Electrical and Computer Engineering

Joe F. Pittman
Interim Dean
Graduate School

BUILT-IN SELF-TEST FOR INPUT/OUTPUT TILES IN FIELD PROGRAMMABLE GATE ARRAYS

Lee W. Lerner

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
May 10, 2008

BUILT-IN SELF-TEST FOR INPUT/OUTPUT TILES IN FIELD PROGRAMMABLE GATE ARRAYS

Lee W. Lerner

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 the Author

____

Date of Graduation

VITA

Lee W. Lerner, son of K. Lee and Brenda Lerner, was born in Dallas, Texas on October 13, 1982. Lee is an IEEE published author and has also contributed to several volumes on science, computer science, and technology for students at the high school and early college level. In fall of 2003, he accepted a position at the NASA Kennedy Space Center where he worked as a student co-op / engineer trainee in the Information Technology and Communications Directorate. In spring 2006, he graduated from Auburn University with a Bachelor of Electrical Engineering with a major focus in Computer Engineering. Upon graduating, he began working on his Master of Science degree at Auburn University under the advisement of Dr. Charles E. Stroud, a professor and IEEE Fellow who has made significant contributions in the area of BIST for VLSI devices. Lee also spent two months during summer 2007 working with graduate students and professors at IIT Madras as well as Indian industry leaders to produce an international case-study under a National Science Foundation grant. In August 2007, Lee married author and editor, Alicia Cafferty Lerner, in Dublin.

THESIS ABSTRACT

BUILT-IN SELF-TEST FOR INPUT/OUTPUT TILES IN FIELD PROGRAMMABLE GATE ARRAYS

Lee W. Lerner

Master of Science, May 10 2008
(B.E.E., Auburn University, 2006)

135 Typed Pages

Directed by Charles E. Stroud

Very large scale integration (VLSI) circuits use input/output (I/O) cells to both send and receive signals from external resources. Field Programmable Gate Arrays (FPGAs) and System-on-Chips (SoCs) with FPGA cores offer increasingly complex I/O cell resources with new generations of device architectures. I/O cells can often be grouped together to form I/O tiles that can support even more complex features. The ever increasing complexity of I/O tiles is indicative of the need for a reliable testing methodology to ensure the functionality of device resources. Built-in self-test (BIST) is one such testing methodology that incorporates the testing circuitry with the device under test.

A total of 78 BIST configurations are developed to test the I/O tile logic resources and supported I/O standards in Xilinx Virtex-4 FPGAs. The BIST configurations are implemented and verified on Virtex-4 FPGAs. The BIST configurations are generated

for all Virtex-4 FPGAs. The general BIST approach presented in this thesis is applicable to any FPGA or SoC with a FPGA core. The BIST approach can be used for both manufacturing and system level testing for I/O tiles with both bonded and unbonded I/O buffers.

Acknowledgements

It is important to me to thank the many people who have made this thesis possible. I would particularly like to thank Dr. Stroud for his encouragement, for sharing his invaluable knowledge, and his guidance throughout my studies at Auburn University. I would like to recognize Dr. Agrawal for his teachings, attitude towards science, and overall inspiring demeanor. I would also like to thank Dr. Nelson, Dr. Singh, and the remaining professors of the AU Electrical and Computer Engineering Department for providing me with important teachings full of knowledge and wisdom. My educational experience through this department is applicable both inside and outside of the laboratory. I would like to acknowledge Bobby, Daniel, Jie, Mustafa, Sachin, and Sudheer for their friendship and assistance throughout my research. I owe a special thanks to Brad for putting in those late Friday nights that produced valuable contributions to this thesis. My friends and family should also be recognized for their endless encouragement and support. Notably: my father for teaching me how to handle any situation; my mother for reassuring me that I was not dying, rather, just stressed; my older sister, Adrienne, for relating her graduate school experience; my youngest sister, Adeline, for her encouraging wit; and my sister, Amanda, for keeping my spirits high and driving me to class when I did not feel like walking. Finally, I am indebted to my wife, Alicia Cafferty Lerner, for sharing her love of everything that is good in life.

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

Computer software used The entire document, including text, figures, and tables, was prepared using *Microsoft Word*.

TABLE OF CONTENTS

LIST OF FIGURES xi

LIST OF TABLES xiii

1 INTRODUCTION 1
1.1 Overview of Field-Programmable Gate Arrays (FPGAs) 2
1.2 Overview of Input/Output Tiles (I/O Tiles) 4
1.3 Overview of Built-In Self-Test (BIST) 6
1.4 Overview of Prior Work in Testing I/O Tiles 8
1.5 Thesis Statement 9

2 BACKGROUND INFORMATION 10
2.1 Virtex-4 Architecture 10
2.1.1 Virtex-4 Configurable Logic Blocks 11
2.1.2 Virtex-4 Block RAMs 12
2.1.3 Virtex-4 Digital Signal Processors 12
2.1.4 Virtex-4 Digital Clock Managers 13
2.1.5 Virtex-4 Input/Output Cells 14
2.2 Details of Virtex-4 General Purpose Input/Output Tiles 17
2.2.1 ILOGIC 19
2.2.2 OLOGIC 24
2.2.3 SERDES 26
2.2.4 I/O Buffer 29
2.2.5 Programming Interface 36
2.3 Built-In Self-Test for FPGAs 38
2.4 Previous Work in I/O Testing 41
2.5 Thesis Restatement 44

3 I/O TILE BIST FOR FPGAS 46
3.1 BIST Architecture 46
3.2 Configurations to Test I/O Tile Logic Resources 51
3.2.1 Configurations to Test ILOGIC, OLOGIC, and I/O Buffers 51
3.2.2 Configurations to Test SERDES 59
3.3 Configurations to Test I/O Standards 70
3.3.1 Testing Single-Ended I/O Standards 72
3.3.2 Testing Single-Ended I/O Standards Requiring a Reference Voltage 73
3.3.3 Testing Complementary Differential I/O Standards 74
3.3.4 Testing I/O Standards with Digitally Controlled Impedance 77

3.4 Summary of I/O Tile BIST Configurations 78

4 EXPERIMENTAL RESULTS 81
4.1 Generating I/O Tile BIST Configurations 81
4.1.1 Summary of I/O Tile BIST Programs 82
4.1.2 BIST Configuration Generation Procedure 84
4.2 Executing I/O Tile BIST Configurations 88
4.3 Experimental Results Obtained..90
4.3.1 Configurations to test ILOGIC, OLOGIC, and I/O Buffers..................90
4.3.2 Configurations to Test SERDES Logic Resources 92
4.3.3 Configurations to Test I/O Standards 93
4.4 Capabilities and Limitations 95

5 SUMMARY AND CONCLUSION 99
5.1 Summary of Virtex-4 I/O Tile BIST...99
5.2 Areas of Future Research and Development 100
5.3 General Application to FPGAs and SoCs 102

BIBLIOGRAPHY 104

APPENDICES 107

A BATCH FILE FOR GENERATING I/O TILE BIST CONFIGURATIONS 107

B REDUCED CONFIGURATIONS FOR ILOGIC, OLOGIC, AND I/O BUFFERS 119

C LIST OF ACRONYMS 121

LIST OF FIGURES

1.1 Typical FPGA Architecture 3

1.2 Basic PIP Structure 4

1.3 Basic I/O Tile Architecture 5

1.4 Basic BIST Architecture 7

2.1 Example FPGA Input/Output Cell 16

2.2 Virtex-4 I/O Tile Architecture Block Diagram [5] 19

2.3 Virtex-4 Input Logic (ILOGIC) Architecture [5] 21

2.4 Virtex-4 Output Logic (OLOGIC) Architecture [5] 25

2.5 OSERDES 4:1 DDR Timing Diagram from [5] 29

2.6 Virtex-4 I/O Buffer Architecture [5] 30

2.7 Types of Virtex-4 Supported I/O Standards. (a) Single-Ended. (b) Single-Ended Requiring a VREF. (c) Complementary Differential. 32

2.8 Input I/O Tile with Complementary Differential I/O Standard 34

2.9 BIST Architecture presented in [27]. (a) TPG, BUT, ORA connections. (b) Floorplan for first test session. (c) Floorplan for second test session. 39

2.10 BRAM BIST Architecture Presented in [19] 40

2.11 BIST Architecture from [8], also representative of [14][15] 41

2.12 Comparison Based ORA Architecture [9] 42

2.13 Primary I/O Buffer in Atmel FPGA [14] 43

3.1 Basic I/O Tile BIST Architecture 47
3.2 I/O Tile BIST ORA Architecture 50
3.3 I/O Tile to ORA Connections 57
3.4 Virtex-4 I/O Cell in ISERDES/OSERDES Mode of Operation 64
3.5 SERDES BIST BITSLIP Synchronizer Circuit 67
3.6 SERDES BIST TDI Logic Circuit 70
3.7 I/O Tile BIST Configuration for Single-Ended I/O Standards 72
3.8 I/O Tile BIST Configuration for Single-Ended Standards Requiring a V_{REF} 73
3.9 I/O Tile BIST Configuration for Complementary Differential I/O Standards 74
3.10 I/O Tile Components Used for Complementary Differential I/O Standards 76
4.1 All I/O Buffers Under Test of a Virtex-4 FX20 with One Power PC [Screenshot from Xilinx FPGA Editor] 88
4.2 Virtex-4 FX 12 I/O Buffers in Power PC Rows Not Under Test [Screenshot from Xilinx FPGA Editor] 97

LIST OF TABLES

2.1 Virtex-4 Digital Clock Manager Clock Output Ports 14

2.2 Virtex-4 I/O Tiles and I/O Banks for Various Device Families [16][18]................17

2.3 General Purpose I/O Bank 7 for Virtex-4 Package FF668 36

3.1 DSP Signals and Attributes for All Configurations..49

3.2 Configuration Modes of ILOGIC Components 53

3.3 Configuration Modes of OLOGIC Components...54

3.4 Configuration Modes of I/O Buffer Components...55

3.5 BRAM Signals and Attributes for ILOGIC, OLOGIC, and I/O Buffer Testing 56

3.6 TPG and ORA Connections for ILOGIC, OLOGIC, and I/O Buffer Testing..........58

3.7 BIST Configurations to Test ISERDES..60

3.8 BIST Configurations to Test OSERDES 62

3.9 TPG and ORA Connections for SERDES Testing 65

3.10 SERDES BITSLIP Synchronizer Circuit Timing..68

3.11 BIST Configurations to Test I/O Standards Independently.....................................71

3.12 Summary of Virtex-4 I/O Tile BIST Configurations...79

4.1 SERDES BIST Execution Results..93

B.1 Proposed Configuration Modes of ILOGIC Components 119

B.2 Proposed Configuration Modes of OLOGIC Components....................................120

B.3 Proposed Configuration Modes of I/O Buffer Components..................................120

Chapter One

Introduction

Field Programmable Gate Arrays (FPGAs) and configurable System-on-Chips (SoCs) that use FPGA cores are reconfigurable devices that are becoming increasingly important for applications in radiation prone environments, such as space, as well as critical applications that cannot allow for system downtime or the transmission of erroneous data. Circuits used in aircraft systems, for example, are susceptible to single event effects, in which a radiation particle can cause an error in system functionality [1]. The ever increasing purposes such devices serve in these types of applications are indicative of the need for a reliable testing methodology to ensure the functionality of device resources.

A harsh operating environment or critical system application, however, is not required to raise concern over the testing of FPGAs. The basic reliability of FPGAs, like most chips created in a very large scale integration (VLSI) process, is decreasing as the complexity of such devices increases and the feature size decreases. There can be errors at the design level, defects sustained during the manufacturing process, or even faults that occur during system operation [2]. The generally accepted Moore's Law predicts that the transistor feature sizes on a VLSI chip reduce by as much as 10.5% each year, and consequently, the transistor density of a VLSI chip can increase by as much as 22.1% each year [3]. These trends indicate that the transistor count on VLSI chips can nearly

double each year, which is historically accurate. One approach to testing is to use automatic test equipment (ATE). However, ATE has proven to be expensive and complex. Several design for testability (DFT) techniques, such as scan design and built-in self-test (BIST), have been developed to try to overcome the high cost of ATE equipment while still providing an acceptable level of fault coverage for a device [2].

1.1 Overview of Field Programmable Gate Arrays (FPGAs)

On the most fundamental level, an FPGA can be described as a "general-purpose, multi-level programmable logic device that is customized in the package by the end users" [4]. FPGAs are widely used because of their ability to be programmed to implement any digital logic circuit. A typical FPGA architecture consists of an array of programmable logic blocks (PLBs), Input/Output (I/O) cells, and a programmable interconnect network, as exemplified in Figure 1.1. Xilinx Virtex-4 FPGAs also contain columns of other elements that may be configured to implement a system function, such as block RAMs (BRAMs) and digital signal processors (DSPs) [5]. The PLBs in FPGAs typically consist of look-up tables (LUTs), multiplexers, and flip-flops. The PLBs are configured individually to implement low level digital logic. Then, the PLB inputs and outputs are connected together via the programmable interconnect network, also known as the programmable routing network, to implement larger circuits [6]. The routing network also provides signal paths from the I/O cells to the PLBs. The I/O cells provide a means for the FPGA to exchange information with external devices also implemented in a system or directly with system inputs and outputs. In Virtex-4 devices, two I/O cells are grouped together to form an I/O tile [5].

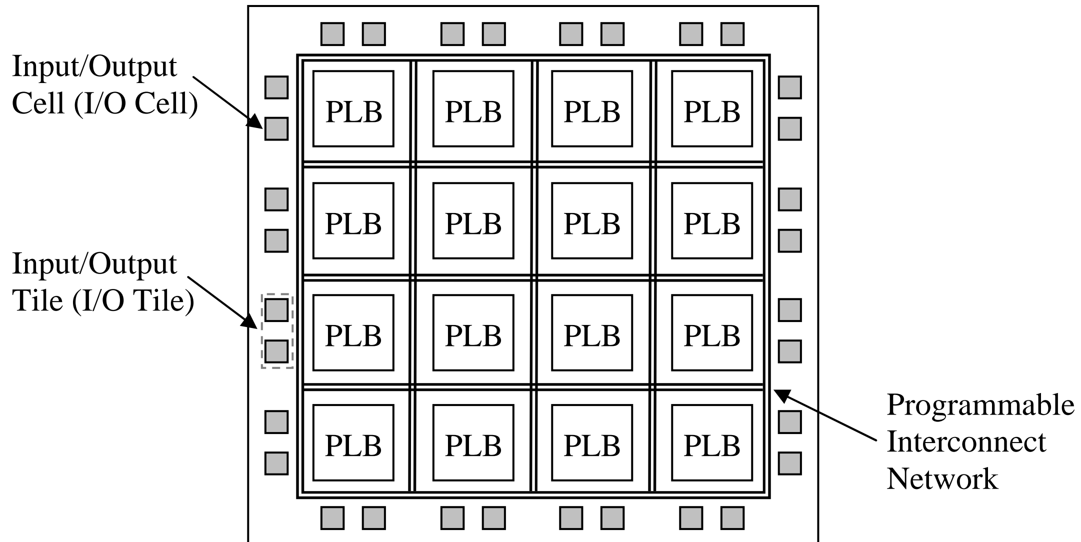


Figure 1.1: Typical FPGA Architecture

The programmable interconnect network comprises segmented wiring that is connected or disconnected through the use of programmable interconnect points (PIPs). A basic PIP structure, illustrated in Figure 1.2, consists of a configuration memory bit that controls a transmission gate placed between two wire segments. The configuration memory bit used to control the transmission gate is often implemented as either a static RAM (SRAM) bit or an antifuse. While a SRAM bit can be set high or low throughout system use, consequently turning on or off the transmission gate, an antifuse permanently joins wire segments according if antifuse is blown. There are several types of PIPs, including break-point PIPs, cross-point PIPs, multiplexer PIPs, and switch-box PIPs [2][7].

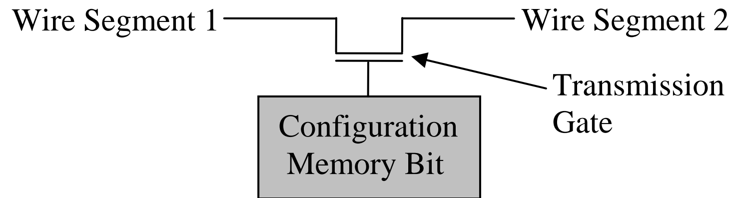


Figure 1.2: Basic PIP Structure

The programmable routing network also consists of local routing, which interconnects PLBs to form larger circuits, and global routing comprising wire segments that span various lengths of PLBs. Global routing consists of horizontal and vertical lines that make use of periodically placed buffers to prevent signal degradation [7].

1.2 Overview of Input/Output Tiles (I/O Tiles)

An I/O tile consists of two I/O cells in a Virtex-4 FPGA, as seen in Figure 1.3. An I/O cell typically consists of an I/O buffer, input logic (ILOGIC) circuitry, output logic (OLOGIC) circuitry, and supporting routing resources [8][9]. The ILOGIC component comprises flip-flops, multiplexers, PIPs, and routing resources used to communicate from the I/O buffer and to the internal device resources. ILOGIC components only receive signals from the I/O buffer while both sending and receiving signals from the internal device resources. An ILOGIC component also contains an input delay element (IDELAY or IODELAY), shown as a grey box in Figure 1.3, used to add a fixed delay time to a signal. The OLOGIC component similarly comprises flip-flops, multiplexers, PIPs, and routing resources. However, an OLOGIC component typically only receives signals from the internal device resources and sends signals to the I/O buffer. The I/O buffer comprises a pad, an input buffer, a tri-state output buffer, various digital logic resources such as multiplexers or PIPs, routing resources, and various analog

logic resources such as pull-up or pull-down transistors [5][10]. The ILOGIC, OLOGIC, and I/O buffers can be configured to operate in various modes and standards (i.e. slew rate, drive strength, and input delay are adjustable). Also, there are some additional connections between ILOGIC, OLOGIC, and I/O buffers that will be discussed in more detail in subsequent chapters.

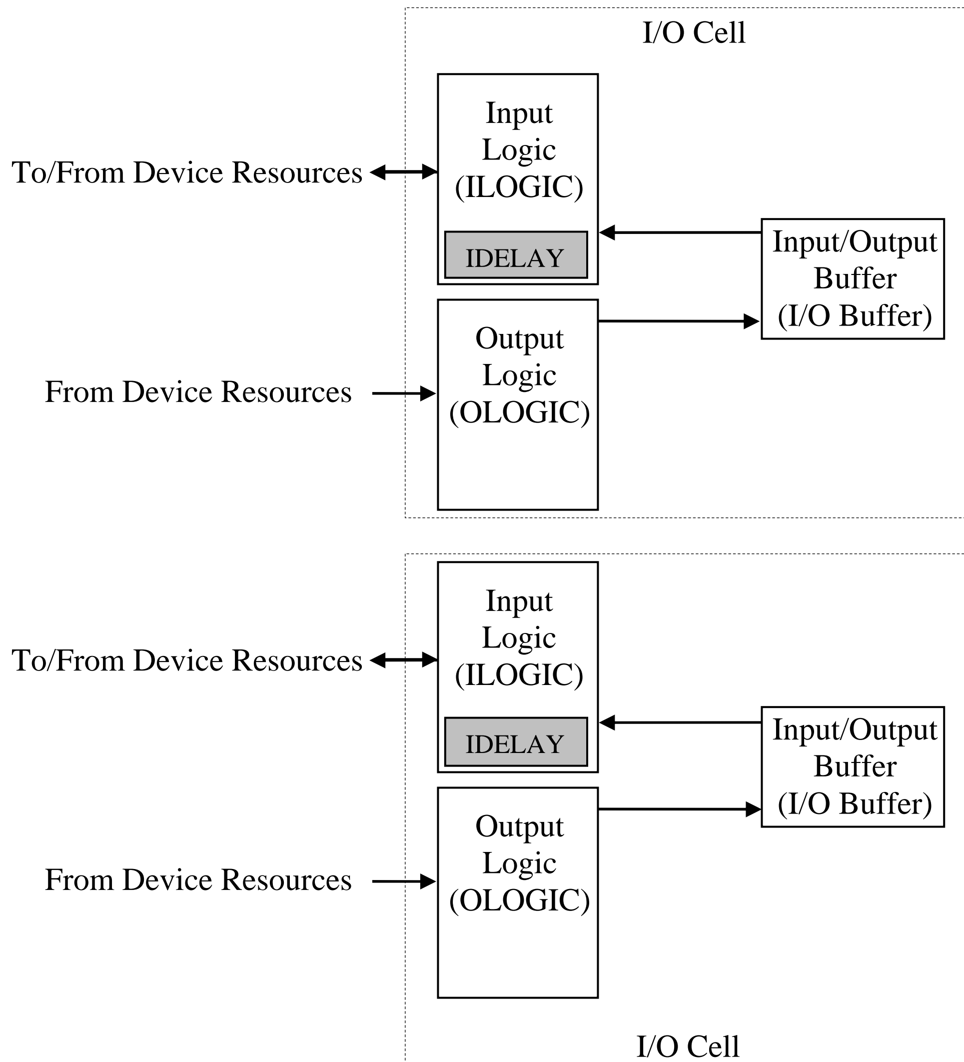


Figure 1.3: Basic I/O Tile Architecture

The pad acts as the interface between the die and package, and can be bonded or unbonded. A pad is considered bonded if it is wire-bonded to a pin or ball on the device package allowing communication between the pad and other system signals external to the FPGA. An unbonded pad is not wire-bonded to a pin or ball on the device package and cannot communicate with external system signals. However, the resources of an I/O cell with an unbonded pad may still be used by the FPGA to implement system functions [8][11]. Therefore, both I/O cells with bonded and unbonded pads should be functionally tested.

I/O tiles can operate in various modes as single-ended or differential primitives. When operating as single-ended primitives, the I/O cells of an I/O tile typically function independently. When operating as differential primitives, the two I/O cells of an I/O tile are paired together to perform differential operations. Both single-ended and differential primitives allow for I/O cells to be configured as input, output, or bidirectional cells [5][10]. Input cells use the input buffer and ILOGIC resources to communicate a signal from an external source to the internal resources of the FPGA. In contrast, output cells use the output buffer and OLOGIC to communicate internal device signals off chip. Bidirectional cells use both ILOGIC and OLOGIC resources to allow internal FPGA resources to communicate both to and from the I/O buffers or external signals.

1.3 Overview of Built-In Self-Test (BIST)

Historically, most DFT techniques aim to satisfy three principle goals: high fault coverage, fast testing speed, and low cost. BIST is one such DFT technique that can satisfy these principle goals, as well as overcome several other major testing problems,

such as test generation and application. BIST is a DFT technique in which the hardware for testing a device is built into or around the device itself [12]. Built-in hardware allows for increased controllability and observability of the potential faults in a device to be tested. The basic methodology of BIST can be described as: "to design a circuit so that the circuit can test itself and determine whether it is 'good' or 'bad' (fault-free or faulty, respectively)" [2].

A basic BIST architecture, shown in Figure 1.4, typically consists of essential BIST circuitry and a circuit under test (CUT). The essential BIST circuitry, shown as grey boxes in Figure 1.4, typically includes a test pattern generator (TPG), output response analyzer (ORA), and test controller [12]. The TPG generates a sequence of test patterns that are applied to the CUT to activate potential fault sites. Then, the outputs of the CUT are compacted in the ORAs which in turn produces a Pass/Fail result [2]. The test controller can be used to implement a number a functions, such as starting the BIST, initializing the CUT, or providing an indication that the BIST has finished.

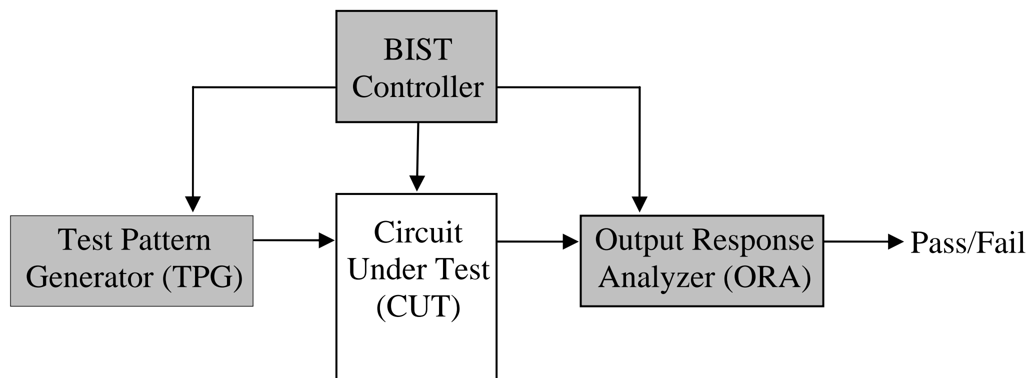


Figure 1.4: Basic BIST Architecture

BIST for FPGAs, unlike BIST for most other circuitry, can be implemented such as not to incur system performance or area overhead penalties, meaning system execution

time remains high and DFT costs remain low. Using only additional memory (external to the FPGA) to store BIST configurations, there is no additional circuitry permanently added into the FPGA design itself. FPGAs are configured repeatedly with different BIST configurations stored in the external memory to test various resources of the FPGA [8]. Consequently, BIST proves to be a testing technique for FPGAs that well satisfies the principal goals of DFT.

1.4 Overview of Prior Work in Testing I/O Tiles

A number of BIST architectures have been developed to test the various resources of FPGAs. However, there has been little focus on developing BIST approaches for FPGA I/O cells. A BIST for I/O speed testing was developed in reference [13]. The architecture presented in reference [13] used a delay-locked loop based BIST to test the setup and hold times of chip I/O buffers and registers. While effective for testing I/O speed, the architecture presented in reference [13] does not fully test the resources contained in I/O tiles.

The most significant contributions to BIST for FPGA I/O resources were developed in references [8],[9],[14], and [15]. These references developed a general BIST approach applicable to any FPGA or SoC with an FPGA core. PLBs are used to implement TPGs that supply test patterns to I/O cells. The I/O cells are configured as bidirectional buffers such that they receive test patterns from the TPGs and send results to the ORAs. The output responses of the I/O cells are then circularly compared with identically configured I/O cells by ORAs which are also implemented in PLBs [8][9][14][15]. References [8] and [9] suggest that 98.6% fault coverage can be obtained

with seven BIST configurations for the I/O cells of a Virtex-4 device. However, these are theoretical results based on simulations and have not been implemented in an actual Virtex-4 FPGA.

1.5 Thesis Statement

FPGAs are ever increasing in transistor count and logic resources, lending them to be more susceptible to faults in manufacturing or system operation. Also, as these increases occur, FPGAs become harder to test to ensure their functionality. To date, little work has been done targeting the I/O tiles of Xilinx Virtex and Spartan series FPGAs. This thesis presents a general BIST architecture developed to test the functionality of the various resources present in I/O tiles of FPGAs as well as specific BIST configurations developed and implemented in Xilinx Virtex-4 devices.

The remainder of this thesis is organized as follows: Chapter 2 provides architectural details of Virtex-4 FPGAs, an overview of prior work in BIST for FPGAs, and previous work pertaining to I/O testing. Chapter 3 presents BIST configurations developed specifically to test the I/O tiles present in Xilinx Virtex-4 FPGAs. Chapter 4 provides experimental results obtained from testing the configurations developed for Virtex-4 I/O tile BIST. Finally, Chapter 5 provides a summary and conclusion of the thesis as well as suggestions for future research and application.

CHAPTER TWO

BACKGROUND INFORMATION

This chapter provides an overview of FPGAs and BIST techniques that have been developed for them that are similar to the one presented in this thesis. Chapter 2 begins with discussion of the architectural details of the Xilinx Virtex-4 FPGA, including the programmable logic blocks, block RAMs, digital signal processors, digital clock managers, I/O tiles, and programming interface. The chapter then goes on to provide background on previous BIST techniques that have been developed for FPGAs, including BIST for programmable logic blocks, BIST for block RAMs, and BIST for I/O cells. Finally, the chapter concludes with a thesis restatement in greater detail than written previously.

2.1 Virtex-4 Architecture

Xilinx Virtex-4 FPGAs are fabricated on a 90-nm copper process using 300-mm wafer technology. Each Virtex-4 FPGA contains a variety of configurable elements and embedded IP cores that are optimized to implement high-density and high-performance system designs. Some of the components common to all Virtex-4 devices include: configurable logic blocks (CLBs) that provide combinatorial and sequential logic; block RAM modules; digital signal processors; digital clock managers that perform self-

calibrating high-speed clock operations and clock distribution; and input/output tiles enhanced for source-synchronous applications in which the data clock is provided by the data source. Each component is interconnected by an array of routing switches that form a hierarchical general routing matrix that supports high-speed system designs [5][16].

Virtex-4 FPGAs are classified into three families: LX for high-performance logic applications; SX for high-performance DSP applications; and FX high-performance embedded platform applications. Some device families include features in addition to the common features listed above. For example, FX devices contain at least one PowerPC processor core. FX devices also contain RocketIO Multi-Gigabit Transceivers that support up to 6.5 Gb/s data rates [17]. The three device families are further divided into package categories that define chip area and pinout specifications such as the maximum number of I/O pins. Device area and general purpose I/O pin count ranges from 17x17 mm in the SF363 with 240 maximum I/O pins, to 40x40 mm in the FF1517 package with 768 maximum general purpose I/O pins [5][16][18].

2.1.1 Virtex-4 Configurable Logic Blocks

Virtex-4 FPGAs contain an array of NxM programmable logic blocks, also known as configurable logic blocks (CLBs), where N is the number of rows and M is the number of columns of CLBs. Array sizes range from 64x24 CLBs in the LX15 to 192x116 CLBs in the LX200. The CLB array is the main resource available on an FPGA for implementing combinatorial and sequential logic circuit designs. Each CLB joins the interconnect network of the FPGA through the use of a switch matrix.

A single Virtex-4 CLB contains four interconnected slices, which are further grouped into pairs. Each slice contains two logic function generators implemented as 4-input look up tables. A look up table is the basic unit of configurable logic and implements combinational logic as a 2^nx1 memory, where n is the number of inputs, and each bit in the LUT memory corresponds to a configuration bit [4]. Each slice also contains two storage elements (flip-flop or latch), wide function multiplexers, carry logic, and arithmetic gates [5].

2.1.2 Virtex-4 Block RAMs

As with other components on a Virtex-4 FPGA, the total number of block RAM memories depends on the family and size of the device. The number of BRAMs on a Virtex-4 range from 36 in a FX12 to 552 in a FX140 and are organized into columns on the FPGA. A BRAM is equal in height to four Virtex-4 CLBs. Virtex-4 BRAMs store 18K bits of data, and act as true dual port RAMs with dual address and data input and output lines. The memory array can be configured with different aspect ratios offering different numbers of data outputs and RAM word depths, word widths, and number of parity bits. Typical aspect ratios range from 16Kx1 to 512Kx36. The dual BRAM ports can be independently configured with different aspect ratios [5][19]. Virtex-4 BRAM memory content is defined or cleared by the configuration memory bitstream.

2.1.3 Virtex-4 Digital Signal Processors

Digital signal processors are used to analyze and manipulate signals by implementing digital signal processing algorithms. They can also be configured to act as simple arithmetic circuits, such as counters. Similarly to BRAMs, the DSPs on Virtex-4

devices are organized into vertical columns on the FPGA. The number of DSPs on a Virtex-4 device ranges from 32 on a LX15 to 512 on a SX55. There are two DSP slices in every DSP tile of a DSP column. A DSP tile is equal in height to four Virtex-4 CLBs [20].

2.1.4 Virtex-4 Digital Clock Managers

Digital clock managers (DCMs) are used to produce or manage clock signals for a Virtex-4 FPGA. The number of DCMs on a Virtex-4 device range from four on the LX15, to 20 on the FX140, and are always located in the center column of every Virtex-4 device. They provide powerful clock management features such as clock deskew, phase shifting, frequency synthesis, and dynamic reconfiguration. The DCMs are instantiated in a continuous calibration mode in every Virtex-4 configuration if not used in the user design to avoid the effects of negative-bias temperature instability [21]. The predicted motion vector (PMV) module is used to maintain unused DCM components set in a continuous auto calibration state. The PMV module is an on-chip oscillator that is not readily accessible to user designs.

One of the most powerful features of a DCM is its ability to produce a clock signal that is some factor or fraction of its input clock signal. Separate outputs of a DCM can be used to produce either multiplied or divided clocks, as seen in Table 2.1. Some DCM clock outputs are used to produce a phase shifted clock relative to the input clock. For example, the CLK2X and CLK2X180 outputs provide a doubled clock frequency relative to that of the DCM input clock. The CLK2X180 output phase shifts the clock signal by 180 degrees. The CLKDV output produces a clock output signal frequency that

is a specified fraction of the DCM clock input frequency. CLKFX and CLKFX180 output a clock signal frequency that is derived from simultaneous frequency division and multiplication (specified by the user) of the DCM clock input frequency [5].

Table 2.1: Virtex-4 Digital Clock Manager Clock Output Ports

Clock Output Port	CLKIN Frequency Adjustment	Phase Shift
CLK0	None	0°
CLK90	None	90°
CLK180	None	180°
CLK270	None	270°
CLK2X	Multiplied by 2	0°
CLK2X180	Multiplied by 2	180°
CLKDV	Divided by integer range 1.5 to16	0°
CLKFX	Multiplied by integer range 2 to 32 Divided by integer range 1 to 32	0°
CLKFX180	Multiplied by integer range 2 to 32 Divided by integer range 1 to 32	180°

2.1.5 Virtex-4 Input/Output Cells

General purpose I/O cells provide a means for the internal logic of FPGAs to communicate with external sources. Dedicated I/O cells, on the other hand, are also needed so that the FPGA has a means of being configured with a bitstream downloaded to the device. Dedicated I/O cells in Virtex-4 devices include configuration pins, such as JTAG MODE, PROGRAM_B, DONE, etc., that are all contained within a dedicated I/O bank. The dedicated I/O bank in Virtex-4 FPGAs is BANK 0 and contains a constant voltage supply, $V_{CC_CONFIG,}$ and operates at a LVCMOS level. Bank 0 is located in the center column of Virtex-4 devices. Dedicated I/O cells can not be used as general purpose I/O cells at any time during system operation. Dual-purpose I/O cells, however, can operate as general purpose I/O cells after device configuration is completed.

Examples of dual-purpose I/O cells are the SelectMAP Data pins located in Bank 2 of a Virtex-4 device [22].

General purpose I/O cells consist of input logic, output logic, and an I/O buffer, as seen in Figure 2.1. The input logic, output logic, and I/O buffer can be configured to operate in input, output, or bidirectional mode. When operating in input mode, an I/O cell uses the pad, input buffer, and input logic resources to communicate a signal from an external source to the internal resources of the FPGA. In contrast, when operating in output mode, an I/O cell uses the output logic and output buffer to communicate internal device signals off chip. When operating in bidirectional mode, an I/O cell uses both input and output logic resources to allow internal FPGA resources to communicate both to and from the I/O buffers or external signals. The input and output signal pins connecting the I/O buffer to the input and output logic are labeled as T, O, and I in Figure 2.1. The input logic can either route I/O buffer data directly to internal device resources or perform various operations on the data before routing it to internal device resources. Data can be inverted, delayed, or sent via a registered or unregistered path through the use of flip-flops, multiplexers, delay elements, and inverters. Output data logic can also perform various operations, such as inverting or registering, on data in the data path coming from internal device resources before it is routed to the I/O buffer. Aside from the output data path, output logic typically supports a tri-state control data path. The tri-state data path allows the output buffer to be tri-stated, or put into a high impedance state. The output buffer is typically tri-stated when the I/O cell is used in an input only mode [8].

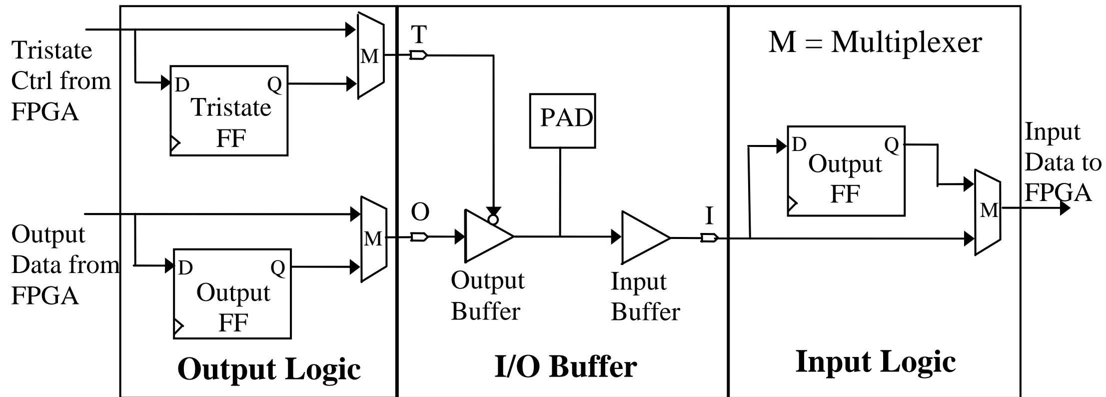


Figure 2.1: Example FPGA Input/Output Cell

Most input and output signal properties are managed by the I/O buffer, such as output drive strength, pull, slew rate, and I/O standard. The output drive strength parameter is chosen to match power and loading on the pad. Pull transistors can be configured as pull-up, pull-down, or keeper mode. Pull-up and pull-down modes supply a weak logic high or low, respectively, to the pad to maintain known voltage levels. Keeper mode retains the last known logic value until a definitive logic value arrives. Slew rate limits the rate of change of a signal. I/O standards define specific voltage requirements for the I/O cell. For example, I/O standards requiring a differential amplifier input may require an external reference voltage, V_{REF}, to be supplied to the amplifier. I/O standards also require a constant supply voltage, V_{CCO}, to power all or part of the I/O cell. V_{CCO} is used to set the output voltage level of an output buffer [8][23].

As previously stated, I/O cells in FPGAs often contain logic resources that can be used to implement a system function, much like the use of CLBs. The logic resources contained in modern I/O cells of FPGAs can include flip-flops, latches, multiplexers, and even delays elements. The logic in an I/O cell is often utilized to implement a system function even if no input or output from the FPGA is realized for that particular I/O cell.

Furthermore, the logic resources in I/O cells that are unbonded are also used to implement system functions, in which case there can be no input or output between the FPGA and external signals [8][11].

2.2 Details of Virtex-4 General Purpose Input/Output Tiles

This thesis focuses on test development for Virtex-4 FPGA general purpose input/output tiles. Virtex-4 I/O cells are grouped into I/O banks. An I/O bank consists of 32 tiles spanning 32 CLBs and two clock regions. The number of I/O banks in a Virtex-4 FPGA depends on the device family, as seen in Table 2.2. Not every I/O bank contains general purpose I/O cells. For example, Bank 0 in all Virtex-4 devices contains only configuration and dedicated signals [5].

Table 2.2: Virtex-4 I/O Tiles and I/O Banks for Various Device Families [16][18]

	Package	**SF363**	**FF668**	**FF1148**	**FF1513**	**FF672**	**FF1152**	**FF1517**	**I/O Banks**
	Pins	**240**	**448**	**768**	**960**	**352**	**576**	**768**	
I/O Tiles Per Device	**LX15**	120	160						9
	LX25	120	224						11
	LX40		224	320					13
	LX60		224	320					13
	LX80			384					15
	LX100			384	480				17
	LX160			384	480				17
	LX200				480				17
	SX25		160						9
	SX35		224						11
	SX55			320					13
	FX12	120	160						9
	FX20					160			9
	FX40					176	224		11
	FX60					176	288		13
	FX100						288	384	15
	FX140							384	17

A Virtex-4 I/O tile consists of two I/O cells, as seen in Figure 2.2. The top and bottom I/O cells are sometimes referred to as master and slave, respectively. The total

components of an I/O tile include: two ILOGICs used for input data; two OLOGICs used for output data; and two I/O buffers used for input and output data. The two I/O buffers are connected together to support differential I/O standards that require a pair of inputs or outputs. The ILOGIC components of an I/O tile are also connected together to support larger data widths in deserialization operations. Similarly, the OLOGIC components of an I/O tile are connected together to support large data widths in serialization operations. The input clocks of the two ILOGIC or OLOGIC components in an I/O tile can be shared or not shared. The set and reset (SR and REV, respectively) signals are shared between the ILOGIC components of an I/O tile. Similarly, the SR and REV signals are shared between the OLOGIC components of a single I/O tile [5][10]. The architecture and functionality of all three types of components of a Virtex-4 I/O tile are discussed in detail in this section.

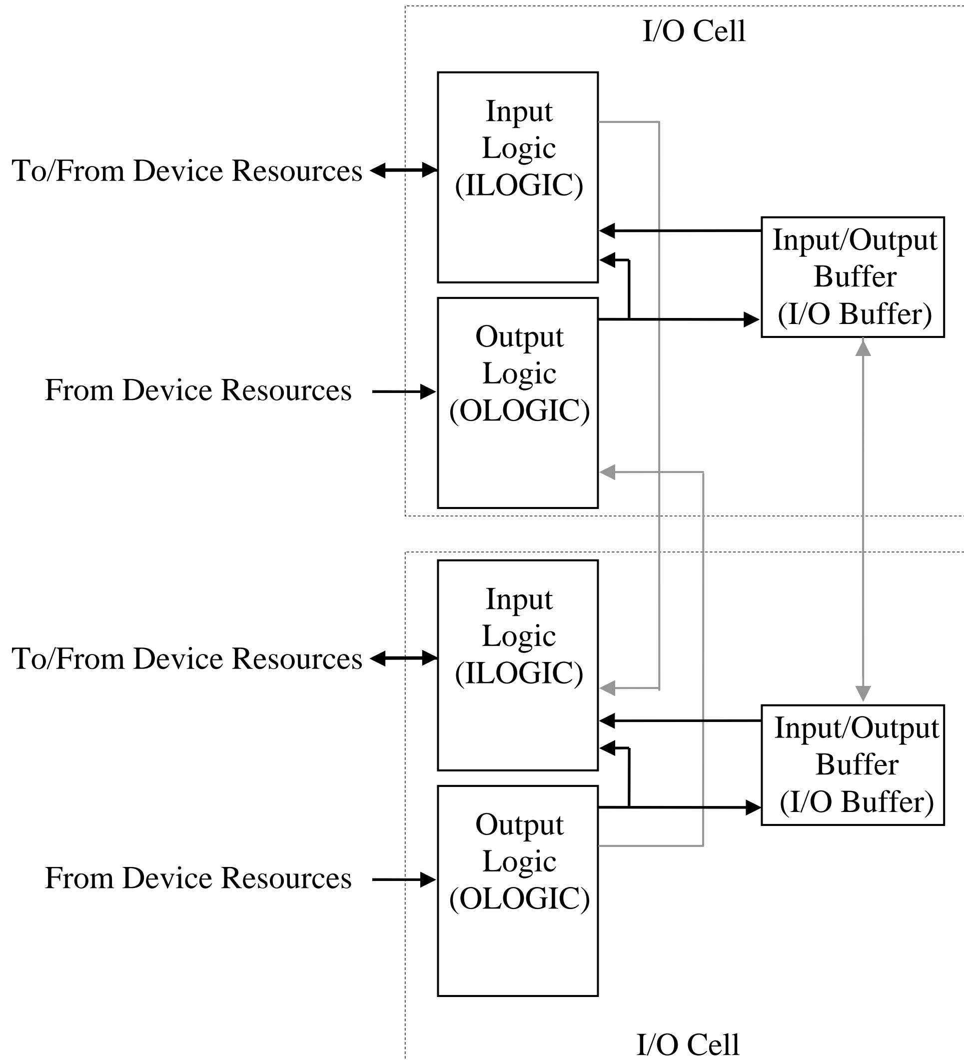


Figure 2.2: Virtex-4 I/O Tile Architecture Block Diagram [5]

2.2.1 ILOGIC

A Virtex-4 ILOGIC component, seen in Figure 2.3, contains logic resources to support both combinational and registered I/O and tristate output control. An ILOGIC also contains optional inverters on most input signals and a programmable delay module for fine delay tuning. The combinatorial path, seen as the non-shaded path from input to output in Figure 2.3, provides a direct connection from the I/O buffer input driver to the

FPGA internal resources [5]. This path contains minimal logic to propagate the signal including two multiplexers and a PIP. The combinatorial path is sometimes referred to as a routethrough in this thesis.

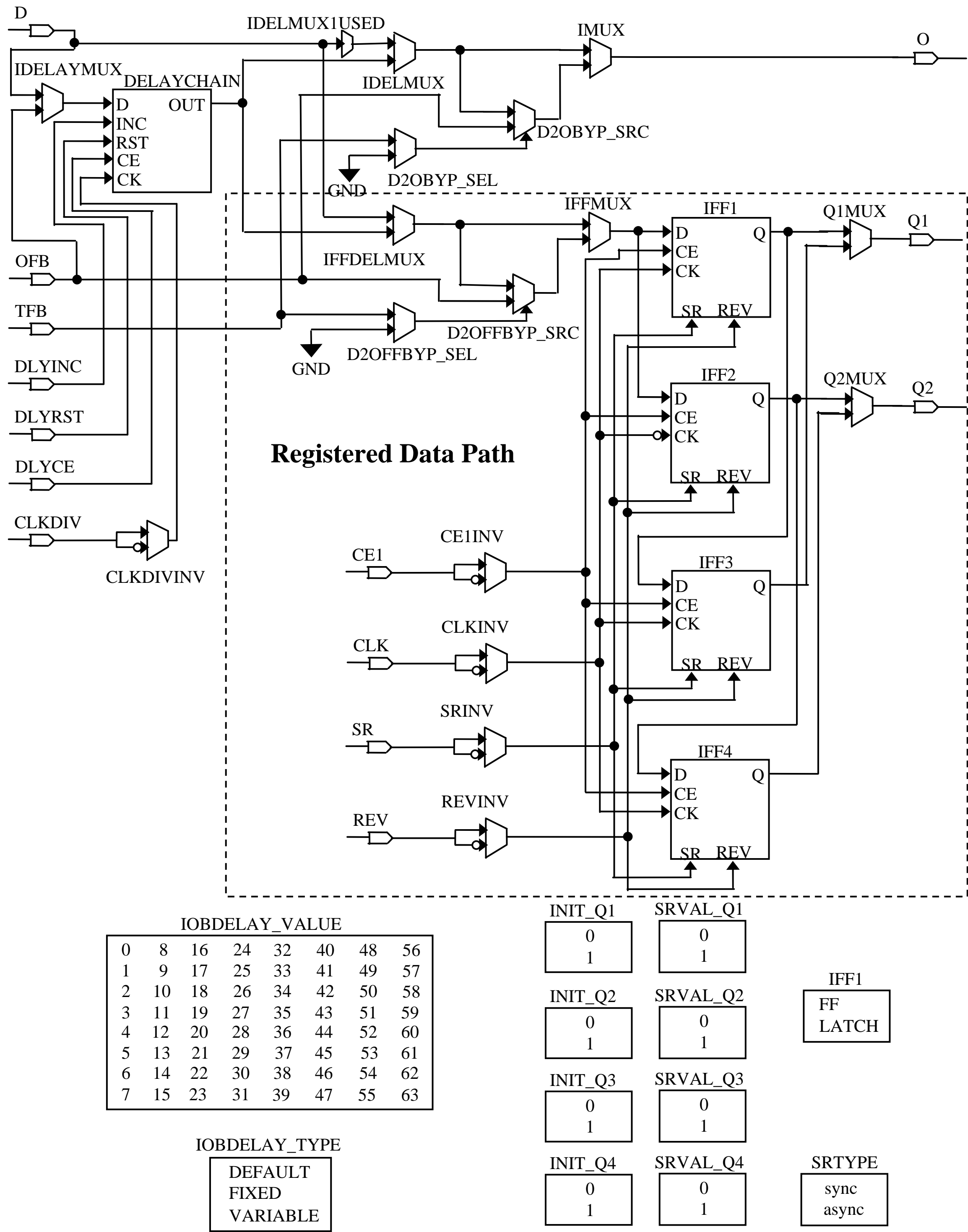


Figure 2.3: Virtex-4 Input Logic (ILOGIC) Architecture [5]

An ILOGIC component contains four registers (flip-flops) to support a registered I/O data path, seen in the shaded region of Figure 2.3. The initial output value of each register is set by the INIT_Q1, INIT_Q2, INIT_Q3, and INIT_Q4 attribute. IFF1 can be used in single-data-rate (SDR) operation in which data is present on ILOGIC outputs after each clock cycle. IFF1 can also be configured as a level sensitive latch or a D type flip-flop. The remaining flip-flops, IFF2, IFF3, and IFF4, are used to implement double-data-rate (DDR) input. All four registers share a common active high clock enable signal, CE1, and synchronous or asynchronous set/reset via SR and REV signals. The SR control sets the four registers into the states specified by SRVAL attributes. The REV signal forces the four registers into the opposite states specified by the SRVAL attributes. The SRVAL values can be set for each register individually. However, all registers are configured as having a synchronous or asynchronous SR signal, specified by the SRTYPE attribute. The SR and REV signals are also shared with adjacent ILOGIC blocks [5].

Clock signals are not shared between adjacent ILOGIC blocks. All clock signals of a Virtex-4 I/O cell are fully multiplexed to prevent clock sharing. For DDR mode, an ILOGIC uses only a single clock input. The ILOGIC supports three modes of DDR operation: opposite-edge; same-edge; and same-edge pipelined. These three modes of DDR operation use the flip-flops in various configurations. In opposite-edge mode, IFF1 and IFF2 are configured as rising-edge triggered with a common data input signal. IFF2 is supplied an inverted version of IFF1's clock signal. In this scheme, the ILOGIC is ready to pass data from its Q1 output on a rising clock edge and from its Q2 output on a falling clock edge. In same-edge mode, IFF4 is configured as rising-edge triggered with

a non-inverted clock signal and is placed on the output path of IFF2. This allows the ILOGIC to pass data on its Q1 and Q2 outputs on the same clock edge with only a one clock cycle separation for data on Q1 and Q2. In same-edge pipelined mode, all four registers of the ILOGIC are used. Same-edge pipelined mode is similar to same-edge mode, except that IFF3 is placed on the output of IFF1. This allows for data to be present on Q1 and Q2 on the same clock edge [5].

Each ILOGIC component contains a programmable input delay module, IDELAY, which can be applied to the both the combinational and registered data paths. The IDELAY module is a 64-tap wrap-around delay module that provides a guaranteed fixed tap resolution. The IDELAY module is used for fine delay tuning of input signals independent of process, voltage, and temperature variation. The module can be configured to operate in three different modes: DEFAULT; FIXED; and VARIABLE. In DEFAULT mode, zero-hold time delay is applied. In FIXED mode, a hold time corresponding to the specified tap value between 0 and 63 is applied. Each tap represents an added 78 pico-second delay. The VARIABLE mode also adds a hold time delay corresponding to the specified tap value. However, in VARIABLE mode, the tap value can be incremented or decremented. For either FIXED or VARAIABLE modes of operation to work correctly, an IDELAYCTRL module must be instantiated in any given I/O bank configured with such parameters. The IDELAYCTRL elements continuously calibrate the individual IDELAY modules in each I/O bank. To calibrate the IDELAY modules correctly, the IDELAYCTRL modules must be given a 200 MHz +- 10% reference clock signal [5].

2.2.2 OLOGIC

Virtex-4 OLOGIC components, as seen in Figure 2.4, contain two sets of functionally equivalent registers. The top set of three registers, TFF1, TFF2, and TFF3 as seen in the non-shaded region in Figure 2.4, is used for tristate control of the I/O buffer. The bottom set of registers, OFF1, OFF2, and OFF3 as seen in the shaded region of Figure 2.4, is used for output data. The tristate register set shares a common active high clock enable, TCE. Similarly, the data register set shares a common active high clock enable, OCE. All six OLOGIC registers share common SR and REV signals with similar characteristics described for the ILOGIC. Also, like the ILOGIC, the topmost flip-flop of each set can be used to implement a D-type flip-flop or level sensitive latch. The remaining registers in each set are used to support output DDR. SDR and DDR modes operate in the same manner as described for the ILOGIC. However, unlike ILOGIC, the OLGIC resources do not have a fourth register to support same-edge pipelined mode. Output multiplexers are used to select between flip-flop data paths. An OLOGIC also has optional inverters for control signal active levels [5].

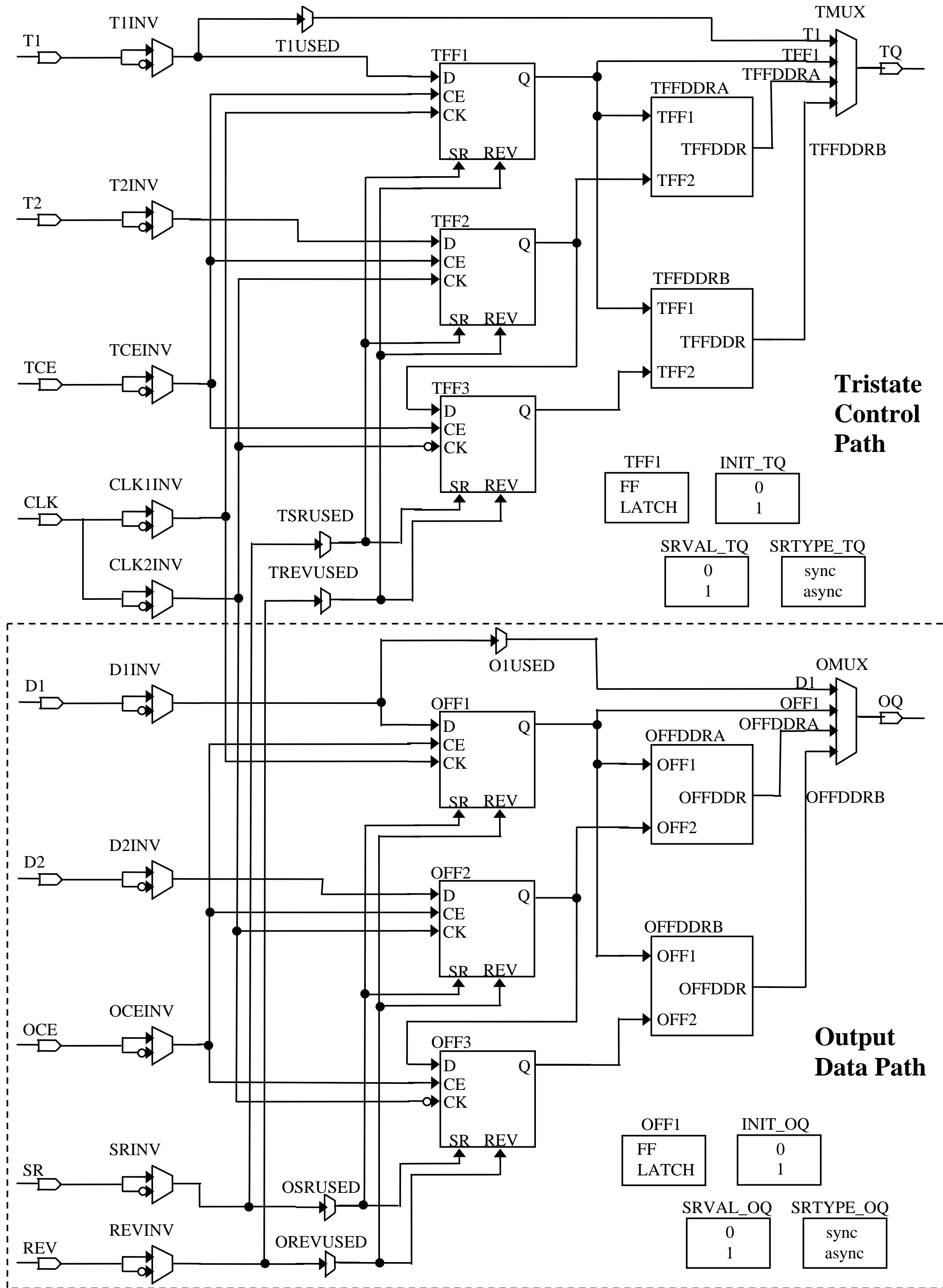


Figure 2.4: Virtex-4 Output Logic (OLOGIC) Architecture [5]

2.2.3 SERDES

ILOGIC and OLOGIC components can be configured as input serial-to-parallel logic resources (ISERDES) and or output parallel-to-serial logic resources (ORSERDES) components respectively. ISERDES allow for high speed serial-to-parallel conversion of data input to the FPGA from external resources. OSERDES allow for high speed parallel-to-serial data conversion for data output from the FPGA's internal resources. The ISERDES and OSERDES components can operate in SDR mode with data widths of two, three, four, five, six, seven or eight. They can also operate in DDR mode with data widths of four, six, eight, or ten. ISERDES/OSERDES operate in master/slave mode when the two ISERDES/OSERDES blocks of each I/O tile are joined together via the shift lines to allow for parallel-to-serial/serial-to-parallel data conversion widths higher than six.

An ISERDES component contains a serial input from the I/O buffer, SR and REV inputs, clock inputs, an input clock enable module, an IDELAY module with corresponding input control lines, a BITSLIP module, six registered outputs, a combinatorial output, and two shift lines. The SR is an active high reset to the ISERDES. The REV line cannot be used and should be grounded.

The clock signals of an ISERDES include a CLK, CLKDIV, and OCLK inputs. The CLK signal is used as the clock to the serial input data stream. The CLKDIV signal is a divided version of the CLK signal used to control parallelization of the input data as well as the ISERDES sub modules. The OCLK signal is a high speed clock line that can be used to implement high speed memory applications. When the ISERDES is used in MEMORY mode, the OCLK is used to transfer strobe-based memory data on a

peripheral device to the free running clock domain of the FPGA. The NETWORKING attribute of the ISERDES is used when I/O that does not require the OCLK is used. The input clock enable module contains a multiplexer that switches between two registered clock enable signals on opposite edges of the CLKDIV signal. The output of the clock enable module controls the clock enables of the input registers located in the ISERDES. The clock enable module can also be configured such that only one unregistered clock enable is selected to directly control the clock enables of the ISERDES input registers.

The BITSLIP module can be used to shift output line data one bit at a time until the ISERDES output lines are synchronized or ordered correctly to properly reflect the serial input data. When invoked, the BITSLIP operation is performed synchronous to the CLKDIV input causing the parallel outputs of the ISERDES to be shifted to allow every pattern of a repeating serial input. The shifting pattern for SDR mode is: shift output pattern left one position every clock cycle. The shifting pattern for DDR mode is: alternate between a shift left by one and a shift right by three positions. The BITSLIP activation signal must be asserted for one and only one clock cycle and then deasserted for at least one clock cycle before performing another BITSLIP operation. There is a two clock cycle latency before a BITSLIP operation is reflected on the ISERDES outputs. The BITSLIP operation should be disabled when the parallel data is synchronized to the correct position on the ISERDES outputs [5].

An OSERDES component also contains CLK, CLKDIV, SR, REV, and shift signals lines that are used as described for ISERDES. In addition, OSERDES contain six registered data inputs, an active high output clock enable input, four active high parallel

tristate inputs, an active high tristate clock enable signal, a serialized data output, and a tristate data output.

The SERDES mode of operation is demonstrated in Figure 2.5. This example shows an OSERDES 4:1 operation in DDR mode with four tristate inputs. At clock event one, the data present on D1-D4 (E, F, G, and H respectively) is captured on the rising edge of the CLKDIV signal. Clock event two occurs at the next rising edge of the CLK signal, at which time the data captured at clock event one begins to be transmitted to the output signal. A new data value is transmitted at each edge of the CLK signal in DDR mode. Because this example shows 4:1 data serialization in DDR mode, the second clock event is a falling and rising edge after the first. Note that the G data value from input D3 is not transmitted to the output because the third tristate signal is a logic high value when the data is captured at clock event one.

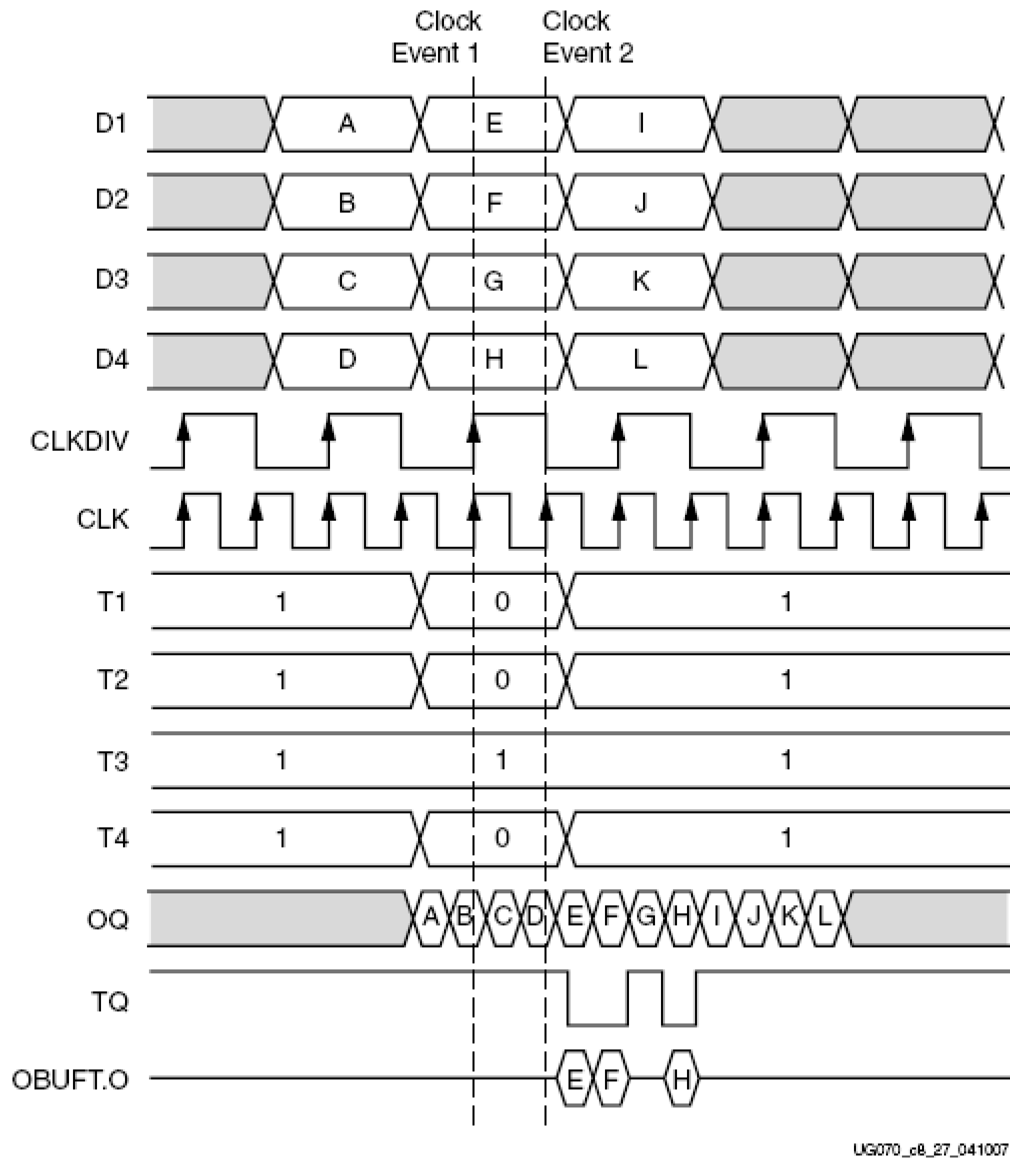
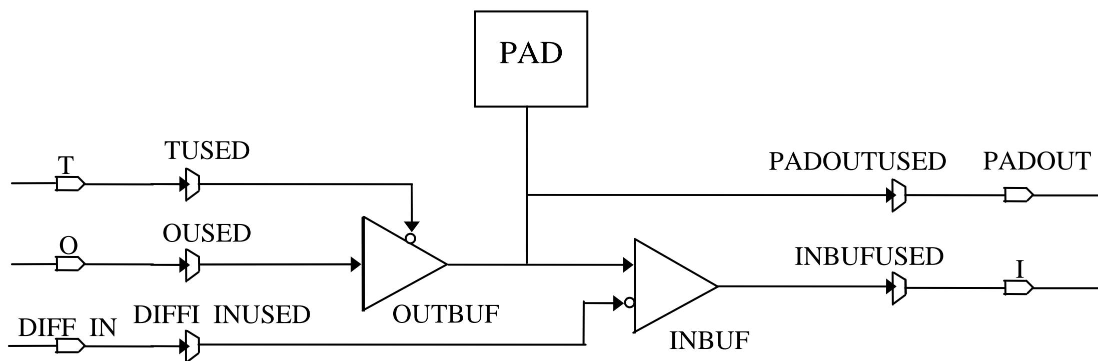


Figure 2.5: OSERDES 4:1 DDR Timing Diagram from [5]

2.2.4 I/O Buffer

The architecture of a Virtex-4 I/O buffer can be seen in Figure 2.6. The I/O buffers support a wide variety of standard interfaces including both single-ended and differential I/O standards. I/O buffers include programmable control of output drive strength, slew rate, and on-chip termination. A Virtex-4 I/O buffer contains input, output, and tristate drivers. The I/O buffer architecture consists of PIPs to activate or deactivate

I/O signals, an output buffer, an input buffer, and a PAD that is either bonded or unbonded. The output buffer, OUTBUF, provides a data path from the OLOGIC to the PAD of an I/O cell. The OUTBUF can also be tristated through the use of the active low tristate signal. The input buffer, INBUF, allows a path for data coming from the PAD sent to the ILOGIC of an I/O cell. The INBUF has a differential input to support I/O standards requiring differential operations, or single-ended standards requiring a voltage reference. An I/O buffer also has a PADOUT line which connects directly to an adjacent I/O buffer's differential input buffer to support complementary differential I/O standards [5].



IO STANDARD

LVTTL	GTL
LVCMOS33	GTLP
LVCMOS25	SSTL2_I
LVCMOS18	SSTL2_II
LVCMOS15	SSTL18_I
PCI33_3	SSTL18_II
PCI66_3	LVPECL_25
PCIX	LVDS_25
HSTL_I	LVDSEXT_25
HSTL_II	BLVDS_25
HSTL_III	ULVDS_25
HSTL_IV	LDT_25
HSTL_I_18	RSDS_25
HSTL_II_18	DIFF_SSTL2_II
HSTL_III_18	DIFF_SSTL18_II
HSTL_IV_18	DIFF_HSTL_II
HSTL_I_12	DIFF_HSTL_II_18

TERMINATED IO STANDARD

LVDCI_33	HSTL_I_DCI
LVDCI_25	HSTL_II_DCI
LVDCI_18	HSTL_III_DCI
LVDCI_15	HSTL_IV_DCI
LVDCI_DV2_25	HSTL_I_DCI_18
LVDCI_DV2_18	HSTL_II_DCI_18
LVDCI_DV2_15	HSTL_III_DCI_18
HSTL_II_T_DCI	HSTL_IV_DCI_18
HSTL_II_T_DCI_18	SSTL2_I_DCI
SSTL2_II_T_DCI	SSTL2_II_DCI
SSTL18_II_T_DCI	SSTL18_I_DCI
DIFF_ SSTL2_II_T_DCI	SSTL18_II_DCI
DIFF_ SSTL18_II_T_DCI	LVDS_25_DCI
DIFF_HSTL_II_DCI	LVDSEXT_25_DCI
DIFF_HSTL_II_DCI_18	HSLVDCI_33
GTL_DCI	HSLVDCI_25
GTLP_DCI	HSLVDCI_18
	HSLVDCI_15

DRIVE SRENGTH

2	12
4	16
6	24
8	

PULL

KEEPER
PULLDOWN
PULLUP

SLEW RATE

SLOW
FAST

DIFF TERM

TRUE
FALSE

DISABLE_GTS

Figure 2.6: Virtex-4 I/O Buffer Architecture [5]

Virtex-4 FPGAs support a total of 69 I/O standards that define various DC interface parameters. The I/O buffers in every I/O bank of a Virtex-4 FPGA support all 3.3V I/O standards. However, many I/O standards have technical restrictions that limit the general use of the I/O buffers. One restriction is that nine I/O standards do not support bidirectional mode of operation in the I/O buffer, including LVDS_25, LVDSEXT_25, ULVDS_25, LDT_25, RSDS_25, SSTL2_I_DCI, SSTL18_I_DCI, LVDS_25_DCI, and LVDSEXT_25_DCI [5].

There are four main types of Virtex-4 supported I/O standards, represented in Figure 2.7. These four types include single-ended, single-ended requiring a voltage reference (V_{REF}), complementary differential, and digitally controlled impedance (DCI) I/O standards. The two grey dotted lines running the length of Figure 2.7 separate the source and destination I/O buffers used in each type of standard. Figure 2.7(a) shows the connections of a single-ended standard, in which a single wire connects the output buffer of one device to the input buffer of another.

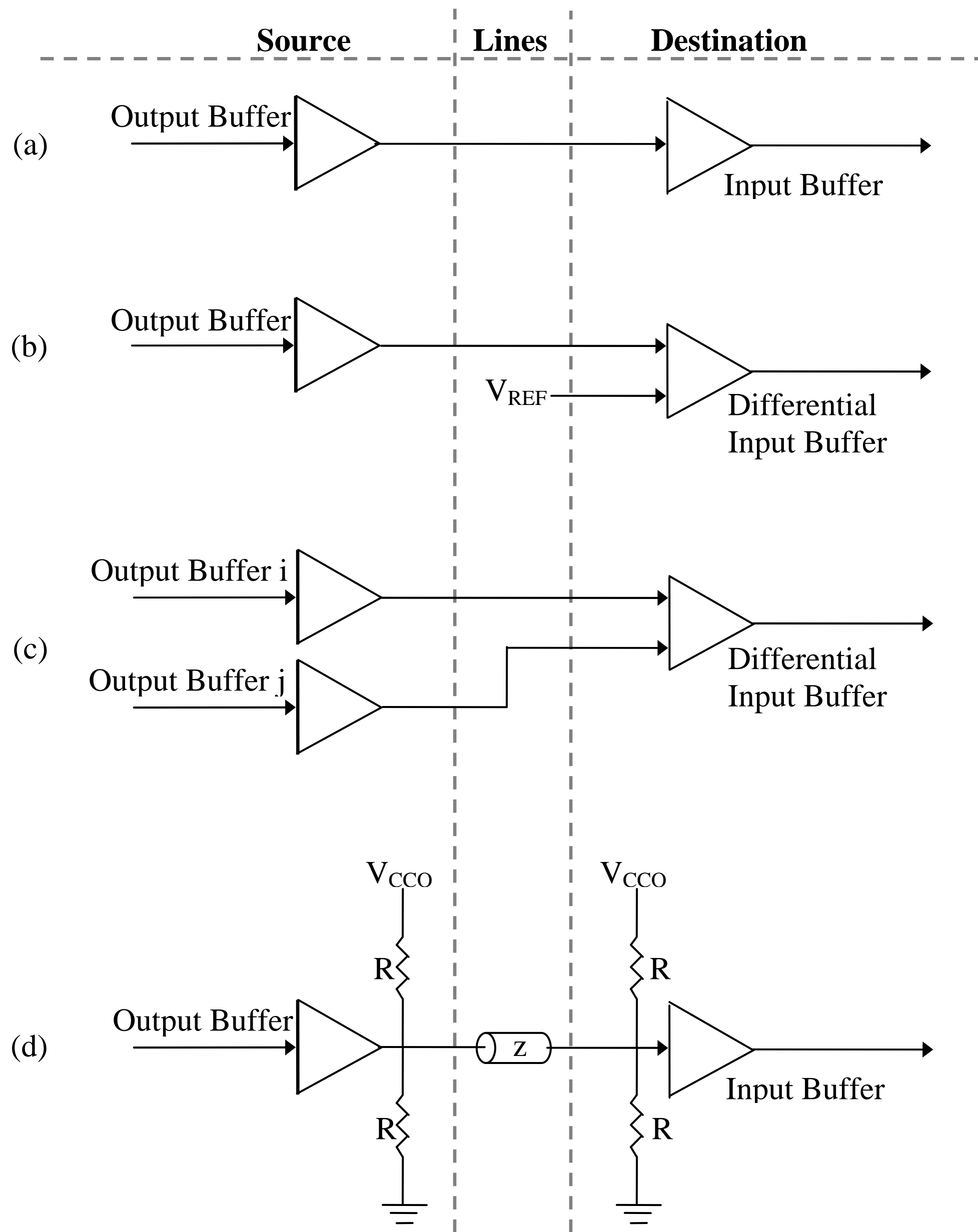


Figure 2.7: Types of Virtex-4 Supported I/O Standards. (a) Single-Ended. (b) Single-Ended Requiring a V_{REF}. (c) Complementary Differential. (d) DCI

Single-ended I/O standards, such as HSTL and SSTL standards, require a reference voltage to be applied to the I/O buffer's differential input buffer, as seen in Figure 2.7(b). When using such a standard, one of every 16 general I/O buffers in an I/O bank must be configured as a V_{REF} input. The specific I/O buffer in every set of 16 that

must be configured as a V_{REF} input for each Virtex-4 device and package is located in the bottom I/O cell of the fifth I/O tile row from the bottom edge in each I/O bank, as indicated in [24]. This V_{REF} input is routed internally to the V_{REF} pin of each differential input buffer in the set of 16 I/O buffers to which the V_{REF} input is assigned [5]. If any one I/O buffer in a set of 16 is configured with a standard requiring a V_{REF}, then the V_{REF} pin associated with that set cannot be configured by the user. The designer must supply the required V_{REF} from an external source directly to each V_{REF} input.

Complementary differential I/O standards require two output buffers at the source, as seen in Figure 2.7(c). Output buffers *i* and *j* send complemented data to an input buffer at the destination side. In Virtex-4 devices, complementary differential I/O standards require the use of both I/O cells in an I/O tile. Figure 2.8 demonstrates an I/O tile operating as a system input with a complementary differential standard. A data line is supplied to the positive terminal of the input buffer and its complement (shown as a grey dotted line) is applied to the negative input of the buffer. A master I/O cell contains the differential input buffer. A slave I/O cell in the same I/O tile sends data to the differential input buffer of the master I/O cell via its PADOUT line. In this configuration, logic high on the positive terminal and logic low on the negative terminal will produce logic high output, while the opposite arrangement of values will produce logic low output. If the I/O tile were configured as an output operating with a complementary differential I/O standard, an OLOGIC can be used to complement the data line. The data line would be sent out via one output buffer while its complement would be sent out via the other output buffer in the I/O tile.

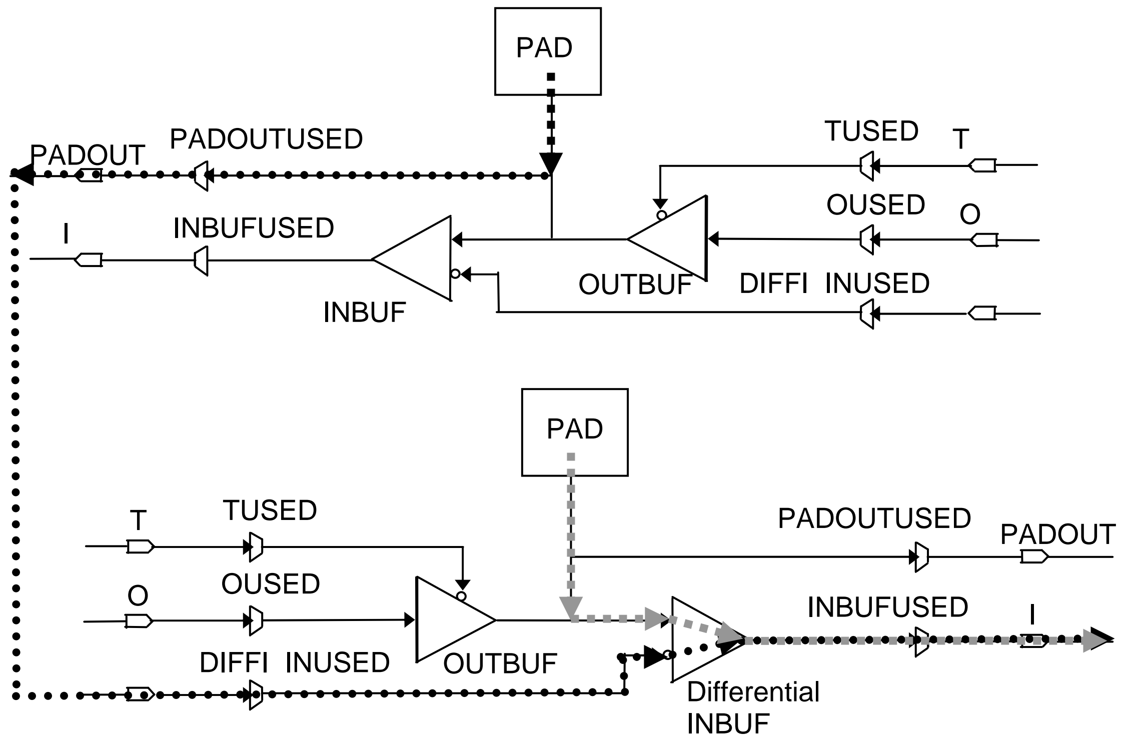


Figure 2.8: Input I/O Tile with Complementary Differential I/O Standard

Virtex-4 devices provide a DCI option for most I/O standards which facilitates line termination, as seen in Figure 2.7(d). The characteristic impedance of the driven line in Figure 2.7(d) is represented by the Z element. Some I/O standards require line termination to V_{CCO} with resistance equal to the characteristic impedance of the driven line. Still other I/O standards require line termination to $V_{CCO}/2$ with resistance equal to the twice the characteristic impedance of the driven line. To match the line impedance, DCI can provide both single and split termination resistors at the source, destination, or both the source and destination [5].

The termination resistance used to control the DCI line impedance is set by adding external reference resistors to two multipurpose reference pins in each I/O bank configured with a DCI I/O standard. The two DCI reference pins in each I/O bank

consist of V_{RN} and V_{RP} pins that must be pulled to V_{CCO} and ground, respectively, by external reference resistors [5]. When using a DCI I/O standard with any one I/O buffer in an I/O bank, the user must not configure the two DCI reference pins of the given I/O bank so that they can be connected to external reference resistors. Also, Banks one and two of a Virtex-4 device do not support DCI standards. The V_{RN} and V_{RP} DCI reference I/O buffers are located in the tenth row from the bottom edge of each I/O bank, with the exception of banks in the center column, as specified in [24]. The V_{RN} and V_{RP} DCI reference I/O buffers of I/O banks in the center column are located in the second row from the bottom edge and second row from the top edge. I/O banks one and two located in the center column above and below the center of every Virtex-4 package do not support DCI I/O standards.

Table 2.3 lists the general purpose I/O buffers for bank seven of a Virtex-4 with package FF668. The table demonstrates the location of the DCI reference resistor pins within the bank. Table 2.3 also lists the location of the four V_{REF} pins and two IDELAYCTRL modules in the bank. The IDELAYCTRL modules are listed between rows as they are placed on the device. The use of these pins and the testability of the various I/O standards will be discussed in greater detail in subsequent chapters.

Table 2.3: General Purpose I/O Bank 7 for Virtex-4 Package FF668

Row	**Component**	**Other Function**	**Row**	**Component**	**Other Function**
0	I/O Buffer AD21		16	I/O Buffer AC24	
0	I/O Buffer AE21		16	I/O Buffer AC23	
1	I/O Buffer AE18		17	I/O Buffer AD23	
1	I/O Buffer AF18		17	I/O Buffer AD22	
2	I/O Buffer AF22		18	I/O Buffer AA23	
2	I/O Buffer AF21		18	I/O Buffer AB23	
3	I/O Buffer AB18		19	I/O Buffer AB22	
3	I/O Buffer AC18		19	I/O Buffer AC22	
4	I/O Buffer AC20	V_{REF}	20	I/O Buffer Y23	V_{REF}
4	I/O Buffer AB20		20	I/O Buffer Y22	
5	I/O Buffer AA17		21	I/O Buffer AD26	
5	I/O Buffer Y17		21	I/O Buffer AD25	
6	I/O Buffer AA20		22	I/O Buffer AA26	
6	I/O Buffer AA19		22	I/O Buffer AB26	
7	I/O Buffer AC19		23	I/O Buffer AC26	
7	I/O Buffer AD19		23	I/O Buffer AC25	
	IDELAYCTRL	IDELAY		IDELAYCTRL	IDELAY
8	I/O Buffer AB21		24	I/O Buffer Y24	
8	I/O Buffer AC21		24	I/O Buffer AA24	
9	I/O Buffer AD20	DCI VRN	25	I/O Buffer AB25	
9	I/O Buffer AE20	DCI VRP	25	I/O Buffer AB24	
10	I/O Buffer AE24		26	I/O Buffer Y26	
10	I/O Buffer AF24		26	I/O Buffer Y25	
11	I/O Buffer Y18		27	I/O Buffer V20	
11	I/O Buffer AA18		27	I/O Buffer W20	
12	I/O Buffer Y21	V_{REF}	28	I/O Buffer W24	V_{REF}
12	I/O Buffer Y20		28	I/O Buffer W23	
13	I/O Buffer AE23		29	I/O Buffer W22	
13	I/O Buffer AF23		29	I/O Buffer W21	
14	I/O Buffer W19		30	I/O Buffer W26	
14	I/O Buffer Y19		30	I/O Buffer W25	
15	I/O Buffer AF20		31	I/O Buffer V22	
15	I/O Buffer AF19		31	I/O Buffer V21	

2.2.5 Programming Interface

All Virtex-4 FPGAs contain four boundary scan (also known as JTAG) modules that conform to the IEEE 1149.1-2001 standard [19][22]. Boundary scan

implementations consist of four major signals: test clock input (TCK), test mode select (TMS), test data in (TDI), and test data out (TDO). TCK provides the test clock signal. TDI receives serial test data and instructions. TDO provides a serial output for test data and instructions [25].

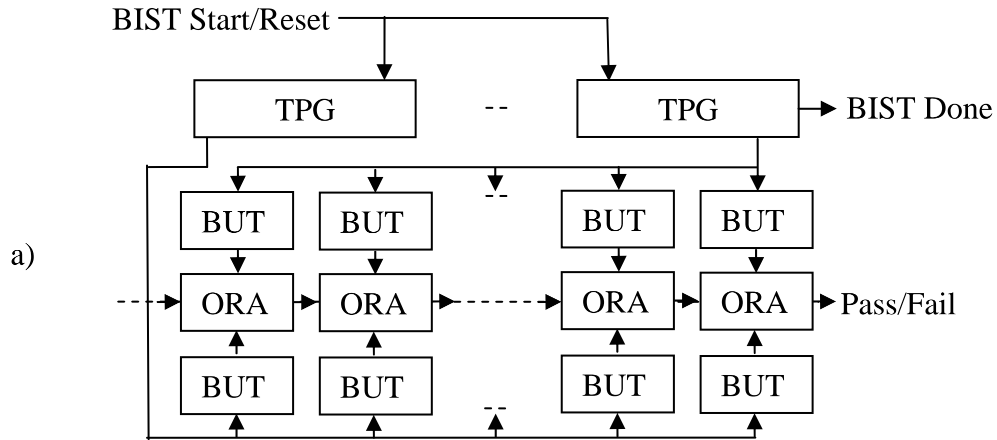
TMS signals are decoded by the test access port (TAP) controller to control test operations. The TAP controller provides access to configuration and test support functions built into the FPGA, including INTEST and EXTEST features as described in [25]. The EXTEST feature can be used to test the buffers, pad, and tristate control of the I/O buffer. However, the EXTEST feature only has access to the I/O buffer and cannot test the remaining logic of the I/O cell. The INTEST feature can be used to test the programmable logic resources in an I/O cell. However, the INTEST feature is supported by few FPGAs [9]. Virtex-4 FPGAs also allow boundary scan signals to be connected directly from the TAP to internal FPGA resources [19][22].

2.3 Built-In Self-Test for FPGAs

The goal of a BIST developed for FPGAs is to test the FPGA components through a series of test configurations. A test procedure for FPGA BIST typically consists of: 1) configure the FPGA with the BIST test circuitry including TPGs, blocks under test (BUTs), and ORAs; 2) initiate the BIST test circuitry; 3) the BIST circuitry generates test patterns with TPGs and sources them to the components under test; 4) generate a comparison based pass/fail indication with BIST circuitry ORAs; and 5) read a pass/fail indication from BIST circuitry ORAs [26]. FPGAs are configured repeatedly in this manner with different BIST configurations stored in the external memory to test

various resources and functionality of the device under test [8]. A group of BIST configurations that test a given component of the FPGA are sometimes collectively referred to as a test session [26][27].

BIST has proven to be an effective technique for testing most major components of a FPGA. For example, [27] presents a general BIST architecture aimed at testing PLBs. The basic BIST architecture consists of alternating rows of PLBs that are configured as BUTs or ORAs, as seen in Figure 2.9. Two identical TPGs are used to drive alternating rows or columns of logic blocks under test to ensure TPG functionality and test diagnostic resolution, as seen in Figure 2.9 (a). The outputs of the BUTs are then compared by PLBs configured as ORAs. Because exactly half of the PLBs are configured as BUTs, only two test sessions are required for the entire FPGA. The floorplans of the two test sessions required are illustrated in Figure 2.9 (b) and (c). When the first test session has completed, the PLBs that are BUTs in session one are reconfigured to be the TPGs and ORAs for session two [27].



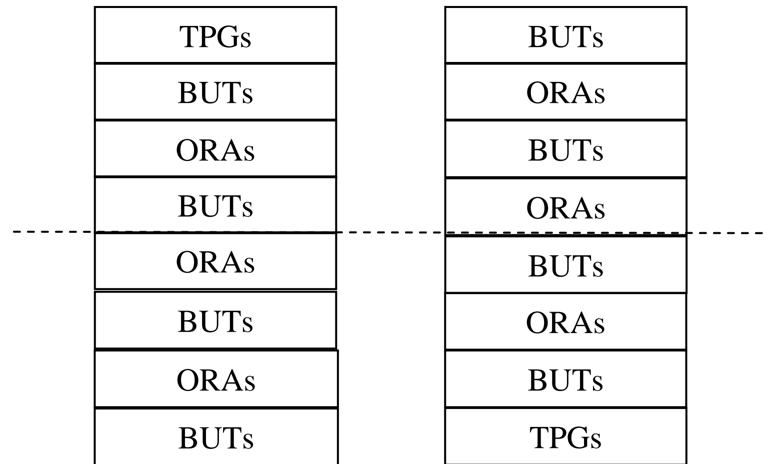


Figure 2.9: BIST Architecture presented in [27]. (a) TPG, BUT, ORA connections. (b) Floorplan for first test session. (c) Floorplan for second test session.

BIST has also proven effective for various FPGA cores. For example, [19] presents a BIST approach for testing BRAMs in Virtex-4 devices. The basic BRAM BIST architecture consists of CLBs configured as TPGs that source test patterns to BRAMs under test, as seen in Figure 2.10. Again, two identical TPGs are used to drive alternating rows of BRAMs under test to ensure TPG functionality. The outputs of the BRAMs under test are then circularly compared by CLBs configured as ORAs. The

ORAs form a circular based comparison to enhance diagnostic resolution of a particular failing BUT or TPG. As indicated by the arrows of Figure 2.10, the topmost BRAMs are compared with the bottommost BRAMs to complete the circular comparison chain [19].

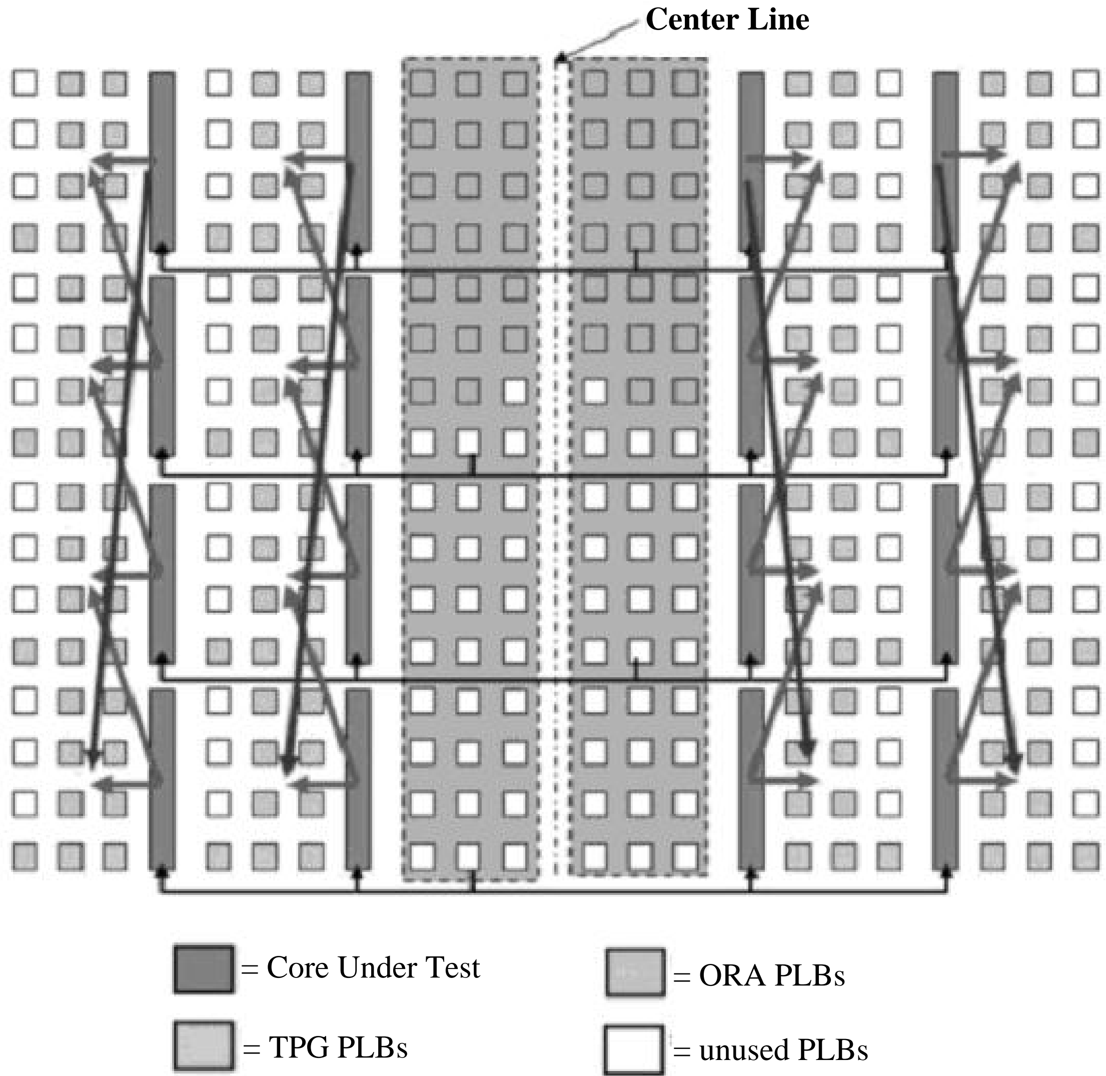


Figure 2.10: BRAM BIST Architecture Presented in [19]

BIST for FPGAs, unlike BIST for most other circuitry, can be implemented such as not to incur system performance or area overhead penalties, meaning system execution time remains high and DFT costs remain low. Using only additional memory (external to

the FPGA) to store BIST configurations, there is no additional circuitry permanently added into the FPGA design itself. When the BIST test session is complete, the BIST circuitry is removed from the FPGA [8].

2.4 Previous Work in I/O Testing

This thesis is primarily built upon the BIST work presented in [8][9][14][15]. In [8][14][15], a system level BIST architecture similar to those described for CLBs and BRAMs is presented for I/O cells of Atmel FPGAs. The BIST architecture, as seen in Figure 2.11, consists of a single TPG implemented in PLBs sourcing test vectors to I/O cells under test. Only a single TPG was implemented under the assumption that internal FPGA resources have already been tested. The architecture presented in [9] added multiple identically configured TPGs that drive alternating I/O cells under test for improved fault detection as will be discussed in Chapter 3.

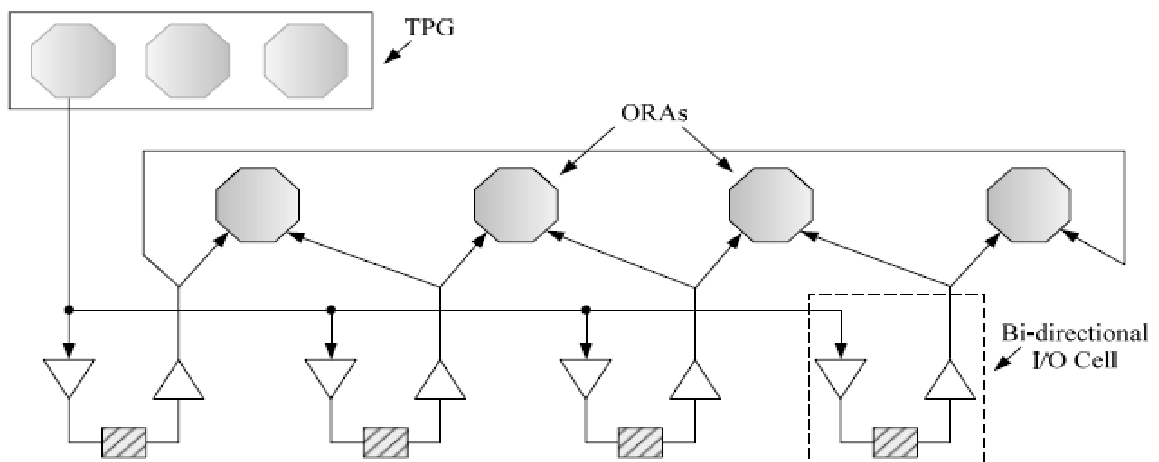


Figure 2.11: BIST Architecture from [8], also representative of [14][15]

The I/O cells are identically configured with bidirectional I/O buffers so that the output responses are sent back into the FPGA internal resources. The output responses of the I/O cells are then compared by PLBs configured as ORAs. Figure 2.12 shows the

ORA architecture used in [8][9][14][15]. The ORAs are configured such that the outputs of every I/O cell are compared with the outputs of their adjacent I/O cells. This configuration is used to implement circular comparison for enhanced diagnostic resolution. The ORA design contains shift data and shift mode lines to support connecting the ORAs as a scan chain or shift register similar to that of boundary scan. The shift lines can be used to shift test results out of the ORAs when the FPGA does not support CLB configuration memory read back. The ORA design also contains feedback from the storage element flip-flop such that failure indications of logic high value due to mismatches in output responses as a result of faults are retained until the BIST results are read from the device [8][9][14][15].

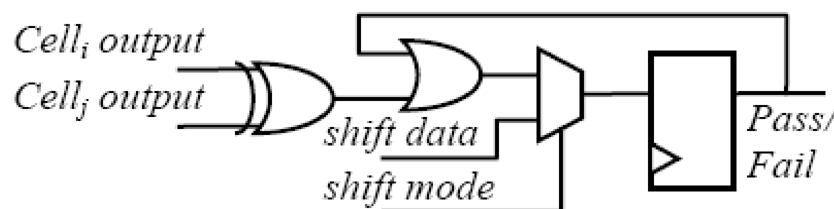


Figure 2.12: Comparison Based ORA Architecture [9]

While presenting a general architecture applicable to any FPGA or SoC with a FPGA core, [8][9][14][15] only develop BIST configurations for the Atmel AT94K SoC and AT40K FPGA. The AT94K and AT40K contain primary and secondary I/O buffers. An Atmel primary I/O buffer, as seen in Figure 2.13, consists of pull-up and pull-down transistors, input and output buffers, and input and output multiplexers. The input buffer has four programmable delay settings. The input buffer can also implement a Schmitt trigger circuit that helps filter input noise. Another feature of the input buffer is the ability to support both TTL and CMOS voltage levels. The output buffer has variable drive strength and tristate control. Both the input and output portions of the I/O buffer

share access to the routing interconnect resources through the use of transmission gates [8][14].

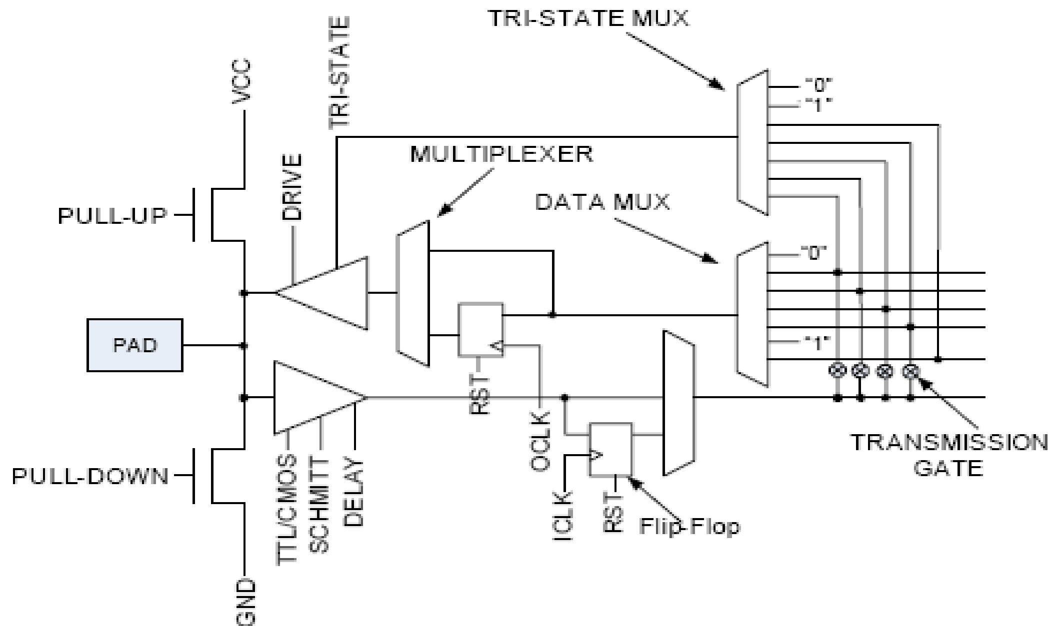


Figure 2.13: Primary I/O Buffer in Atmel FPGA [14]

Secondary I/O buffers are similar in architecture to primary I/O buffers. One difference between the two types of I/O buffers is the secondary I/O buffer has 6-input tristate and output multiplexers whereas the primary I/O buffer has 7-input multiplexers. Another difference is that the primary IO buffer has four transmission gates whereas the secondary I/O buffer only has two. One final difference is that the primary and secondary I/O buffers have different connections to the internal PLBs. Because of the similarity, secondary I/O buffers in Atmel devices can be thought of as a subset of primary I/O buffers [8][14].

References [8][9][14][15] present 23 BIST configurations to obtain 100% stuck-at gate-level fault coverage. This indicates that all gate-level faults located in the logic and routing resources of the primary and secondary I/O buffers can be detected. The BIST

approaches presented can also detect major defects affecting the analog programmable features of the I/O buffers, such as pull-up, pull-down, and tristate termination. However, the BIST approaches presented here cannot detect parametric faults such as delay or current sink and source capabilities. Overall, these BIST approaches provide the ability to perform tests at different frequencies and produce more reliable results than the conventionally accepted boundary scan testing [8][9][14][15].

2.5 Thesis Restatement

This thesis presents a general BIST architecture developed to test the functionality of the various resources present in I/O tiles of FPGAs as well as specific configurations developed and implemented in Xilinx Virtex-4 devices. This thesis builds on the work presented in [8][9][14][15]. However, this thesis offers several improvements over the previously discussed BIST approaches including the use of multiple TPGs for improved fault detection. The TPGs are also implemented in BRAM memory for improved controllability.

The thesis describes BIST configurations developed for testing the full functionality of Virtex-4 I/O tiles that use two I/O cells in conjunction with each other. This thesis first presents BIST configurations to test I/O tile logic resources, including ILOGIC, OLOGIC, SERDES, and I/O buffer resources. Some new modes of operation are addressed, such as I/O tiles operating in master/slave ISERDES/OSERDES mode. The thesis then discusses BIST configurations developed to test Virtex-4 supported I/O standards. This thesis explores the testability of all supported I/O standards, including single-ended, single-ended requiring a reference voltage, complementary differential, and

DCI standards. A detailed description of the I/O tile BIST configurations developed will be given as well as experimental results obtained from their implementation.

CHAPTER THREE

I/O TILE BIST FOR VIRTEX-4 FPGAS

This chapter describes the BIST configurations developed to test the logic and routing resources of I/O tiles in Virtex-4 FPGAs. The chapter begins with a discussion of the BIST architecture that is common to all of the configurations developed, including TPGs and ORAs. Section 3.2 describes the BIST configurations developed to test logic modules present in I/O tiles including ILOGIC, OLOGIC, SERDES, and I/O buffer resources. Section 3.3 then describes BIST configurations developed to test the various I/O standards supported in Virtex-4 devices. Finally, section 3.4 provides a summary of the Virtex-4 I/O tile BIST configurations.

3.1 BIST Architecture

The basic BIST architecture used in all I/O tile BIST configurations is shown in Figure 3.1. In this architecture, TPGs source test vectors to I/O tiles under test, which are configured with bidirectional I/O buffers. The BIST architecture is similar to those presented in [8][14][15]. However, in this architecture a second set of TPGs is added. Multiple sets of TPGs drive alternating I/O tiles under test. This TPG configuration avoids the assumption in previous I/O BIST approaches that internal FPGA resources are already known to be fault free. The concern in a single TPG architecture is that a faulty TPG may skip test patterns necessary to detect faults in the I/O tiles such that they escape

detection. If a TPG is faulty, every I/O tile that it drives will produce identical failures. This will occur because each I/O tile response produced by a faulty TPG is compared with an I/O tile response produced by an alternate TPG. This assumes that the alternate TPG to that of the faulty TPG does not have an identical fault, the likeliness of which is very small.

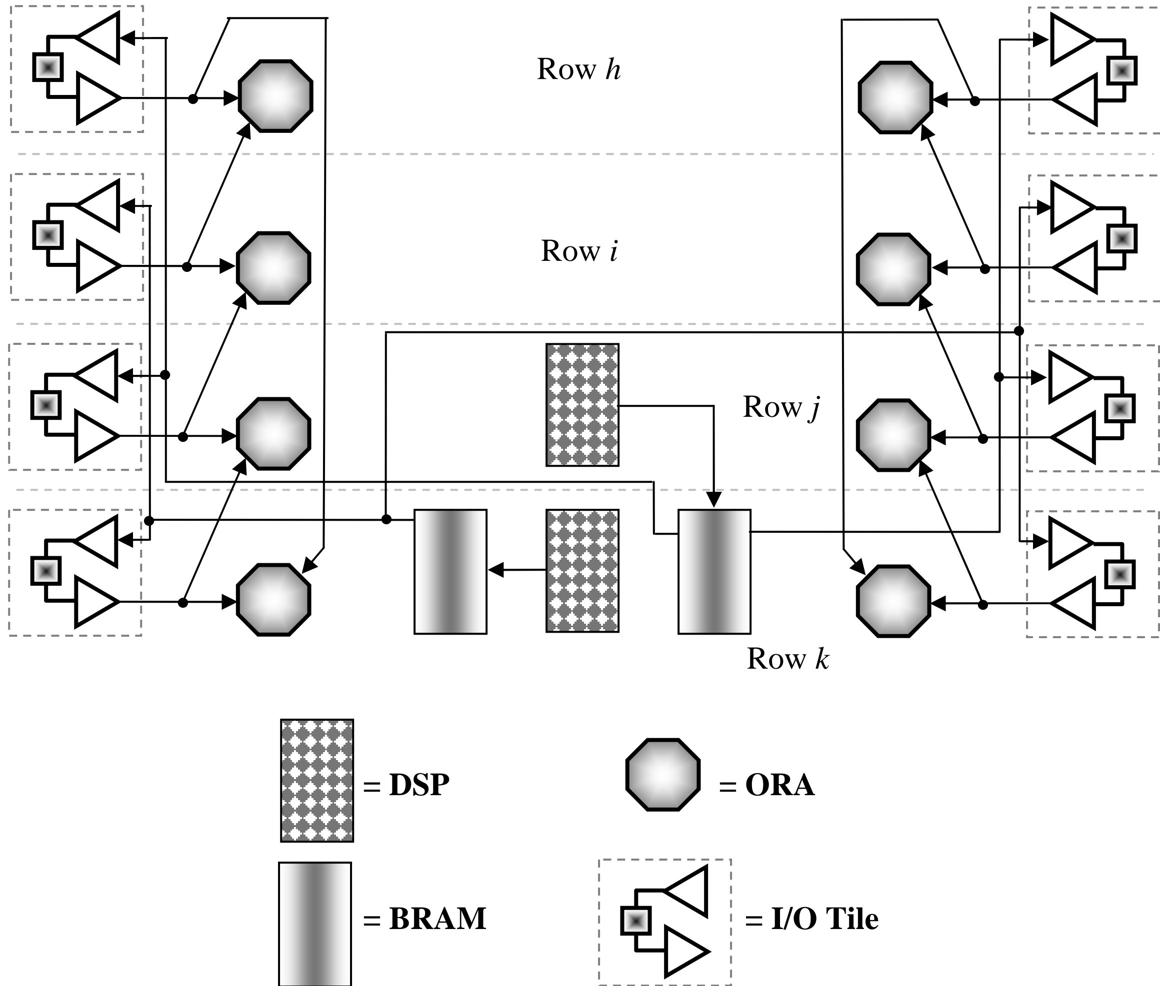


Figure 3.1: Basic I/O Tile BIST Architecture

The TPGs are implemented as one column of DSPs addressing two columns of BRAMs that store test vectors, as seen in Figure 3.1. There are two TPGs placed every four rows of CLBs or I/O tiles in a Virtex-4 device. The use of two TPGs per four rows

of I/O tiles reduces signal loading in FPGAs with large numbers of rows. The two TPGs drive alternating rows of I/O tiles in both the leftmost, middle, and rightmost columns of a device. The BRAMs are either implemented with an aspect ratio of 16Kx1 or 512x36, depending on the BIST configuration developed. The use of BRAMs to store test vectors allows for modification of the test vector set without having to modify the existing BIST architecture. The output data lines of the BRAM TPGs drive both ILOGIC and OLOGIC components of an I/O tile.

In each configuration, there is a single column containing one DSP for each BRAM. Each DSP is configured as a counter such that its outputs, P0-P9, are connected to the D0B0-D0B9 address inputs of a BRAM TPG. A DSP counter increments its count once per clock cycle. The clock supplied to a DSP is the BIST clock generated from either a boundary scan clock or DCM divided clock, depending on the set of BIST configurations. The remaining DSP signals and attributes are summarized below in Table 3.1.

Table 3.1: DSP Signals and Attributes for All Configurations

Signal/Attribute	I/O/Attribute	Connection	Function
A0-A17	Input	GLOBAL 0	Multiplier A inputs
B0	Input	GLOBAL 1	Multiplier B input
B1-b17	Input	GLOBAL 0	Multiplier B inputs
CARRYIN	Input	GLOBAL 0	Carry Input
CARRYINSEL0	Input	GLOBAL 0	Select Carry Source
CARRYINSEL1	Input	GLOBAL 0	Select Carry Source
CEA	Input	GLOBAL 1	Hold AREG
CEB	Input	GLOBAL 1	Hold BREG
CECARRYIN	Input	GLOBAL 0	Hold Clock Enable
CECINSUB	Input	GLOBAL 1	Hold SUBTRACTREG
CECTRL	Input	GLOBAL 1	Hold CARRYINSELREG
CEM	Input	GLOBAL 1	Hold MREG
CEP	Input	GLOBAL 1	Hold PREG
Clock	Input	BIST clock	Clock
OPMODE0-6	Input	1100010	X, Y, Z multiplexers
RSTA	Input	GLOBAL 0	Reset AREG
RSTB	Input	GLOBAL 0	Reset BREG
RSTCARRYIN	Input	GLOBAL 0	Reset CARRYIN
RSTCTRL	Input	GLOBAL 0	Reset SUBTRACTREG
RSTM	Input	GLOBAL 0	Reset MREG
RSTP	Input	GLOBAL 0	Reset PREG
SUBTRACT	Input	GLOBAL 0	ADD/SUBTRACT
P0-P9	Output	D0B0-D0B9	Product Outputs
MREG	Attribute	GLOBAL 1	Number of Pipeline Reg
PREG	Attribute	GLOBAL 1	Number of Pipeline Reg
Remaining Atts.	Attribute	GLOBAL 0	Various

The ORAs are shown in Figure 3.2 where each output signal of every I/O tile under test is sent to an ORA comparator in its associated row as well as the row directly above it. The topmost I/O tile in a column under test sends its output responses to ORA comparators in its associated row as well as the bottommost row under test. This effectively forms a circular comparison chain where every output signal of an I/O tile under test is compared with the identical output signals of two other I/O tiles. Any mismatch places a logic high value into the ORA flip-flop, where it is then retained by

the ORA feedback line until the pass/fail result is read at the end of the BIST sequence. The logic functions indicated are performed by a LUT in each slice. Each slice contains two LUTs, and thus can be used to implement two ORAs. The number of ORAs implemented depends on the set of BIST configurations. When testing I/O tiles in the leftmost and center column of a device, the ORAs are implemented in CLBs beginning one column to the right of the I/O tiles under test, as seen in Figure 3.1. When testing I/O tiles in the rightmost column of a device, the ORAs are implemented in CLBs beginning one column to the left of the I/O tiles under test.

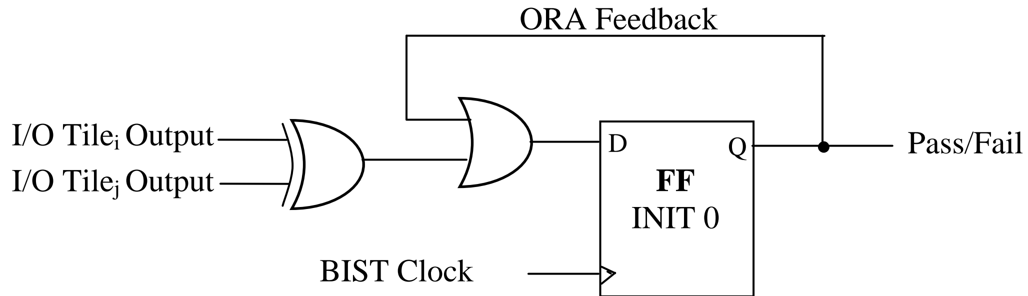


Figure 3.2: I/O Tile BIST ORA Architecture

Each set of BIST configurations instantiates at least one boundary scan module for the BIST sequence execution. The boundary scan clock signal is sent to a clock buffer so that it can then be used as a global BIST clock in BIST configurations other than the SERDES configuration set. The boundary scan TDI signal is used by the BIST configurations as a reset signal to various components, such as IDELAYCTRL modules.

Each Virtex-4 I/O tile BIST configuration instantiates all of the IDELAYCTRL modules of the Virtex-4 device under test. As stated in the previous chapter, the IDELAYCTRL elements continuously calibrate the individual IDELAY modules in each I/O bank and must be instantiated when using the IDELAY element in FIXED or

VARIABLE delay mode. The IDELAYCTRL modules are supplied a reference clock, used for calibration, and a reset line from the boundary scan module TDI output. The IDELAYCTRL modules are reset at the beginning of each BIST configuration, as recommended in [5]. The output of each IDELAYCTRL module indicates that the module has been reset. The I/O tile BIST configurations developed in this thesis leave each IDELAYCTRL module output unconnected, as it is not necessary for a user design.

3.2 Configurations to Test I/O Tile Logic Resources

The BIST configurations presented in this thesis can be divided into three main categories to test logic resources and I/O standards. Section 3.2.1 describes the BIST configurations developed to test logic resources present in the ILOGIC, OLOGIC, and I/O buffer components. Section 3.2.2 then describes BIST configurations developed to test the logic resources of ILOGIC and OLOGIC components when operating in SERDES mode, as well as remaining I/O buffer logic resources. The BIST configurations developed to test the various I/O standards supported by the I/O buffers are discussed in Section 3.3.

3.2.1 Configurations to Test ILOGIC, OLOGIC, and I/O Buffers

The architectures of ILOGIC, OLOGIC, and I/O buffers in Virtex-4 FPGAs were described in Section 2.2. Eight BIST configurations are required to test the ILOGIC and OLOGIC routing and logic resources. This set of eight BIST configurations is also used to test various I/O buffer resources. Tables 3.2, 3.3, and 3.4 in this section list what each of eight BIST configurations in this set test in terms of ILOGIC, OLOGIC, and I/O buffer logic resources.

Table 3.2 shows the various configuration modes of ILOGIC components tested by each of the eight BIST configurations. Configurations one, two, and three test the majority of resources in the ILOGIC registered data path. They test the synchronous functionality of all four flip-flops in the registered data path including INIT and SRVAL attributes. They also test the SR, REV, CE1, and CLK inverters and functionality. Finally, they test the IFF3 and IFF4 selection of the Q1 and Q2 output multiplexers, respectively. The next five configurations test the remainder of the ILOGIC resources using only the top two flip-flops of the registered data path, thus also testing the IFF1 and IFF2 selection of the Q1 and Q2 output multiplexers, respectively. Configuration six tests the asynchronous latch capability of the IFF1 flip-flip. Configurations seven and eight test the full functionality of the IDELAY module. Configuration seven uses the IDELAY module with a fixed delay. Configuration eight sets the IDELAY module in a variable delay mode, which also allows the TPG to increment and decrement the delay module values to test every delay value.

Table 3.2: Configuration Modes of ILOGIC Components

Resources	Cfg. 1	Cfg. 2	Cfg. 3	Cfg. 4	Cfg. 5	Cfg. 6	Cfg. 7	Cfg. 8
DELAYMUX	1	0	0	0	1	1	0	0
DELMUX	0	1	1	0	1	0	0	0
D2OBYPSEL	T	T	GND	GND	GND	GND	GND	GND
IMUX	0	1	0	1	1	1	1	1
IFFDELMUX	1	0	1	1	0	0	1	0
D2OFFBSEL	T	GND	T	GND	T	GND	GND	GND
IFFMUX	1	0	1	0	0	1	0	1
CE1INV	CE1_B	CE1	CE1_B	CE1_B	CE1_B	CE1	CE1_B	CE1_B
CLKINV	CLK	CLK	CLK_B	CLK_B	CLK_B	CLK	CLK_B	CLK_B
SRINV	SR	SR	SR_B	SR_B	SR_B	SR	SR_B	SR_B
REVINV	REV	REV	REV_B	REV_B	REV_B	REV	REV_B	REV_B
Q1MUX	IFF3	IFF3	IFF3	IFF1	IFF1	IFF1	IFF1	IFF1
Q2MUX	IFF4	IFF4	IFF4	IFF2	IFF2	IFF2	IFF2	IFF2
IFF1	FF	FF	FF	FF	FF	Latch	FF	FF
SRVAL[1:4]	0000	1111	1111	1111	0000	1111	1111	1111
INIT[1:4]	0011	1111	0000	1111	1111	1111	1111	1111
SRTYPE	Sync	Sync	Sync	Sync	Sync	Async	Sync	Sync
CLKDIVINV	DIV_B	DIV_B	DIV_B	DIV_B	DIV_B	DIV_B	DIV_B	DIV
DELAYVAL.	0	0	0	0	0	0	2	63
DELAY TYPE	Default	Default	Default	Default	Default	Default	Fixed	Var.

The eight configurations used to test ILOGIC components are also used to test OLOGIC resources. At least four BIST configurations are required to test the four possible values of the TMUX and OMUX output multiplexers. Table 3.3 below shows the various configuration modes of OLOGIC components tested by the eight BIST configurations. Configurations four and five test the asynchronous latch functionality of the topmost flip-flops in both the tristate and output data paths. Configurations three, four, and eight also test the combinatorial paths of the tristate and output data paths, thus testing the T1 and D1 selection of the TMUX and OMUX multiplexers, respectively. As seen in Table 3.3, the remaining BIST configurations in this set test the various other OLOGIC resources and output multiplexer selections.

Table 3.3: Configuration Modes of OLOGIC Components

Resources	Cfg. 1	Cfg. 2	Cfg. 3	Cfg. 4	Cfg. 5	Cfg. 6	Cfg. 7	Cfg. 8
T1INV	T1_B	T1	T1_B	T1_B	T1_B	T1	T1	T1_B
T2INV	T2_B	T2	T2_B	T2_B	T2_B	T2	T2	T2_B
TCEINV	TCE_B	TCE_B	TCE_B	TCE_B	TCE	TCE	TCE_B	TCE_B
CLK1INV	C	C_B	C_B	C	C_B	C_B	C_B	C
CLK2INV	CLK_B	CLK_B	CLK_B	CLK	CLK	CLK	CLK_B	CLK
D1INV	D1	D1_B	D1_B	D1_B	D1_B	D1	D1_B	D1_B
D2INV	D2	D2_B	D2	D2_B	D2_B	D2	D2_B	D2_B
OCEINV	OCE_B	OCE_B	OCE	OCE_B	OCE_B	OCE	OCE_B	OCE_B
SRINV	SR	SR_B	SR	SR_B	SR	SR_B	SR	SR_B
REVINV	REV	REV_B	REV	REV_B	REV_B	REV_B	REV	REV_B
TFF1	FF	FF	FF	Latch	FF	FF	FF	FF
OFF1	FF	FF	FF	FF	Latch	FF	FF	FF
TMUX	DRB	DRA	DRB	TFF1	TFF1	DRA	TFF1	T1
OMUX	DRB	DRA	D1	D1	OFF1	OFF1	DRB	D1
SRVALOQ	1	1	1	1	0	1	0	1
SRVALTQ	1	1	0	0	1	0	0	0
INITOQ	1	0	1	1	0	1	0	1
INITTQ	1	0	0	0	0	1	0	0
SRTYPEOQ	Sync	Sync	Sync	Async	Sync	Sync	Sync	Sync
SRTYPETQ	Sync	Sync	Sync	Sync	Async	Sync	Sync	Sync

The I/O buffer testing is limited by the bidirectional configuration the BIST architecture requires. However, several of the I/O buffer logic resources can be tested with this set of eight BIST configurations, as seen in Table 3.4. The I/O buffers are configured to operate in each of the slew and pull modes. Five of the eight drive strengths are tested: 24, 4, 6, 2, 8. The configurations that use PCI I/O standards do not support testing drive strengths. The remaining drive strengths are tested in BIST configurations developed to test SERDES logic resources. This set of BIST configurations also tests eight different I/O standards, including LVTLL, LVCMOS, and PCI standards. The remainder of the I/O buffer attributes and standards are tested with BIST configurations discussed in the next two sections.

Table 3.4: Configuration Modes of I/O Buffer Components

Resources	**Cfg. 1**	**Cfg. 2**	**Cfg. 3**	**Cfg. 4**	**Cfg. 5**	**Cfg. 6**	**Cfg. 7**	**Cfg. 8**
PULL	Keep.	Down	Pullup	Keep.	Down	Pullup	Keep.	Keep.
GTSATTR	Off	Disable	Disable	Off	Disable	Disable	Off	Off
SLEW	Slow	Fast	Slow	Off	Slow	Off	Fast	Slow
DIFFTERM	NA	NA	NA	NA	NA	NA	NA	NA
DRIVE	24	4	6	Off	2	Off	8	Off
DRIVE0MA	Off	Drive0	Drive0	Off	Drive0	Drive0	Off	Off
IOATTR	LVTTL	LVCM-OS33	LVCM-OS25	PCI66-3	LVCM-OS15	PCI33-3	LVCM-OS18	PCI-X

The architecture of the TPG BRAMs used to test ILOGIC, OLOGIC, and I/O buffer logic and routing resources is summarized in Table 3.5. Address lines ADDRA4-13 are incremented from a DSP counter once per clock cycle to set the next two bytes of BRAM data (test vectors) on the BRAM output bus. The first two test vectors stored in the BRAM are constant logic highs used to test the INIT values. The remaining BRAM test vector contents were generated using a twelve bit linear feedback shift register to provide pseudo-random patterns on the TPG outputs. The linear feedback register used was designed with external feedback and a primitive polynomial of $P(x) = x^{12}+x^6+x^4+x^3+1$ to produce a maximal length pseudo-random sequence [2]. Because there are only ten address lines for each BRAM, only 1024 of the 4096 possible test vectors are used. A program was developed, *Vec2RAM*, to convert the output values of the linear feedback shift register to hexadecimal BRAM contents.

Table 3.5: BRAM Signals and Attributes for ILOGIC, OLOGIC, and I/O Buffer Testing

Signal/Attribute	**I/O/Attribute**	**Connection**	**Function**
ADDRA4-13	Input	D0B0-D0B9	Addressing BRAM contents
CLKA	Input	BIST clock	BRAM clock
ENA	Input	GLOBAL 1	Enable output bus
REGCEA	Input	GLOBAL 0	Enable output register
SSRA	Input	GLOBAL 0	Set/Reset
WEA0	Input	GLOBAL 0	Write Enable 0
WEA1	Input	GLOBAL 0	Write Enable 0
WEA2	Input	GLOBAL 0	Write Enable 0
WEA3	Input	GLOBAL 0	Write Enable 0
D0A0-D0A11	Output	T0B0-T0B11	Data Output Bus
CLKINV	Attribute	CLKA	CLKA Inverter
EXTENSION_A	Attribute	NONE	Extended Mode
DOA_REG	Attribute	0	Pipeline Register
INV_CLK_DOA	Attribute	FALSE	Invert Pipeline Clock
Read width	Attribute	18	Output Bus width <15:0>

Each TPG line from the BRAM output bus connects to one of the twelve inputs of ILOGICs or OLOGICs in an I/O tile. Test vectors are then applied to each of the twelve connected inputs in an I/O tile. Each of the three ILOGIC outputs is then sent to ORAs for comparison, as seen in Figure 3.3. Because three outputs are being compared, this BIST configuration set requires six ORAs per I/O tile or row under test. The six ORAs are implemented in three slices in the CLB directly adjacent to the I/O tile under test. The first slice of each CLB used for ORAs is not instantiated and will appear as two logic high values when read from the configuration memory. All of the ORAs are contained within a single column for each column of I/O tiles under test. Table 3.6 shows the positioning and connections of the TPGs and ORAs in relation to rows of I/O tiles under test. The ORA positioning is read as 11 (slice not used) + 00 (slice one) + 00 (slice two) + 00 (slice three), where each slice has two ORAs. An “X” is used in Table 3.6 to mark

the ORA that is monitoring the given I/O tile signal. Signals from the top and bottom ILOGICs of a tile are distinguished as IOB0 and IOB1, respectively.

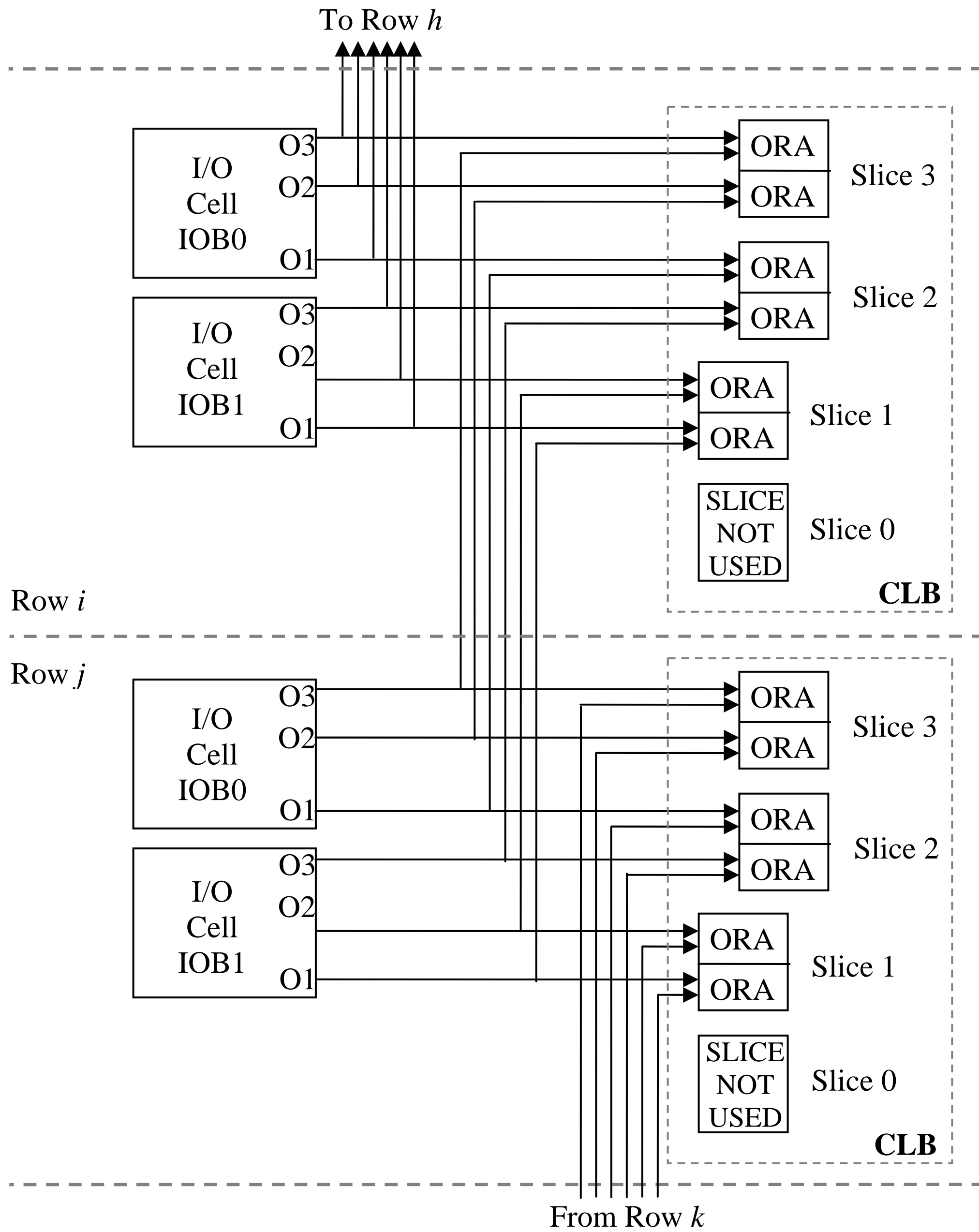


Figure 3.3: I/O Tile to ORA Connections

Table 3.6: TPG and ORA Connections for ILOGIC, OLOGIC, and I/O Buffer Testing

	Signal	From	To	ORA Positioning
To I/O Tile	T0B0	TPG D0A0	ILOGIC CE1	NA
	T0B1	TPG D0A1	ILOGIC/OLOGIC REV	NA
	T0B2	TPG D0A2	ILOGIC/OLOGIC SR	NA
	T0B3	TPG D0A3	OLOGIC D1	NA
	T0B4	TPG D0A4	OLOGIC D2	NA
	T0B5	TPG D0A5	OOGIC OCE	NA
	T0B6	TPG D0A6	OLOGIC TC	NA
	T0B7	TPG D0A7	OLOGIC T1	NA
	T0B8	TPG D0A8	OLOGIC T2	NA
	T0B9	TPG D0A9	ILOGIC DLYINC	NA
	T0B10	TPG D0A10	ILOGIC DLYRST	NA
	T0B11	TPG D0A11	ILOGIC DLYCE	NA
From I/O Tile	IOB0O	ILOGIC O	ORA S01	110X0000
	IOB0Q1	ILOGIC Q1	ORA S10	11000X00
	IOB0Q2	ILOGIC Q2	ORA S11	1100000X
	IOB1O	ILOGIC O	ORA S01	11X00000
	IOB1Q1	ILOGIC Q1	ORA S10	1100X000
	IOB1Q2	ILOGIC Q2	ORA S11	110000X0

To generate this first set of BIST configurations, template and modification programs were created. The template program, *V4iobist*, generates an XDL file that establishes the entire BIST architecture, including the instantiation and placement of the TPGs, ORAs, DCMs, BSCAN, and I/O tile components as well as their interconnections. The template XDL file is then converted to an NCD file. PAR is then used to route the NCD template design before it is converted back to XDL format. A modification program, *V4iobmod*, modifies the XDL of the placed and routed template file. The template XDL file is only modified to contain one of the eight new I/O tile configurations in the first BIST configuration set; placement and routing is left unchanged. The newly modified XDL files for each of the eight BIST configurations can then be converted to

NCD files which can then be used to generate FPGA configuration memory programming files. The remainder of this thesis refers to the eight BIST configurations presented in this section as IO9 through IO16.

3.2.2 Configurations to Test SERDES

The number of configurations to test I/O tiles in a SERDES mode of operation is dominated by the various supported data widths. A BIST configuration is required to test ISERDES/OSERDES components in each of the eight supported data widths. Thus eight BIST configurations are used to test the ISERDES/OSERDES functionality.

Template and modification programs were also created to generate this set of eight BIST configurations. The template program, *V4iobistios*, generates XDL that establishes the placement and interconnection of the entire BIST architecture, including the TPGs, ORAs, DCMs, boundary scan modules, and I/O tile components. Routing is performed using PAR via the procedure discussed in the previous section. The modification program, *V4iobmodios*, modifies the XDL of the placed and routed template created with *V4iobistios*. The template configuration is modified to reflect one of the eight new I/O tile SERDES configurations.

Table 3.7 shows the various modes of ISERDES components and attributes that are tested in each of the eight BIST configurations. Note that the configuration options and attributes shown in dark grey are already tested in the first set of configurations presented in Section 3.2. However, several of them are tested again with this set of eight BIST configurations.

Table 3.7: BIST Configurations to Test ISERDES

Resources	**Cfg. 1**	**Cfg. 2**	**Cfg. 3**	**Cfg. 4**	**Cfg. 5**	**Cfg. 6**	**Cfg. 7**	**Cfg. 8**
NUMCE	1	2	2	1	1	2	2	2
DATARATE	SDR	SDR	DDR	SDR	DDR	SDR	DDR	SDR
DATAWIDTH	2	3	4	5	6	7	8	10
DELAY	NONE	IFD	IBUF	BOTH	BOTH	NONE	IFD	BOTH
DELAYVAL.	0	2	4	8	1	0	16	63
DELAYTYPE	Def.	Fixed	Fixed	Fixed	Var.	Def.	Fixed	Fixed
MODE	Master	Master	Master	Master	Master	Master	Master/ Slave	Slave/ Master
BITSLIPEN.	True	False	True	True	True	True	True	True
INTERFACE	Net.	Mem.	Mem.	Net.	Net.	Net.	Net.	Net.
INIT_Q1	0	1	0	0	0	0	1	0
INIT_Q2	0	1	0	0	0	0	1	0
INIT_Q3	0	1	1	0	0	0	1	0
INIT_Q4	0	1	1	0	0	0	1	0
SRVAL_Q1	0	1	1	1	0	0	1	1
SRVAL_Q2	0	1	1	1	0	0	1	1
SRVAL_Q3	0	1	1	1	1	0	1	1
SRVAL_Q4	0	1	1	1	1	0	1	1
CE1INV	CE1	CE1_B	CE1	CE1_B	CE1	CE1_B	CE1	CE1
CE2INV	CE2	CE2_B	CE2_B	CE2	CE2	CE2_B	CE2_B	CE2
CLKINV	CLK	CLK	CLK_B	CLK_B	CLK	CLK	CLK_B	CLK
CLKDIVINV	DIV	DIV	DIV_B	DIV_B	DIV	DIV	DIV_B	DIV
OCLKINV	CLK	CLK	CLK_B	CLK_B	CLK	CLK	CLK_B	CLK
REVINV	REV	REV	REV	REV	REV	REV	REV	REV
SRINV	SR	SR_B	SR_B	SR	SR	SR_B	SR_B	SR
SHIFTOUT1-2 SERDESi,j	Off	Off	Off	Off	Off	Off	On/ Off	Off/ On
SHIFTIN1-2 SERDESi,j	Off	Off	Off	Off	Off	Off	Off/On	On/Off

Configurations two and three use a memory interface mode in which the OCLK is tested. Configurations seven and eight use both ISERDES of an I/O tile together in a master/slave configuration. Configuration seven uses a data width of eight with the top ISERDES in a tile configured as the master and the bottom ISERDES configured as the slave. In this way, the routing of the SHIFTOUT lines of the top ISERDES and the SHIFTIN lines of the bottom ISERDES are tested. Configuration eight tests the

maximum supported data width of ten with the top ISERDES configured as slave and the bottom ISERDES configured as master. This alternating orientation tests the SHIFTOUT lines of the bottom ISERDES and the SHIFTIN lines of the top ISERDES.

The same set of eight BIST configurations is used to test OSERDES components. Table 3.8 shows the various modes of OSERDES components and attributes that are tested in each of the eight BIST configurations. As before, the configuration options and attributes shown in dark grey are already tested in the first set of configurations presented in Section 3.2. However, several of them are tested again with this set of eight BIST configurations.

Table 3.8: BIST Configurations to Test OSERDES

Resources	**Cfg. 1**	**Cfg. 2**	**Cfg. 3**	**Cfg. 4**	**Cfg. 5**	**Cfg. 6**	**Cfg. 7**	**Cfg. 8**
DATARATEOQ	SDR	SDR	DDR	SDR	SDR	SDR	SDR	SDR
DATARATETQ	DDR	BUF	DDR	DDR	SDR	DDR	SDR	SDR
DATAWIDTH	2	3	4	5	6	7	8	10
MODE SERDESi,j	Master	Master	Master	Master	Master	Master	Master/ Slave	Slave/ Master
TRISTATE WIDTH	4	1	2	2	1	4	1	1
INIT_OQ	0	1	0	0	0	1	0	0
INIT_TQ	0	1	0	0	0	1	0	0
SRVAL_OQ	0	0	0	1	0	0	0	1
SRVAL_TQ	0	0	0	1	0	0	0	1
CLKINV	CLK	CLK	CLK_B	CLK_B	CLK	CLK	CLK_B	CLK_B
CLKDIVINV	DIV	DIV	DIV_B	DIV_B	DIV	DIV	DIV_B	DIV_B
D1INV	D1	D1_B	D1	D1	D1	D1	D1	D1_B
D2INV	D2	D2	D2_B	D2	D2	D2	D2	D2_B
D3INV	D3	D3	D3	D3_B	D3	D3	D3	D3_B
D4INV	D4_B	D4	D4	D4	D4	D4	D4	D4_B
D5INV	D5	D5	D5	D5	D5	D5_B	D5	D5_B
D6INV	D6	D6	D6	D6	D6	D6	D6_B	D6_B
OCEINV	OCE	OCE	OCE_B	OCE_B	OCE	OCE	OCE_B	OCE_B
SRINV	SR	SR	SR_B	SR_B	SR	SR	SR_B	SR_B
REVINV	REV	REV	REV_B	REV_B	REV	REV	REV_B	REV_B
T1INV	T1	T1	T1	T1_B	T1	T1	T1	T1
T2INV	T2	T2	T2	T2_B	T2	T2	T2	T2
T3INV	T3	T3	T3	T3	T3	T3_B	T3	T3
T4INV	T4	T4	T4	T4	T4	T4_B	T4	T4
TCEINV	TCE	TCE	TCE_B	TCE_B	TCE	TCE	TCE_B	TCE_B
SHIFTIN1-2 SERDESi,j	Off	Off	Off	Off	Off	Off	Off/ On	On/ Off
SHIFTOUT1-2 SERDESi,j	Off	Off	Off	Off	Off	Off	On/ Off	Off/ On

Configurations one and six test all four of the tristate signals with a tristate width of four. Configurations five and seven test SDR mode of the tristate and data paths with the tristate operating in a single width. Configurations three and four test DDR mode of the tristate and data paths with the tristate operating in a double data width. Finally, configurations seven and eight test the full data width supported in OSERDES

components as well as the shift routing with master/slave I/O tile configurations in both orientations.

The BIST architecture required to test SERDES components is similar to the architecture developed for the previously discussed configurations. Figure 3.4 illustrates an I/O cell configured as an ISERDES, OSERDES, and bidirectional I/O buffer. Test patterns are applied to the six data inputs of the OSERDES component. The tristate and data output responses of the OSERDES are then sent to an I/O buffer configured in a bidirectional mode of operation. The I/O buffer sends serialized data back to the ISERDES components. Finally, the ISERDES component deserializes the data and sends it to ORAs for comparison.

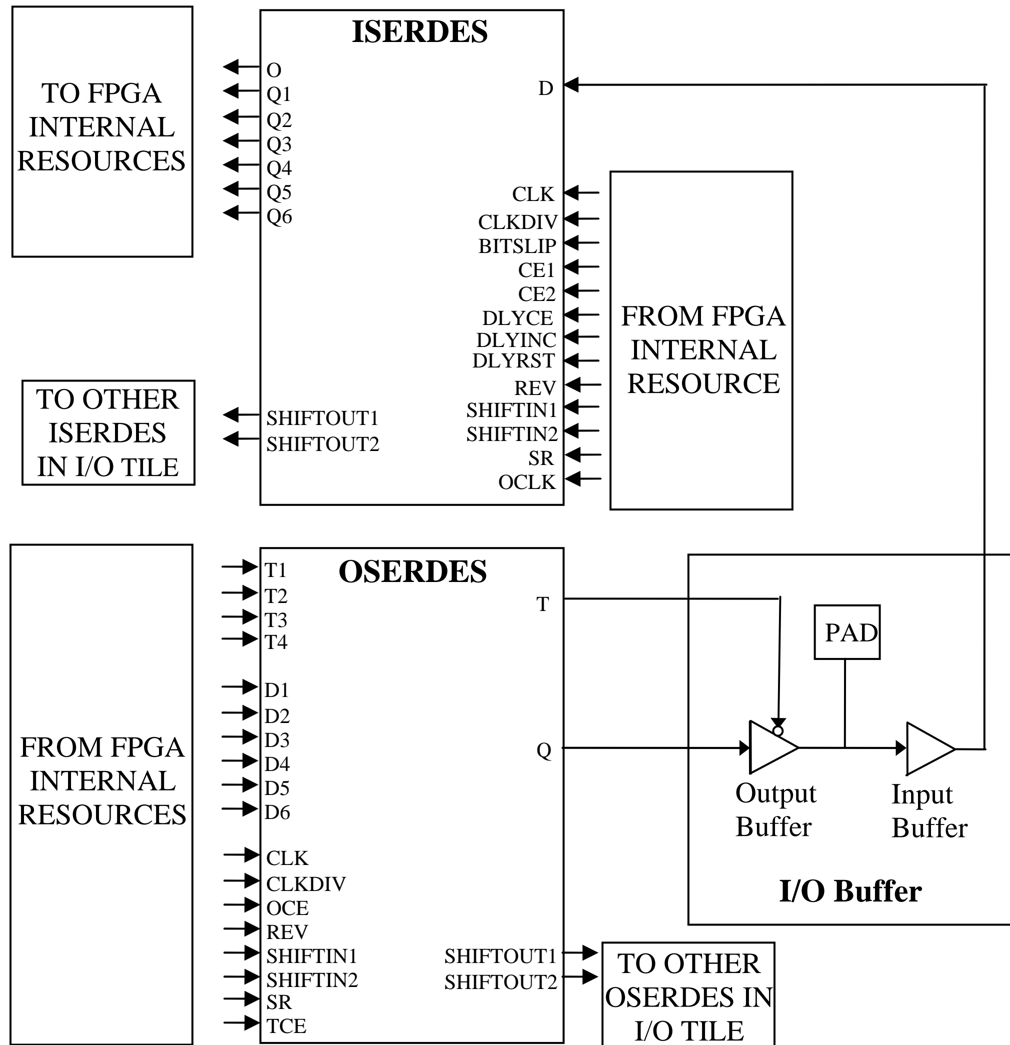


Figure 3.4: Virtex-4 I/O Cell in ISERDES/OSERDES Mode of Operation

The main difference between this architecture and the previously discussed architecture is that SERDES components require more I/O tile input lines to be controlled by TPGs and more output lines to be monitored by ORAs. The TPG test vector width for SERDES configurations is increased to 36 to allow for the 20 TPG outputs needed in SERDES testing. This reduces the total test vector count to 512. The connections of these TPG outputs are summarized in Table 3.9.

Table 3.9: TPG and ORA Connections for SERDES Testing

	Signal	From	To	ORA Positioning
To I/O Tile	T0B0	TPG D0A0	Synchronizer EN 0-1	NA
	T0B1	TPG D0A1	ISERDES CE1	NA
	T0B2	TPG D0A2	ISERDES CE2	NA
	T0B3	TPG D0A3	ISERDES DLYCE	NA
	T0B4	TPG D0A4	ISERDES DLYINC	NA
	T0B5	TPG D0A5	ISERDES DLYRST	NA
	T0B6	TPG D0A6	ISERDES SR	NA
	T0B7	TPG D0A7	OSERDES D1	NA
	T0B8	TPG D0A8	OSERDES D2	NA
	T0B9	TPG D0A9	OSERDES D3	NA
	T0B10	TPG D0A10	OSERDES D4	NA
	T0B11	TPG D0A11	OSERDES D5	NA
	T0B12	TPG D0A12	OSERDES D6	NA
	T0B13	TPG D0A13	OSERDES OCE	NA
	T0B14	TPG D0A14	OSERDES T1	NA
	T0B15	TPG D0A15	OSERDES T2	NA
	T0B16	TPG D0A16	OSERDES T3	NA
	T0B17	TPG D0A17	OSERDES T4	NA
	T0B18	TPG D0A18	OSERDES TCE	NA
	T0B19	TPG D0A19	TPG Bitslip 0-1	NA
	BITSLIP0	BITSLIP0	ISERDES0 BITSLIP	NA
	BITSLIP1	BITSLIP1	ISERDES1 BITSLIP	NA
From I/O Tile	IOB0O	ISERDES O	ORA C1 S00	0X000000
	IOB0Q1	ISERDES Q1	ORA C1 S01	000X0000
	IOB0Q2	ISERDES Q2	ORA C1 S10	00000X00
	IOB0Q3	ISERDES Q3	ORA C1 S11	0000000X
	IOB0Q4	ISERDES Q4	ORA C2 S00	110X0000
	IOB0Q5	ISERDES Q5	ORA C2 S01	11000X00
	IOB0Q6	ISERDES Q6	ORA C2 S10	1100000X
	IOB1O	ISERDES O	ORA C1 S00	X0000000
	IOB1Q1	ISERDES Q1	ORA C1 S01	00X00000
	IOB1Q2	ISERDES Q2	ORA C1 S10	0000X000
	IOB1Q3	ISERDES Q3	ORA C1 S11	000000X0
	IOB1Q4	ISERDES Q4	ORA C2 S01	11X00000
	IOB1Q5	ISERDES Q5	ORA C2 S10	1100X000
	IOB1Q6	ISERDES Q6	ORA C2 S11	110000X0

Table 3.9 also presents the connections from the I/O tiles under test to the monitoring ORAs. Because up to 14 outputs from each I/O tile are being compared (one

combinatorial output and six registered outputs per I/O cell), this BIST configuration set requires 14 ORAs per I/O tile under test. Connections from the I/O tiles under test to the ORAs are done in the same method presented in Figure 3.3. The 14 ORAs are implemented in seven slices in two columns of CLBs adjacent to the I/O tile under test. The first column is directly adjacent to the column of I/O tiles under test and uses all four slices. The ORA positioning in Table 3.9 for this column of ORAs is read as 00 (slice zero) + 00 (slice one) + 00 (slice two) + 00 (slice three), where each slice has two ORAs. The second column of ORAs is read as 11 (slice not used) + 00 (slice four) + 00 (slice five) + 00 (slice six), where each slice has two ORAs. An "X" is used to mark the ORA that is monitoring the given I/O tile signal. Signals from the top and bottom ISERDES of a tile are distinguished as IOB0 and IOB1, respectively.

Another addition to the SERDES BIST architecture is the use of a DCM to produce a divided clock signal. Both ISERDES and OSERDES require a clock input and a divided clock input, the value of which changes for each BIST configuration and is specified in *V4iobmodios*. The divided clock line, CLKDIV, is also used as the BIST clock for every component in the architecture. For each of the eight SERDES BIST configurations, the bottommost DCM in any Virtex-4 device, DCM0, is used to take a user supplied oscillator clock signal and produce the clock and divided clock signals. The DCM must be changed in each configuration to produce the correct divided clock ratio for each data width and rate of operation according to [5]. The eight BIST configurations to test SERDES logic resources, listed as one through eight above, require DCM clock divide values of two, three, four, five, six, seven, eight, and five, respectively. The *V4iobmodios* program developed to modify SERDES BIST

configurations one through eight reconfigures the DCM with the correct clock divide values.

Another difference with SERDES BIST configurations is that SERDES configurations implement a BISTSLIP synchronizer circuit, seen in Figure 3.5. The synchronizer circuit oneshots the BITSLIP module to synchronize the data positioning on the ISERDES output lines. The synchronizer circuit consists of three flip-flops (shown as X, Y, and Z), a four-input AND gate, and a two-input OR gate. The three flip-flops form a shift register with each output being monitored by the four-input AND gate.

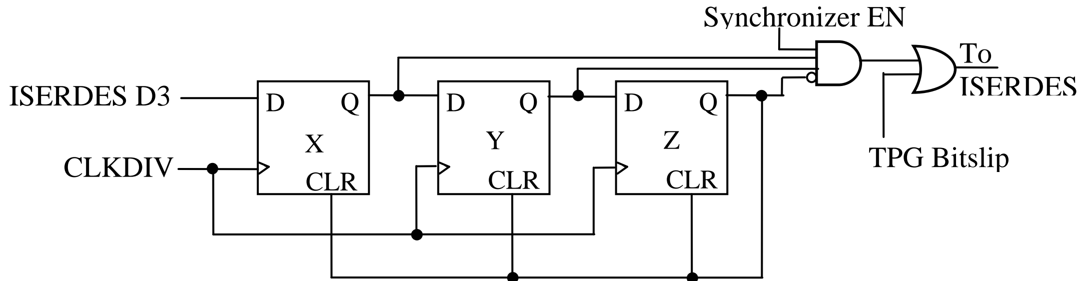


Figure 3.5: SERDES BIST BITSLIP Synchronizer Circuit

With the exception of the first vector, the TPG test vectors were generated using a 32-bit LFSR with external feedback and primitive polynomial $x^{32}+x^{28}+x^{27}+x+1$ to provide a pseudo-random sequence. The first test vector stored in the TPG is used as a training pattern for the ISERDES BITSLIP operation. The training pattern is 00001d81 hexadecimal or 00000000000000000001110110000001 binary, where the least significant bit corresponds to T0B0 of the BRAM TPG outputs. In this training pattern, the TPG line supplied to the OSERDES' D3 data input line contains a logic low. The TPG lines supplied to the D1, D2, D4, D5 and D6 inputs contain logic high values. The remaining TPG lines can be either logic high or low for the training pattern. The input of flip-flop X in the BITSLIP circuit monitors the Q3 output of the ISERDES component

that is supposed to have a logic low output when the BITSLIP operation is completed. During download the circuit is initialized with flip-flop Z containing logic high, as seen in Table 3.10, which strobes synchronous active high clear on the X, Y, and Z flip-flops on the first clock cycle of operation. Then, if the data line being monitored by the X flip-flop contains logic high, the logic high will be shifted through the flip-flops to perform a oneshot BITSLIP action. If the ISERDES data line being monitored by the X flip-flop initializes as logic low, the synchronizer circuit will not oneshot the BITLSIP because the data is already in the correct location.

Table 3.10: SERDES BITSLIP Synchronizer Circuit Timing

Clock Cycle	Init	1	2	3	4	5	6	7	8	9
ISERDES D3	1	1	1	1	1	1	0	0	0	0
X FF	X	0	1	1	1	0	1	0	0	0
Y FF	X	0	0	1	1	0	0	1	0	0
Z FF	1	0	0	0	1	0	0	0	1	0
BITSLIP	0	0	0	1	0	0	0	0	0	0

Assuming the BITSLIP operation has no more than two clock cycle latency, the worst case scenario is that the ISERDES' output data lines will be synchronized in a maximum of 4(N-1) clock cycles where N is the number of parallel data bits. A synchronizer enable line which is controlled by TPGs is one of the four inputs of the AND gate to allow the circuit to be disabled when testing SERDES functions other than the BITSLIP. The output of the AND gate is supplied to a two-input OR gate that outputs the oneshot to the BITSLIP module. The other input of the OR gate is controlled by BRAM TPG vectors, allowing further control and testing of the BITSLIP operation. Each BITSLIP synchronizer circuit is implemented in two slices and is replicated for

each ISERDES component under test to account for ISERDES components located in different areas of the Virtex-4 FPGA synchronizing at different times.

SERDES BIST configurations also implement a TDI logic circuit, seen in Figure 3.6. The TDI logic circuit is used to generate a BIST clock enable to the TPGs and ORAs. The TDI logic circuit is used to manually enable the clock to the TPGs and ORAs, thus providing the BITSLIP synchronizer circuit time to align the ISERDES' outputs. The circuit is also used to generate the IDELAYCTRL RST control. The circuit consists of an AND gate with inputs of TDI and SEL2 of a second boundary scan module, two flip-flops clocked by the divided clock signal from the DCM, and an inverter. The user enables BIST execution by entering USR2 and toggling TDI high. When flip-flop I receives a logic high, the IDELAYCTRL RST will be released. Flip-flop J will receive the logic high on the next clock cycle to enable the BIST circuitry. Two flip-flops are used to synchronize the BIST clock enable signal with the BIST clock and to ensure that the IDELAYCTRL RST and BIST clock enable do not change on the same clock edge. This architecture assumes that by the time the user can enter USR2 and toggle TDI high, the BITSLIP operations have completed. When the BIST clock enable is asserted high, the TPGs can begin outputting test vectors other than the training pattern and the ORAs can begin monitoring the I/O tile responses. The circuit requires a single CLB slice and is only implemented once in each BIST configuration.

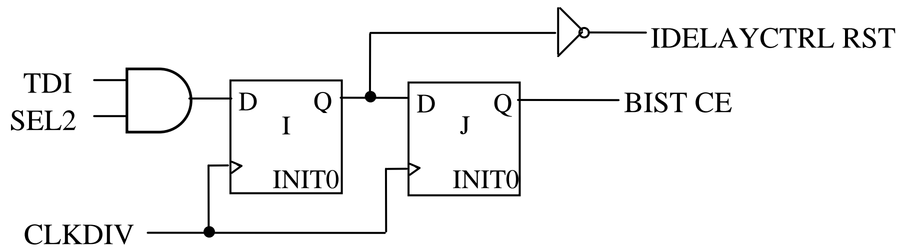


Figure 3.6: SERDES BIST TDI Logic Circuit

3.3 Configurations to Test I/O Standards

Table 3.11 lists each I/O standard and indicates whether it is single-ended or complementary differential. Table 3.11 also indicates if an I/O standard requires a reference voltage, V_{REF}. The table also indicates if the I/O standard uses DCI. Finally, Table 3.11 lists which I/O standards can be tested with each of the 70 required BIST configurations. The nine I/O standards shown as grey boxes in Table 3.11 are not available for testing with this BIST architecture because they do not support a bidirectional mode of operation. Complementary differential standards are listed twice in Table 3.11 because they must be tested in both directions. As seen from the table, a total of 70 BIST configurations are required to test all of the various I/O standards supported by Virtex-4 I/O tiles that can be tested by BIST. The first eight I/O standards are tested in the previously discussed set of BIST configurations, used to test logic resources.

Table 3.11: BIST Configurations to Test I/O Standards Independently

I/O Standard	Type	V_{REF}	Cfg	DCI I/O Standards	Type	V_{REF}	Cfg
LVTTL	SE	N/R	1	LVDCI_33	SE	N/R	36
LVCMOS33	SE	N/R	2	LVDCI_25	SE	N/R	37
LVCMOS25	SE	N/R	3	LVDCI_18	SE	N/R	38
LVCMOS18	SE	N/R	4	LVDCI_15	SE	N/R	39
LVCMOS15	SE	N/R	5	LVDCI_DV2_25	SE	N/R	40
PCI_33_3	SE	N/R	6	LVDCI_DV2_18	SE	N/R	41
PCI_66_3	SE	N/R	7	LVDCI_DV2_15	SE	N/R	42
PCIX	SE	N/R	8	HSTL_II_T_DCI	SER	0.75	43
HSTL_I	SER	0.75	9	HSTL_II_T_DCI_18	SER	0.9	44
HSTL_II	SER	0.75	10	SSTL2_II_T_DCI	SER	1.25	45
HSTL_III	SER	0.9	11	SSTL18_II_T_DCI	SER	0.9	46
HSTL_IV	SER	0.9	12	DIFF_SSTL2_II_DCI	CD	N/R	47
HSTL_I_18	SER	0.9	13	DIFF_SSTL2_II_DCI	CD	N/R	48
HSTL_II_18	SER	0.9	14	DIFF_SSTL18_II_DCI	CD	N/R	49
HSTL_III_18	SER	1.1	15	DIFF_SSTL18_II_DCI	CD	N/R	50
HSTL_IV_18	SER	1.1	16	DIFF_HSTL_II_DCI	CD	N/R	51
HSTL_I_12	SER	0.6	17	DIFF_HSTL_II_DCI	CD	N/R	52
GTL	SER	0.8	18	DIFF_HSTL_II_DCI_18	CD	N/R	53
GTLP	SER	1	19	DIFF_HSTL_II_DCI_18	CD	N/R	54
SSTL2_I	SER	1.25	20	GTL_DCI	SER	0.8	55
SSTL2_II	SER	1.25	21	GTLP_DCI	SER	1	56
SSTL18_I	SER	0.9	22	HSTL_I_DCI	SER	0.75	57
SSTL18_II	SER	0.9	23	HSTL_II_DCI	SER	0.75	58
LVPECL_25	CD	N/R	24	HSTL_III_DCI	SER	0.9	59
LVPECL_25	CD	N/R	25	HSTL_IV_DCI	SER	0.9	60
LVDS_25	CD	N/R	NA	HSTL_I_DCI_18	SER	0.9	61
LVDSEXT_25	CD	N/R	NA	HSTL_II_DCI_18	SER	0.9	62
BLVDS_25	CD	N/R	26	HSTL_III_DCI_18	SER	1.1	63
BLVDS_25	CD	N/R	27	HSTL_IV_DCI_18	SER	1.1	64
ULVDS_25	CD	N/R	NA	SSTL2_I_DCI	SER	1.25	NA
LDT_25	CD	N/R	NA	SSTL2_II_DCI	SER	1.25	65
RSDS_25	CD	N/R	NA	SSTL18_I_DCI	SER	0.9	NA
DIFF_SSTL2_II	CD	N/R	28	SSTL18_II_DCI	SER	0.9	66
DIFF_SSTL2_II	CD	N/R	29	LVDS_25_DCI	CD	N/R	NA
DIFF_SSTL18_II	CD	N/R	30	LVDSEXT_25_DCI	CD	N/R	NA
DIFF_SSTL18_II	CD	N/R	31	HSLVDCI_33	SER	$V_{cco}/2$	67
DIFF_HSTL_II	CD	N/R	32	HSLVDCI_25	SER	$V_{cco}/2$	68
DIFF_HSTL_II	CD	N/R	33	HSLVDCI_18	SER	$V_{cco}/2$	69
DIFF_HSTL_II_18	CD	N/R	34	HSLVDCI_15	SER	$V_{cco}/2$	70
DIFF_HSTL_II_18	CD	N/R	35	**Total Required Configurations**			**70**

SE = Single-Ended SER = Single-Ended Requiring V_{REF}
CD = Complementary Differential

The following sections describe the BIST architecture used to test each type of I/O standard. Section 3.3.1 describes testing single-ended I/O standards, such as LVTTL and LVCMOS33. Section 3.3.2 describes testing single-ended I/O standards that require a voltage reference, such as HSTL and SSTL. Section 3.3.3 describes testing of complementary differential standards that require the use of both I/O buffers in an I/O tile, such as DIFF_HSTL and DIFF_SSTL. Finally, Section 3.3.4 describes testing standards that use DCI line termination.

3.3.1 Testing Single-Ended I/O Standards

A *V4iobist* template file is used to test single-ended standards that do not require a reference voltage. The template file is modified to contain the new I/O standard with a new modification program, *V4iobrmod*. The I/O buffer configuration, shown in Figure 3.7, is a simple bidirectional buffer that adopts a new I/O standard and removes any I/O buffer attributes. The I/O buffer attributes, such as drive strength and slew rate, are removed because most I/O standards require that none be set. Test patterns, shown as the dotted line in Figure 3.7, are sourced to the output buffer of an I/O cell and back into the FPGA through the input buffer.

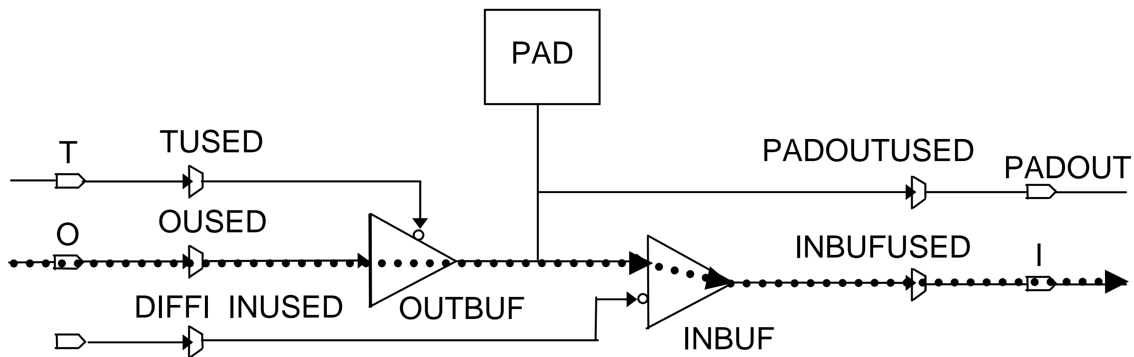


Figure 3.7: I/O Tile BIST Configuration for Single-Ended I/O Standards

3.3.2 Testing Single-Ended I/O Standards Requiring a Reference Voltage

A *V4iobist* template file is used to test single-ended I/O standards that require a reference voltage be supplied to the differential input buffer, as seen in Figure 3.8. The same modification program, *V4iobrmod*, is used to modify the template file to reflect a new I/O standard with all other I/O attributes removed.

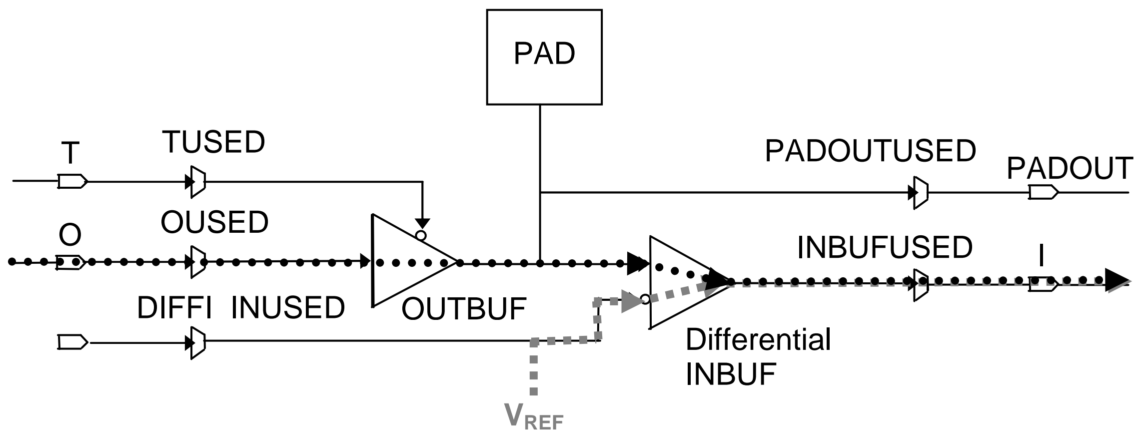


Figure 3.8: I/O Tile BIST Configuration for Single-Ended Standards Requiring a V_{REF}

The BIST architecture is identical for both configurations that test single-ended standards that require a V_{REF} and standards that do not. The difference between BIST configurations developed to test I/O standards that require a V_{REF} and those that do not is that the assigned V_{REF} pins for each I/O bank under test must not be configured. There is one V_{REF} pin assigned for every group of 16 I/O buffers in an I/O bank. *V4iobist* optionally creates templates without V_{REF}, DCI, or both V_{REF} and DCI I/O buffers configured. When executing the BIST configurations, a user must supply the V_{REF} voltage level to the V_{REF} pins of every I/O bank under test.

3.3.3 Testing Complementary Differential I/O Standards

BIST configurations to test complementary differential I/O standards require two data lines to be supplied to the differential input buffer, as seen in Figure 3.9. A TPG data line is supplied to the positive (non-inverting) terminal of the input buffer and its complement is applied to the negative (inverting) input of the buffer in a master I/O cell. The D1INV of an OLOGIC component is used in each I/O tile to produce the complemented TPG data line that is supplied to the slave I/O buffer. The slave I/O cell then sends the complemented data to the differential input buffer of the master I/O cell via its PADOUT line.

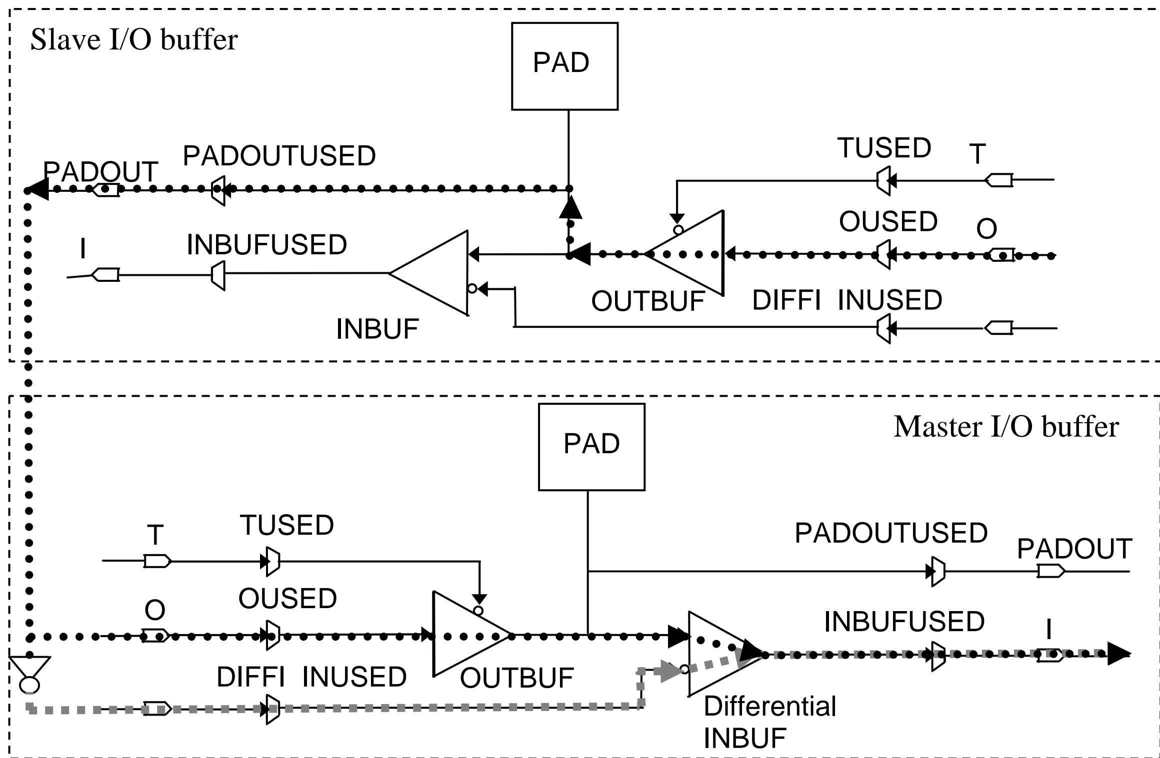


Figure 3.9: I/O Tile BIST Configuration for Complementary Differential I/O Standards

The BIST architecture to test complementary differential I/O standards, shown in Figure 3.10, differs from the BIST architecture used to test single-ended I/O standards.

For each I/O tile, one OLOGIC component is instantiated in the slave I/O cell to complement TPG data. The remaining OLOGIC and ILOGIC components of the I/O tile are instantiated as routethroughs. The ILOGIC of the slave I/O cell is not used at all as it serves no function in complementary differential operation. Another difference is that only two TPG lines are required for each I/O tile under test. One TPG line is used to source test vectors to the data output pins of both I/O buffers in an I/O tile. A second TPG line is used to source test vectors to the tristate control pins of both I/O buffers in an I/O tile. One final difference is that only one ORA is required for each I/O tile as opposed to six used for the ILOGIC, OLOGIC, and I/O buffer BIST configuration set.

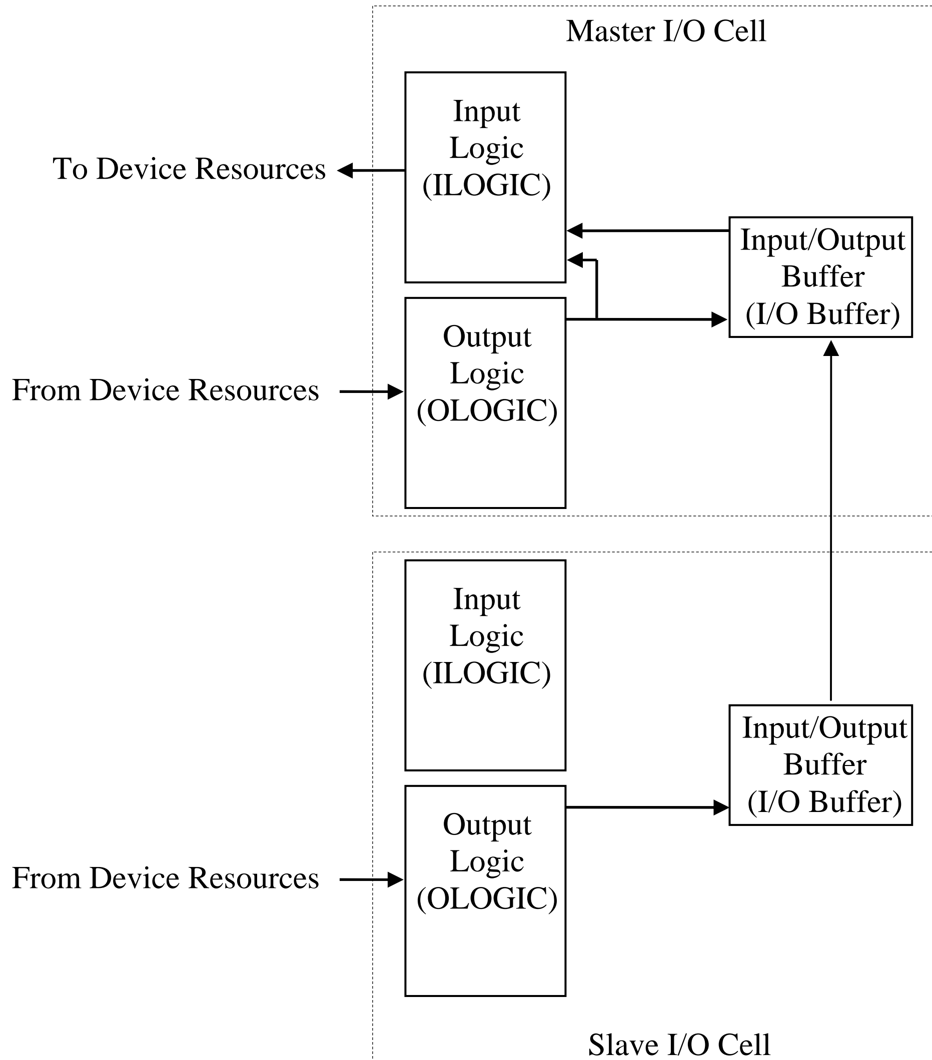


Figure 3.10: I/O Tile Components Used for Complementary Differential I/O Standards

A second template generation program, *V4iobistd*, was created to generate the new I/O tile configuration and to facilitate these differences. The program generates an XDL template file similar to that of *V4iobist*. The TPGs and ORAs are generated identically to that of *V4iobist*. However, the connections made from the TPGs and to the ORAs are different. Only two TPG signals are routed to each I/O tile under test to supply test vectors to the data and tristate signals of each I/O buffer. Because each I/O tile only contains one output line that must be monitored for comparison, the output signal from

each I/O tile is routed to all six ORAs in each row and the six ORAs in the adjacent row. Therefore, if one I/O tile is faulty on every output line, then twelve corresponding ORAs will indicate failures.

The *V4iobistd* template generation program generates both orientations of an I/O tile configured with a complementary differential standard. The first orientation template configures the top and bottom I/O cells of the tile as the master and slave, respectively. The second orientation reverses the master slave roles of the I/O cells under test. Each orientation is required to test the separate PADOUT lines connecting the two I/O cells of an I/O tile. Also, each orientation must be tested for all complementary differential standards because only one differential input buffer of the I/O tile is used per orientation. Table 3.11 lists all complementary differential standards twice to represent that they must be tested in both orientations. After a *V4iobistd* generated XDL template is created and routed by PAR, *V4iobrmod* is used to modify the I/O standard for each separate required BIST configuration.

3.3.4 Testing I/O Standards with Digitally Controlled Impedance

I/O tile BIST configurations to test DCI I/O standards, such as HSTL_I_DCI, are generated in the same manner as their non-DCI counterparts, using a template from *V4iobist* or *V4iobistd* that removes DCI reference pins. When executing DCI BIST configurations, the two DCI multipurpose reference pins, V_{RN} and V_{RP}, in each I/O bank must not be configured. The V_{RN} pin in each bank under test should be connected to V_{CCO} by an external reference resistor equal to the characteristic impedance of the driven line. Similarly, the V_{RP} pin in each bank under test should be connected to ground by an

external reference resistor equal to the characteristic impedance of the driven line. For DCI I/O standards that also require a reference voltage to the differential input buffer, the V_{REF} pins in each I/O bank also must not be configured. Therefore, if testing a full I/O bank with 32 I/O buffers operating in an I/O standard that requires both DCI and V_{REF} inputs, a total of six I/O buffers in that bank cannot be tested with that BIST configuration.

3.4 Summary of I/O Tile BIST Configurations

A set of 78 BIST configurations were developed to test Virtex-4 I/O tiles, as seen in Table 3.12. The overall number of BIST configurations is dominated by the 69 supported I/O standards. Each single-ended I/O standard requires one individual BIST configuration. Each complementary differential I/O standard requires two individual BIST configurations. Nine of the I/O standards cannot be tested because they do not support a bidirectional mode of operation. Thus, 70 total BIST configurations were developed to test the supported I/O standards. This includes the eight configurations developed to test ILOGIC, OLOGIC, and I/O buffer logic resources. The eight configurations developed to test SERDES components should also use the first eight I/O standards because they are the only ones that do not require pins be removed for voltage references or DCI resistors. The experimental implementation of each of these BIST configurations is discussed in the next chapter.

Table 3.12: Summary of Virtex-4 I/O Tile BIST Configurations

Cfg	I/O Standard	Target	Cfg	I/O Standard	Target
1	LVTTL	SERDES	40	HSTL_II_T_DCI_18	SER Standard
2	LVCMOS33	SERDES	41	SSTL2_II_T_DCI	SER Standard
3	LVCMOS25	SERDES	42	SSTL18_II_T_DCI	SER Standard
4	LVCMOS18	SERDES	43	GTL_DCI	SER Standard
5	LVCMOS15	SERDES	44	GTLP_DCI	SER Standard
6	PCI_33_3	SERDES	45	HSTL_I_DCI	SER Standard
7	PCI_66_3	SERDES	46	HSTL_II_DCI	SER Standard
8	PCIX	SERDES	47	HSTL_III_DCI	SER Standard
9	LVTTL	I/O LOGIC	48	HSTL_IV_DCI	SER Standard
10	LVCMOS33	I/O LOGIC	49	HSTL_I_DCI_18	SER Standard
11	LVCMOS25	I/O LOGIC	50	HSTL_II_DCI_18	SER Standard
12	LVCMOS18	I/O LOGIC	51	HSTL_III_DCI_18	SER Standard
13	LVCMOS15	I/O LOGIC	52	HSTL_IV_DCI_18	SER Standard
14	PCI_33_3	I/O LOGIC	53	SSTL2_II_DCI	SER Standard
15	PCI_66_3	I/O LOGIC	54	SSTL18_II_DCI	SER Standard
16	PCIX	I/O LOGIC	55	HSLVDCI_33	SER Standard
17	LVDCI_33	SE Standard	56	HSLVDCI_25	SER Standard
18	LVDCI_25	SE Standard	57	HSLVDCI_18	SER Standard
19	LVDCI_18	SE Standard	58	HSLVDCI_15	SER Standard
20	LVDCI_15	SE Standard	59	LVPECL_25	CD Standard
21	LVDCI_DV2_25	SE Standard	60	BLVDS_25	CD Standard
22	LVDCI_DV2_18	SE Standard	61	DIFF_SSTL2_II	CD Standard
23	LVDCI_DV2_15	SE Standard	62	DIFF_SSTL18_II	CD Standard
24	GTL	SER Standard	63	DIFF_HSTL_II	CD Standard
25	GTLP	SER Standard	64	DIFF_HSTL_II_18	CD Standard
26	SSTL2_I	SER Standard	65	LVPECL_25	CD Standard
27	SSTL2_II	SER Standard	66	BLVDS_25	CD Standard
28	SSTL18_I	SER Standard	67	DIFF_SSTL2_II	CD Standard
29	SSTL18_II	SER Standard	68	DIFF_SSTL18_II	CD Standard
30	HSTL_I	SER Standard	69	DIFF_HSTL_II	CD Standard
31	HSTL_II	SER Standard	70	DIFF_HSTL_II_18	CD Standard
32	HSTL_III	SER Standard	71	DIFF_SSTL2_II_DCI	CD Standard
33	HSTL_IV	SER Standard	72	DIFF_SSTL18_II_DCI	CD Standard
34	HSTL_I_18	SER Standard	73	DIFF_HSTL_II_DCI	CD Standard
35	HSTL_II_18	SER Standard	74	DIFF_HSTL_II_DCI_18	CD Standard
36	HSTL_III_18	SER Standard	75	DIFF_SSTL2_II_DCI	CD Standard
37	HSTL_IV_18	SER Standard	76	DIFF_SSTL18_II_DCI	CD Standard
38	HSTL_I_12	SER Standard	77	DIFF_HSTL_II_DCI	CD Standard
39	HSTL_II_T_DCI	SER Standard	78	DIFF_HSTL_II_DCI_18	CD Standard

SE = Single-Ended **SER = Single-Ended Requiring V_{REF}**
CD = Complementary Differential

The BIST configurations in Table 3.12 are ordered by changes in BIST architecture to reduce partial reconfiguration file size. The first eight configurations test SERDES mode components with the BIST architecture discussed in 3.2.2. BIST configurations 9 through 58 use the same BIST architecture discussed in Section 3.2.1. Configurations 59 through 78 test complementary differential standards with the BIST architecture presented in Section 3.3.3. The groups of complementary differential standards shaded in grey are repeated to test both orientations. The DCI I/O standards are grouped together within each of the sets of configurations.

Chapter Four

Experimental Results

This chapter describes the implementation and testing of the I/O tile BIST configurations presented in Chapter 3. The chapter aims to carefully explain which BIST configurations work as expected, and to explain the problems associated with those that do not. For the BIST configurations that are not working, this chapter will provide a detailed account of generation errors or failing patterns associated with each configuration. This chapter also details other experimental results observed as well as the capabilities and limitations of this BIST approach.

4.1 Generating I/O Tile BIST Configurations

This section begins with a summary of the BIST programs used to generate all 78 I/O tile BIST configurations. Section 4.1.1 describes how the BIST programs function, as well as the command line options that are used to generate the BIST configurations. Section 4.1.2 provides an overview of the generation procedure for all 78 BIST configurations presented in Table 3.12 using the BIST programs developed and Xilinx ISE command line tools. A complete description of the Xilinx ISE command line tools usage can be found in [5].

4.1.1 Summary of I/O Tile BIST Programs

There are three template generation programs and three template modification programs created for Virtex-4 I/O tile BIST. The three template generation programs are *V4iobistios*, *V4iobist*, and *V4iobistd*. Each of the three template programs follows the same basic procedure for generating a XDL template:

1. Instantiate the components in the I/O tiles under test. This also includes the routing between the I/O buffer and the ILOGIC and OLOGIC components.
2. Instantiate the ORA slices (*V4iobistios* also instantiates BITSLIP synchronizer circuits at this point)
3. Instantiate all of the IDELAYCTRLs
4. Instantiate the BRAMs used for TPGs
5. Route the TPGs to the I/O tiles under test
6. Route the I/O tiles under test to the ORAs
7. Instantiate the DSPs used to address the BRAMs
8. Route the DSPs to the BRAM TPGs
9. Instantiate the BSCAN modules (*V4iobistios* also instantiates the TDI logic BIST clock enable circuit at this point)
10. Instantiate the DCMs and clock routing (*V4iobistios* also instantiates the on-board oscillator I/O buffer at this point)

The command line options for *V4iobistios* and *V4iobist* are as follows: <xdlfile> <startrow> <startcol> <endrow> <endcol> <dev> <part> <package> <pad>. The first command line option, <xdlfile>, is the user designated name for the template

file. This name is appended automatically with the .xdl file extension. The <startrow> <startcol> <endrow> <endcol> options allow the user to specify a start row and column beginning at the lower left end of the array as well as an end row and column ending at the upper right end of the area for I/O tiles to be tested. The end row or column values are not included in the I/O tile instantiation. The <dev> <part> <package> options support all valid Virtex-4 FX, LX, and SX device and package combinations, as specified in Chapter 2 and [5]. The final command line option, <pad> supports three possible values: "v" to specify that V_{REF} pins should be removed, "d" to specify that DCI pins should be removed, and "b" to specify that both V_{REF} and DCI pins should be removed. If the <pad> command line option is left blank, neither V_{REF} nor DCI pins are removed.

The command line options for *V4iobistd* are the same as those described for *V4iobistios* and *V4iobist* with one additional option, <configNum>, located in the command option sequence before the <pad> option. The <configNum> option is used to specify which orientation of the complementary differential standard should be generated. Two options are supported: "1" generates a master/slave orientation and "2" generates a slave/master orientation. Master/slave or slave/master orientations configure complementary differential input buffer in the top or bottom I/O cell, respectively. The "v" option should not be used in the <pad> command line option because complementary differential I/O standards do not require V_{REF} pins to be removed.

The three modification programs are *V4iobmodios*, *V4iobmod*, and *V4iobrmod*. Each of the three modification programs work by replacing the configuration settings for each component to be modified. *V4iobmodios* and *V4iobmod* replace the ILOGIC, OLOGIC, and I/O buffer configuration data in each I/O tile under test. *V4iobrmod* is

used for I/O standards and only replaces I/O buffer configuration data in each I/O cell under test. *V4iobmodios* also replaces the DCM configuration data to reflect the proper data width clock division required for SERDES operation.

The command line options for all three modification programs are as follows: <xdl_in> <xdl_out> <phase>. The <xdl_in> <xdl_out> options are used to specify the XDL input file name to be modified and the modified XDL output file name to be created. The .xdl file extension is not automatically appended to the specified names. The <phase> command line option for *V4iobmodios* and *V4iobmod* should be a value between one and eight, where each value is the SERDES or ILOGIC/OLOGIC BIST configuration to be generated, respectively. The <phase> option for *V4iobrmod* should be a value between one and 69, where the value corresponds to the I/O standard in Figure 2.6 by order of appearance. The <phase> value required for each I/O standard can also be seen by typing only the name of the program in the command prompt.

4.1.2 BIST Configuration Generation Procedure

The following procedure can be used to generate all 78 I/O tile BIST configurations for any Virtex-4 device and package. The specific commands for each step can be found in the batch file of Appendix A.

1. Generate all BIST configuration template XDL files
    a. *V4iobistios* – template file for SERDES configurations
    b. *V4iobist* – template file for ILOGIC/OLOGIC configurations
    c. *V4iobist* option d – template file for single-ended DCI I/O standards

d. *V4iobist* option v – template file for single-ended standards requiring V_{REF} pins to be removed

e. *V4iobist* option b – template file for I/O standards requiring both V_{REF} and DCI pins to be removed

f. *V4iobistd* option 1 – template file for master/slave complementary differential standards

g. *V4iobistd* option 2- template file for slave/master complementary differential standards

h. *V4iobistd* option 1 d – template file for master/slave complementary differential standards with DCI

i. *V4iobistd* option 2 d – template file for slave/master complementary differential standards with DCI

2. Convert template XDL files to NCD format using force option
3. Run PAR on NCD template files without the placer option
4. Convert routed NCD template files back to XDL format
5. Modify the BIST template files to BIST configurations IO1 through IO78

a. Modify *V4iobistios* XDL template file with *V4iobmodios* for BIST configurations IO1 through IO8

b. Modify *V4iobist* XDL template file with *V4iobmod* for BIST configurations IO9 through IO16

c. Modify *V4iobist* XDL template file with *V4iobrmod* for BIST configurations IO17 through IO18

d. Modify *V4iobist* option d template file with *V4iobrmod* for BIST configurations IO19 through IO25

e. Modify *V4iobist* option v template file with *V4iobrmod* for BIST configurations IO26 through IO38

f. Modify *V4iobist* option b template file with *V4iobrmod* for BIST configurations IO39 through IO58

g. Modify *V4iobistd* option 1 template file with *V4iobrmod* for BIST configurations IO59, IO61, IO63, IO65, IO67, and IO69

h. Modify *V4iobistd* option 2 template file with *V4iobrmod* for BIST configurations IO60, IO62, IO64, IO66, IO68, and IO70

i. Modify *V4iobistd* option 1 d template file with *V4iobrmod* for BIST configurations IO71, IO73, IO75, and IO77

j. Modify *V4iobistd* option 2 d template file with *V4iobrmod* for BIST configurations IO72, IO74, IO76, and IO78

6. Convert all 78 newly modified XDL files to NCD format using the force option
7. Generate configuration download files for all 78 NCD files
8. Generate partial configuration files for IO2 through IO78 by comparing the programming file of IO(N) to the NCD file of IO(N+1), where $1 < N < 79$

The user has several options to consider when generating I/O tile BIST configurations. IO1 can be generated as a full or compressed configuration programming file. Steps 5-d, 5-e, and 5-f can use the IO10 configuration XDL as a template file, as

discussed in Chapter 3. In order to accomplish this, the template files from steps 1-c, 1-d, and 1-e must first be modified to three distinct versions of IO10 with *V4iobmod* that can then be used in steps 5-d, 5-e, and 5-f, respectively. The user also has the option of stopping at step 7 to generate full or compressed configurations instead of partial configurations. One final step in the generation procedure that may be useful in verifying the BIST configurations would be to create a DRC error report for each of the 78 BIST configurations. This can be seen in the batch file example presented in Appendix A.

Figure 4.1 illustrates an I/O tile BIST configuration generated for a Virtex-4 FX 20 FPGA. Note the single DSP column and two BRAM TPG columns required to test all three columns of I/O tiles. Also note that the BIST routing goes straight through the on-chip Power PC module without conflict.

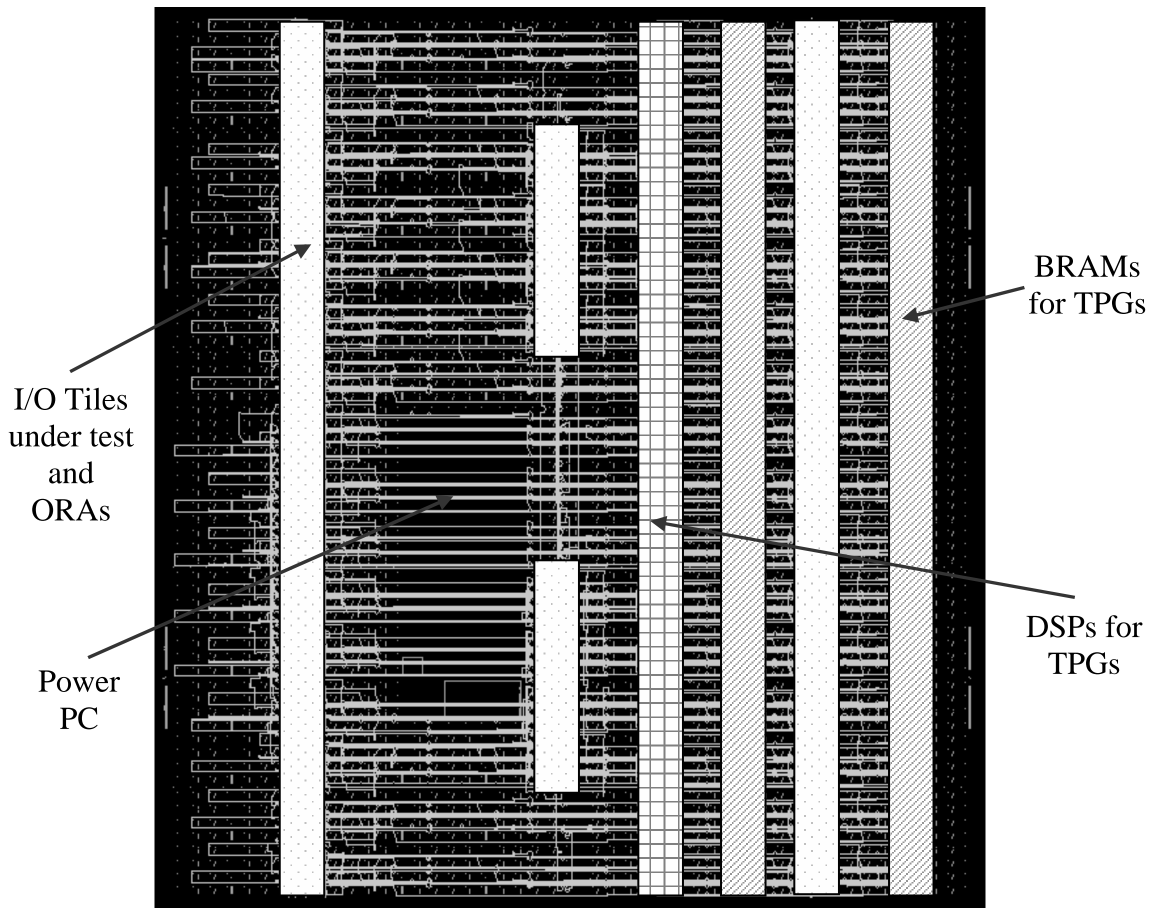


Figure 4.1: All I/O Buffers Under Test of a Virtex-4 FX20 with One Power PC [Screenshot from Xilinx FPGA Editor]

4.2 Executing I/O Tile BIST Configurations

The execution of I/O tile BIST configurations is separated into two main groups: IO1 through IO8 and IO9 through IO78. The two groups require different execution sequences due to architectural differences in the BIST configurations described in Chapter 3. The differences in BIST ORA numbers and locations also require at least two separate configuration memory readbacks to retrieve the BIST results. A readback is required at the end of each BIST configuration group when partial reconfigurations are used. However, a readback is required at the end of each BIST configuration when

partial reconfigurations are not used because the contents of the ORAs will be cleared with each download. The following procedure summarizes executing I/O tile BIST configurations IO1 through IO8 used to test SERDES logic resources:

1. Download the BIST configuration to the device
2. Wait for BISTSLIP synchronizer circuits to finish. This timing is specified in Section 3.2.2.
3. Enter Test Logic Reset
4. Toggle SEL2 high by entering USR2 instruction to the second boundary scan module
5. Toggle TDI to logic high (this enables the TDI logic circuit)
6. Wait for the all 512 TPG test vectors to be cycled through (this depends on the speed of the on-board oscillator used)
7. Read back Pass/Fail results from ORAs (this step only has to be executed once at the end of running each group of BIST configurations if partial reconfiguration files are used)
8. Repeat for each additional BIST configuration

The following procedure summarizes executing I/O tile BIST configurations IO9 through IO78 used to test ILOGIC, OLOGIC, and I/O buffer logic resources as well supported I/O standards:

1. Supply V_{REF} voltages or DCI resistors if required
2. Download BIST configuration to the device
3. Enter Test Logic Reset
4. Toggle SEL1 high by entering USR1 of the first boundary scan module

5. Toggle TDI low to high for fifteen clock cycles to reset IDELAYCTRLs
6. Execute 1024 BIST clock cycles on DRCK1 to cycle through all BRAM TPG test vectors
7. Read back Pass/Fail results from ORAs (this step only has to be executed once at the end of running each group of BIST configurations if partial reconfiguration files are used)
8. Repeat for each additional BIST configuration

4.3 Experimental Results Obtained

The following sections describe the experimental results from generating all 78 I/O tile BIST configurations for all FX, LX, and SX device and package combinations. These sections also describe the experimental results observed when executing the BIST configurations on both Virtex-4 SX 35 package FF668 and LX 60 package FF668 FPGAs on a HW-AFX-FF668-400 prototyping platform. Section 4.3.1 describes the results obtained from generating and executing BIST configurations IO9 through IO16 used to test ILOGIC, OLOGIC, and I/O buffer logic resources. Section 4.3.2 describes the results obtained from generating and executing BIST configurations IO1 through IO8 used to test SERDES logic resources. Finally, section 4.3.3 describes the results obtained from generating and executing BIST configurations IO17 through IO78.

4.3.1 Configurations to Test ILOGIC, OLOGIC, and I/O Buffers

I/O tile BIST configurations IO9 through IO16 produce several DRC errors during the generation process that can be ignored. These errors are related to the input line of the IDELAY modules of the ILOGIC components. The DRC errors report that the

pins DLYCE, DLYINC, and DLYRST of the IDELAY modules are used but have no child signals. These DRC errors occur when the IDELAY module is used in a DEFAULT or FIXED mode in which the DLYCE, DLYINC, and DLYRST inputs are not used. The errors are produced because the TPG routes signals to all of the IDELAY module input lines even when they are not being used by the module. These errors can be ignored during the generation process because they do not affect the functionality of the BIST configurations.

I/O tile BIST configurations IO9 through IO16 also produce several DRC warnings during the generation process that can be ignored during the BIST generation process. The first warning states that the IDELAY module inputs discussed above will be ignored when the IDELAY module is acting in a DEFAULT or FIXED mode. The other two warnings indicate that O1USED and T1USED PIPs of the OLOGIC components have dangling pins. These warnings occur because the BIST configurations always turn on the O1USED and T1USED PIPs, regardless of whether they are being used by the configuration. These warnings could be avoided if the V4iobmod program did not turn on the O1USED and T1USED PIPs when they are not required for the BIST configuration.

BIST configuration IO14 produces DRC errors that cannot be ignored during the generation process. Configuration IO14 tests ILOGIC components with IFF1 configured as an asynchronous latch. The DRC error produced for this configuration states that the ILOGIC components are configured illegally. The errors also state that the remaining flip-flops in the ILOGIC component are required for operation in the latch configuration. It was observed that when the IO14 BIST configuration does not attempt to use IFF1 in

an asynchronous latch mode, the configuration generates without any DRC errors. It should be noted that BIST configurations IO12 and IO13 test OLOGIC components with TFF1 and OFF1 configured as asynchronous latches, respectively. However, these two BIST configurations do not produce DRC errors during generation.

All of the BIST configurations produce passing ORA results, with the exception of IO14 that uses the ILOGIC IFF1 in an asynchronous latch mode. The ORA failing pattern for IO14 indicates that the ORAs comparing only the latch output lines of the bottom ILOGIC components fail. The exact ORA failure positioning can be read as "00010011" when referring to Table 3.6. When the IFF1 component is configured as a flip-flop instead of an asynchronous latch, the BIST configuration produces passing ORA results. The experimental results presented in this section indicate that BIST configurations IO9 through IO16 generate and function correctly, with the exception of testing the ILOGIC in an asynchronous latch mode.

4.3.2 Configurations to Test SERDES Logic Resources

No DRC errors are produced during the generation of SERDES BIST configurations IO1 through IO8. Configurations IO1 through IO5 produce no DRC warnings. Configurations IO6 through IO8 produce warnings of unexpected tristate widths for the SDR and DDR data widths assigned. Configurations IO7 and IO8 warn of connecting the unused shift lines between master and slave SERDES components because *V4iobmodios* leaves the shift lines connected for all eight SERDES configurations. These warnings do also do not affect the operation of the BIST

configurations and could be avoided by adjusting *V4iobmodios* to remove the shift line routing when it is unused.

The execution of the SERDES BIST configurations was performed on an SX35 FPGA with package type FF668, and the failing results are summarized in Table 4.1. Various experiments were designed and performed to determine the cause of the failures. These experiments primarily consisted of probing internal signals to LEDs via I/O pins to examine the internal operation of the TPG, BITLSIP module, BISTSLIP synchronizer circuit, TDI logic circuit, and DCMs as well as the ISERDES and OSERDES themselves. While all of these circuits appeared to be functioning properly, the cause of the failures of the three BIST configurations during execution could not be determined.

Table 4.1: SERDES BIST Execution Results

Configuration	**Pass/Fail**	**ORA Failure Pattern**
IO1	Pass	
IO2	Fail	No recognizable pattern
IO3	Fail	No recognizable pattern
IO4	Pass	
IO5	Pass	
IO6	Pass	
IO7	Fail	DCI pin locations
IO8	Pass	

4.3.3 Configurations to Test I/O Standards

All BIST configurations for I/O standards have been generated and executed, including IO17 through IO78 in Table 3.12. The I/O standards used in the first 16 BIST configurations were also generated from a *V4iobrmod* configuration template for execution testing.

No DRC errors or warnings, aside from those observed in the template file generation discussed previously, occurred when generating BIST configurations IO17 through IO78. The BIST configurations were executed on a Virtex-4 SX35 device with package type FF668. Single-ended standards were tested on every I/O cell of the device. Single-ended standards requiring a V_{REF} were tested on every I/O cell of the device, with the exception of the V_{REF} I/O cells, of the device. The V_{REF} I/O cells were given the voltage level specified in Table 3.11 and [5]. Complementary differential standards were tested on every I/O cell of the device. Both single-ended and complementary differential I/O standards using DCI were tested on every I/O cell of the device, with the exception of DCI reference resistor pins and I/O banks one and two. The V_{RP} and V_{RN} DCI reference resistor pins of each bank were supplied 51 Ohm resistors.

In each case, data lines connecting the I/O tiles under test to their respective ORAs were probed to LEDs on the development board. It was observed that test vectors are sourced through the I/O buffers for each standard being tested. It was also observed that each type of I/O standard produces passing ORA results, with exceptions for expected failures.

Single-ended I/O standards that require a V_{REF} produce expected ORA failures in the ORAs rows associated with the removed V_{REF} pins. For example, Table 2.3 indicates that a V_{REF} pin is located in the bottom I/O cell of row four in bank seven. Removing this pin creates failures sent to the ORAs monitoring the ILOGIC O line of V_{REF} I/O cell. This creates a single ORA failure in the same row as the V_{REF} and row above it, or rows four and five in this example. Therefore, there are two expected ORA failures per V_{REF} pin or eight expected ORA failures per 32 rows of an I/O bank.

Similarly, I/O standards that use DCI produce expected failures in the ORAs associated with the DCI V_{RP} and V_{RN} pins in the manner described for the V_{REF} pins. For example, the DCI pins for the I/O bank shown in Table 2.3 are both located in row nine. Executing a single-ended DCI BIST configuration on this bank produces two expected ORA failures in both rows nine and ten. Therefore, there are four expected ORA failures per 32 rows of an I/O bank for single-ended DCI standards. Executing a complementary differential DCI BIST configuration on this bank produces six expected failures in both rows nine and ten. Therefore, there are twelve expected ORA failures per 32 rows of an I/O bank for single-ended DCI standards. If a DCI standard that also requires a V_{REF} is used, then the expected ORA failures of the DCI pins and V_{REF} pins must both be accounted for. The generation and execution results presented in this section indicate that I/O tile BIST configurations IO17 through IO78 function correctly.

4.4 Capabilities and Limitations

The major capability of this BIST approach is the ability to detect catastrophic faults in routing, logic, and configuration memory bits. Another of the capabilities of this BIST approach is that both bonded and unbonded I/O buffers can be tested, therefore it is a package independent test. The BIST approach can be used to test both the logic resources and supported I/O standards associated with I/O tiles. However, this BIST approach cannot test I/O standards that do not support a bidirectional mode of operation. This BIST approach can be used in both manufacturing and system level testing. However, the BIST approach is sensitive to system component connections. System component connections must be able to operate in a tri-state mode to prevent back

driving that causes BIST failures. The effect of back driving and system loading were observed in the work presented in [9].

One of the limitations of this BIST approach is that it cannot detect all parametric faults associated with the I/O tiles. For example, all of the I/O standards were experimentally tested with the same V_{REF} voltage level of 0.5 Volts. All of the DCI I/O standards were experimentally tested with reference resistor values of 51 Ohms. In both cases, the majority of the I/O standards were tested outside of the specified standard V_{REF} or DCI resistor value operating ranges. However, in both experiments, every I/O standard produced passing ORA results. This indicates that the V_{REF} and DCI values do not have to be adjusted throughout the execution of the BIST configurations. However, this also indicates that the BIST configurations are not robust enough to detect minor faults in V_{REF} voltage levels and DCI resistor values.

One of the limitations for testing Virtex-4 FPGAs is that the current I/O BIST generation programs currently rely on at least two columns of BRAMs for every four rows of I/O Buffers under test. The Virtex-4 FX 12 FPGA contains only a single column of BRAMs in rows where the Power PC lies. Therefore, current BIST configuration generation programs cannot test all of the I/O buffers on an FX 12 device, as seen in Figure 4.2 below. The template programs could be adapted to generate a unique I/O BIST architecture for FX 12 devices. The I/O Buffers in Power PC rows of all FX devices, other than the FX-12, can be tested. This can also be observed in Figure 4.1, which illustrates an FX 20 containing only one Power PC with all I/O tiles under test.

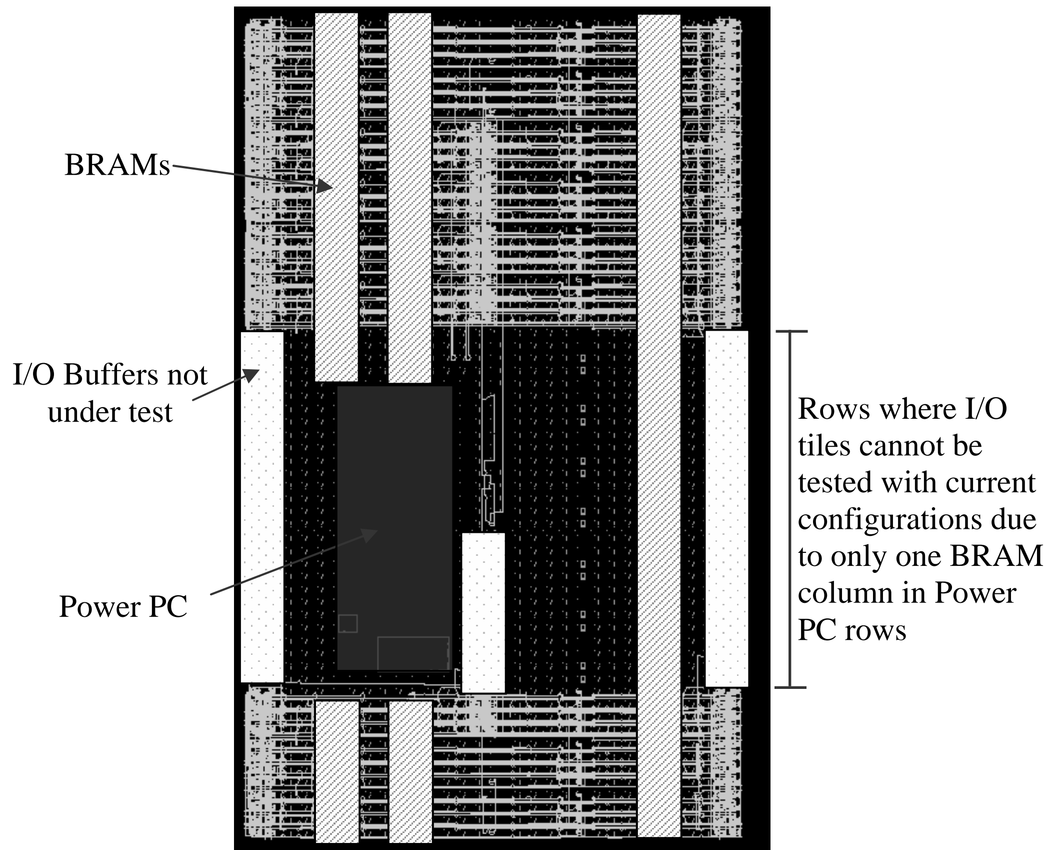


Figure 4.2: Virtex-4 FX 12 I/O Buffers in Power PC Rows Not Under Test
[Screenshot from Xilinx FPGA Editor]

One final limitation of the currently developed BIST configurations is that only the SERDES configurations have been developed to operate at true system speeds. Not testing all of the logic resources at high speeds greatly limits fault detection capabilities. For example, a fault injection simulation was run on the eight configurations presented in this thesis to test ILOGIC, OLOGIC, and I/O buffer logic resources. The BIST configurations were executed using the boundary scan module clock, operating at approximately 50 KHz. At this testing speed, only approximately 60% fault coverage was achieved for the ILOGIC, OLOGIC, and I/O buffer resources. Faults such as slew rate and some of those associated with the IDELAY module were not detected because of

the slow testing speed. Most of the remaining faults that were not detected were associated with the analog programming features, such as pull and drive strength.

CHAPTER FIVE

SUMMARY AND CONCLUSION

This chapter provides summary and highlights of the work presented in this thesis. Section 5.1 presents a summary of the BIST configurations developed to test Virtex-4 I/O tiles. Section 5.2 describes areas of future research and development that can be done to improve the BIST configurations. Finally, Section 5.3 describes the general application this I/O tile BIST approach to other FPGAs and SoCs with FPGA cores.

5.1 Summary of Virtex-4 I/O Tile BIST

The work presented in this thesis develops BIST configurations for the I/O tiles of Virtex-4 FPGAs. Eight configurations were developed to test the ILOGIC, OLOGIC, and I/O buffer logic resources. Eight configurations were also developed to test the logic resources of ISERDES and OSERDES components. A total of 70 BIST configurations were developed to test the Virtex-4 supported I/O standards, including single-ended, single-ended requiring a V_{REF}, complementary differential, and DCI standards. Eight configurations developed to test single-ended I/O standards were combined with the eight configurations to test ILOGIC, OLOGIC, and I/O buffer logic resources. Thus, a total of 78 BIST configurations were developed to test Virtex-4 I/O tiles. The 78 BIST configurations are summarized in Table 3.12 in an order that aims to minimize partial reconfiguration programming file size.

The I/O tile BIST configurations presented in this thesis are package independent because they can be used to test both I/O tiles with bonded and unbonded I/O buffers. The BIST configurations can be used for both manufacturing and system level testing. When using the BIST configurations for system level testing, all connected outputs from external devices must be used in a tri-state mode to prevent back driving. The BIST approach can be used to detect faults in the configuration memory bits associated with I/O tile logic, routing, and analog features. However, the BIST configurations cannot detect all parametric faults that affect analog features, such as V_{REF} voltage levels or DCI resistance values.

This thesis also presents the process and results of generating and executing each of the 78 I/O tile BIST configurations. The 78 BIST configurations were generated for every Virtex-4 device and package combination. They were then downloaded and executed on Virtex-4 SX 35 and LX 60 devices. The thesis outlines existing problems associated with a few of the BIST configurations developed to test logic resources in ILOGIC, OLOGIC, and SERDES components. However, all BIST configurations developed to test I/O standards were shown to function correctly. These BIST configurations will require future development for reliable use for manufacturing and system level testing.

5.2 Areas of Future Research and Development

The major area of future research and development for the work presented in this thesis is to debug and correct the BIST configurations designed to test logic resources. The first issue to be addressed is the problem associated with testing the ILOGIC

components in an asynchronous latch mode. This problem exists in BIST configuration IO14. The next issue to be addressed is to determine the remaining problems associated with the faulty SERDES BIST configurations. The problematic SERDES BIST configurations are IO2, IO3, and IO7. These SERDES BIST configurations do not function correctly when anything but the BITSLIP training pattern is used in the BRAM TPGs as test vectors.

Another area of research and development for the work presented in this thesis is to consolidate BIST template and modification programs and to streamline BIST configuration generation. When used for system level testing, I/O tile BIST configurations should be generated on a per I/O bank basis. The user should then be given the opportunity to specify which I/O cells should not be included in the testing. I/O cells that cannot be configured in a bidirectional mode or are connected to external system components that will cause back driving should not be included in the tests. The user should also be able to define a clock to input a high speed external clock source. Currently, the batch file in Appendix A is used for generating the complete set of BIST configurations. The BIST programs could be developed to generate such batch files automatically that could then be executed by the user.

The BIST configurations should also be developed and generated such that they all can operate at true system speeds. Currently, only the SERDES BIST configurations support using an on-board oscillator to provide a BIST clock. The remaining BIST configurations rely on the boundary scan module to provide a clock, which is significantly slower. Developing all of the BIST configurations such that they can

operate at faster speeds would improve not only the overall BIST testing execution time but also the level of fault detection that can be obtained.

Another area of future research is to investigate the possibility of reducing the number of required I/O tile BIST configurations. For example, it may be possible to reduce the eight configurations developed to test ILOGIC, OLOGIC, and I/O buffer logic resources to only five configurations. A summary of the configuration settings for a reduced set is presented in Appendix B as three separate tables for the ILOGIC, OLOGIC, and I/O buffer components. These configurations could be generated by modifying the V4iobmod program to reflect the configurations settings presented in the three tables. A fault simulation and fault injection analysis should then be performed to ensure that the reduced set of five BIST configurations provides at least the same level of coverage as the existing set of eight.

5.3 General Application to FPGAs and SoCs

The I/O tile BIST approach presented in this thesis is generally applicable to any FPGA or SoC with an FPGA core. However, the BIST configurations need to change between manufacturers and devices. For example, Xilinx Virtex-5 FPGAs have a similar I/O tile structure to the one presented in this thesis for Virtex-4 FPGAs. Virtex-5 I/O tiles contain an ILOGIC, OLOGIC, and I/O buffer in each I/O cell that can also operate in a SERDES mode [10]. However, the I/O tile of a Virtex-5 FPGA has a different routing and component architecture. For example, the IDELAY module is contained between the ILOGIC and OLOGIC components. The BIST approaches and BIST

configuration template generation and modification programs presented in this thesis must be adapted for the architectural differences.

BIBLIOGRAPHY

[1] J. Barth, C. Dyer, and E. Stassinopoulos, "Space, Atmospheric and Terrestrial Radiation Environments," *IEEE Trans. on Nuclear Science*, Vol. 50, No. 3: pp. 466-482, 2003.

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

[3] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing*, New York, NY: Springer, 2000.

[4] S. Trimberger, *Field-Programmable Gate Array Technology*, Boston, MA: Kluwer Academic Publishers, 1994.

[5] __, Virtex-4 User Guide, User Guide UG070 (v 2.1), Xilinx, Inc., 2007 (available at www.xilinx.com).

[6] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Boston, MA: Kluwer Academic Publishers, 1999.

[7] L. Lerner and C. Stroud, "An Architecture for Fail-Silent Operation of FPGAs and Configurable SoCs," *Proc. International Conf. on Embedded Systems and Applications*, pp. 176-182, 2006.

[8] S. Vemula, "Built-In Self-Test for Input/Output Cells in Field Programmable Gate Arrays," Masters Thesis, Auburn University, 2006.

[9] L. Lerner, S. Vemula, and C. Stroud, "System-Level BIST for Programmable I/O Cells in FPGAs and SoCs," *Proc. IEEE North Atlantic Test Workshop*, pp. 1-9, 2006.

[10] __, Virtex-5 User Guide, User Guide UG190 (v 3.0), Xilinx, Inc., 2007 (available at www.xilinx.com).

[11] __, Answers Database, Answer Record #11594, Xilinx, Inc., 2002 (available at www.xilinx.com).

[12] V. Agrawal, C. Kime, and K. Saluja, "A Tutorial on Built-In Self-Test – Part 1: Principles," *Proc. IEEE Design and Test of Computers*, Vol. 10, Issue: 1, pp. 73-82, 1993.

[13] C. Jia and L. Milor, "A BIST Solution for the Test of I/O Speed," *Proc. IEEE International Test Conference*, pp. 1023-1030, 2003.

[14] S. Vemula and C. Stroud, "Built-In Self-Test for I/O Buffers in FPGAs" *Proc. IEEE North Atlantic Test Workshop*, pp. 31-36, 2005.

[15] S. Vemula and C. Stroud, "Built-In Self-Test for Programmable I/O Buffers in FPGAs and SoCs," *Proc. IEEE Southeastern Symp. on System Theory*, pp. 534-538, 2006.

[16] __, Virtex-4 Family Overview, Data Sheet DS112 (v 2.0), Xilinx, Inc., 2007 (available at www.xilinx.com).

[17] __, Virtex-4 RocketIO Multi-Gigabit Transceiver, User Guide UG076 (v 4.0), Xilinx, Inc., 2007 (available at www.xilinx.com).

[18] __, Virtex-4 Product Matrix, Publication 1Q2007, Xilinx, Inc., 2007 (available at www.xilinx.com).

[19] D. Milton, "Built-In Self-Test Configurable Memory Resources in Field Programmable Gate Arrays," Masters Thesis, Auburn University, 2007.

[20] __, XtremeDSP for Virtex-4 FPGAs User Guide, User Guide UG073 (v 2.5), Xilinx, Inc., 2007 (available at www.xilinx.com).

[21] __, Negative-Bias Temperature Instability (NBTI) Effects in 90 nm PMOS, White Paper WP224 (v 1.1), Xilinx, Inc., 2005 (available at www.xilinx.com).

[22] __, Virtex-4 Configuration Guide, User Guide UG071 (v 1.8), Xilinx, Inc., 2007 (available at www.xilinx.com).

[23] W. Mak, "Modern FPGA Constrained Placement," *Asia and South Pacific Design Automation Conference*, Vol. 2, pp. 779-784, Jan. 2005.

[24] __, Virtex-4 Packaging and Pinout Specification, User Guide UG075 (v 3.1), Xilinx, Inc., 2007 (available at www.xilinx.com).

[25] The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard Test Access Port and Boundary-Scan Architecture*, New York, NY: 2001.

[26] G. Gibson, L. Gray, and C. Stroud, "Boundary Scan Access to BIST for Field Programmable Gate Arrays," *Proc. IEEE International Application Specific Integrated Circuits Conf.*, pp. 57-61, 1997.

[27] M. Abramovici and C. E. Stroud, "BIST-based test and diagnosis of FPGA logic blocks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 9, No. 1, pp. 159-172, Feb. 2001.

APPENDIX A

BATCH FILE FOR GENERATING I/O TILE BIST CONFIGURATIONS

```
echo ******************************************************************
echo * This batch file generates all 78 I/O tile BIST configurations for a Virtex-4 SX 35
echo * FPGA.  This batch file can be used to generate I/O tile BIST configurations for
echo * any Virtex-4 device and package by changing the device and package
echo * specifications in the first section of code.
echo ******************************************************************

echo Generate XDL template files
echo Specify the device and package in this section
v4iobistios serdes 0 0 15 1 sx 35 ff668
v4iobist io 0 0 15 1 sx 35 ff668
v4iobist iod 0 0 15 1 sx 35 ff668 d
v4iobist iov 0 0 15 1 sx 35 ff668 v
v4iobist iob 0 0 15 1 sx 35 ff668 b
v4iobistd diff1 0 0 15 1 sx 35 ff668 1
v4iobistd diff2 0 0 15 1 sx 35 ff668 2
v4iobistd diff1d 0 0 15 1 sx 35 ff668 1 d
v4iobistd diff2d 0 0 15 1 sx 35 ff668 2 d

echo Convert XDL template files to NCD files
xdl -xdl2ncd -force serdes
xdl -xdl2ncd -force io
xdl -xdl2ncd -force iod
xdl -xdl2ncd -force iov
xdl -xdl2ncd -force iob
xdl -xdl2ncd -force diff1
xdl -xdl2ncd -force diff2
xdl -xdl2ncd -force diff1d
xdl -xdl2ncd -force diff2d

echo Route the NCD template files
par -p -w serdes.ncd serdes.ncd
par -p -w io.ncd io.ncd
par -p -w iod.ncd iod.ncd
par -p -w iov.ncd iov.ncd
par -p -w iob.ncd iob.ncd
```

```
par -p -w diff1.ncd diff1.ncd
par -p -w diff2.ncd diff2.ncd
par -p -w diff1d.ncd diff1d.ncd
par -p -w diff2d.ncd diff2d.ncd

echo Convert routed NCD template files to XDL files
xdl -ncd2xdl serdes
xdl -ncd2xdl io
xdl -ncd2xdl iod
xdl -ncd2xdl iov
xdl -ncd2xdl iob
xdl -ncd2xdl diff1
xdl -ncd2xdl diff2
xdl -ncd2xdl diff1d
xdl -ncd2xdl diff2d


echo ********************************************
echo BIST configs IO1-IO8
echo 1) SERDES
v4iobmodios serdes.xdl io1.xdl 1
v4iobmodios serdes.xdl io2.xdl 2
v4iobmodios serdes.xdl io3.xdl 3
v4iobmodios serdes.xdl io4.xdl 4
v4iobmodios serdes.xdl io5.xdl 5
v4iobmodios serdes.xdl io6.xdl 6
v4iobmodios serdes.xdl io7.xdl 7
v4iobmodios serdes.xdl io8.xdl 8

echo **********************
echo BIST configs IO9-IO16
echo 2) ILOGIC, OLOGIC, I/O Buffer Configs
v4iobmod io.xdl io9.xdl 1
v4iobmod io.xdl io10.xdl 2
v4iobmod io.xdl io11.xdl 3
v4iobmod io.xdl io12.xdl 4
v4iobmod io.xdl io13.xdl 5
v4iobmod io.xdl io14.xdl 6
v4iobmod io.xdl io15.xdl 7
v4iobmod io.xdl io16.xdl 8
```

```
echo **********************
echo BIST configs IO17-IO23
echo 3) single-ended standards with DCI
v4iobrmod iod.xdl io17.xdl 35
v4iobrmod iod.xdl io18.xdl 36
v4iobrmod iod.xdl io19.xdl 37
v4iobrmod iod.xdl io20.xdl 38
v4iobrmod iod.xdl io21.xdl 39
v4iobrmod iod.xdl io22.xdl 40
v4iobrmod iod.xdl io23.xdl 41

echo **********************
echo BIST configs IO24-IO38
echo 4) single-ended standards requiring Vref
v4iobrmod iov.xdl io24.xdl 18
v4iobrmod iov.xdl io25.xdl 19
v4iobrmod iov.xdl io26.xdl 20
v4iobrmod iov.xdl io27.xdl 21
v4iobrmod iov.xdl io28.xdl 22
v4iobrmod iov.xdl io29.xdl 23
v4iobrmod iov.xdl io30.xdl 9
v4iobrmod iov.xdl io31.xdl 10
v4iobrmod iov.xdl io32.xdl 11
v4iobrmod iov.xdl io33.xdl 12
v4iobrmod iov.xdl io34.xdl 13
v4iobrmod iov.xdl io35.xdl 14
v4iobrmod iov.xdl io36.xdl 15
v4iobrmod iov.xdl io37.xdl 16
v4iobrmod iov.xdl io38.xdl 17

echo **********************
echo BIST configs IO39-IO58
echo 5) single-ended standards requiring Vref and DCI
v4iobrmod iob.xdl io39.xdl 42
v4iobrmod iob.xdl io40.xdl 43
v4iobrmod iob.xdl io41.xdl 44
v4iobrmod iob.xdl io42.xdl 45
v4iobrmod iob.xdl io43.xdl 50
v4iobrmod iob.xdl io44.xdl 51
v4iobrmod iob.xdl io45.xdl 52
v4iobrmod iob.xdl io46.xdl 53
v4iobrmod iob.xdl io47.xdl 54
v4iobrmod iob.xdl io48.xdl 55
v4iobrmod iob.xdl io49.xdl 56
v4iobrmod iob.xdl io50.xdl 57
```

```
v4iobrmod iob.xdl io51.xdl 58
v4iobrmod iob.xdl io52.xdl 59
v4iobrmod iob.xdl io53.xdl 61
v4iobrmod iob.xdl io54.xdl 63
v4iobrmod iob.xdl io55.xdl 66
v4iobrmod iob.xdl io56.xdl 67
v4iobrmod iob.xdl io57.xdl 68
v4iobrmod iob.xdl io58.xdl 69

echo *********************
echo BIST configs IO59-IO70
echo 6) complementary differential standards
v4iobrmod diff1.xdl io59.xdl 24
v4iobrmod diff1.xdl io60.xdl 27
v4iobrmod diff1.xdl io61.xdl 31
v4iobrmod diff1.xdl io62.xdl 32
v4iobrmod diff1.xdl io63.xdl 33
v4iobrmod diff1.xdl io64.xdl 34
v4iobrmod diff2.xdl io65.xdl 24
v4iobrmod diff2.xdl io66.xdl 27
v4iobrmod diff2.xdl io67.xdl 31
v4iobrmod diff2.xdl io68.xdl 32
v4iobrmod diff2.xdl io69.xdl 33
v4iobrmod diff2.xdl io70.xdl 34

echo *********************
echo BIST configs IO71-IO78
echo 7) complementary differential standards with DCI
v4iobrmod diff1d.xdl io71.xdl 46
v4iobrmod diff1d.xdl io72.xdl 47
v4iobrmod diff1d.xdl io73.xdl 48
v4iobrmod diff1d.xdl io74.xdl 49
v4iobrmod diff2d.xdl io75.xdl 46
v4iobrmod diff2d.xdl io76.xdl 47
v4iobrmod diff2d.xdl io77.xdl 48
v4iobrmod diff2d.xdl io78.xdl 49



echo *******************************************
echo Convert configurations to ncd
xdl -xdl2ncd -force -nodrc io1
xdl -xdl2ncd -force -nodrc io2
xdl -xdl2ncd -force -nodrc io3
xdl -xdl2ncd -force -nodrc io4
```

```
xdl -xdl2ncd -force -nodrc io5
xdl -xdl2ncd -force -nodrc io6
xdl -xdl2ncd -force -nodrc io7
xdl -xdl2ncd -force -nodrc io8
xdl -xdl2ncd -force -nodrc io9
xdl -xdl2ncd -force -nodrc io10
xdl -xdl2ncd -force -nodrc io11
xdl -xdl2ncd -force -nodrc io12
xdl -xdl2ncd -force -nodrc io13
xdl -xdl2ncd -force -nodrc io14
xdl -xdl2ncd -force -nodrc io15
xdl -xdl2ncd -force -nodrc io16
xdl -xdl2ncd -force -nodrc io17
xdl -xdl2ncd -force -nodrc io18
xdl -xdl2ncd -force -nodrc io19
xdl -xdl2ncd -force -nodrc io20
xdl -xdl2ncd -force -nodrc io21
xdl -xdl2ncd -force -nodrc io22
xdl -xdl2ncd -force -nodrc io23
xdl -xdl2ncd -force -nodrc io24
xdl -xdl2ncd -force -nodrc io25
xdl -xdl2ncd -force -nodrc io26
xdl -xdl2ncd -force -nodrc io27
xdl -xdl2ncd -force -nodrc io28
xdl -xdl2ncd -force -nodrc io29
xdl -xdl2ncd -force -nodrc io30
xdl -xdl2ncd -force -nodrc io31
xdl -xdl2ncd -force -nodrc io32
xdl -xdl2ncd -force -nodrc io33
xdl -xdl2ncd -force -nodrc io34
xdl -xdl2ncd -force -nodrc io35
xdl -xdl2ncd -force -nodrc io36
xdl -xdl2ncd -force -nodrc io37
xdl -xdl2ncd -force -nodrc io38
xdl -xdl2ncd -force -nodrc io39
xdl -xdl2ncd -force -nodrc io40
xdl -xdl2ncd -force -nodrc io41
xdl -xdl2ncd -force -nodrc io42
xdl -xdl2ncd -force -nodrc io43
xdl -xdl2ncd -force -nodrc io44
xdl -xdl2ncd -force -nodrc io45
xdl -xdl2ncd -force -nodrc io46
xdl -xdl2ncd -force -nodrc io47
xdl -xdl2ncd -force -nodrc io48
xdl -xdl2ncd -force -nodrc io49
```

```
xdl -xdl2ncd -force -nodrc io50
xdl -xdl2ncd -force -nodrc io51
xdl -xdl2ncd -force -nodrc io52
xdl -xdl2ncd -force -nodrc io53
xdl -xdl2ncd -force -nodrc io54
xdl -xdl2ncd -force -nodrc io55
xdl -xdl2ncd -force -nodrc io56
xdl -xdl2ncd -force -nodrc io57
xdl -xdl2ncd -force -nodrc io58
xdl -xdl2ncd -force -nodrc io59
xdl -xdl2ncd -force -nodrc io60
xdl -xdl2ncd -force -nodrc io61
xdl -xdl2ncd -force -nodrc io62
xdl -xdl2ncd -force -nodrc io63
xdl -xdl2ncd -force -nodrc io64
xdl -xdl2ncd -force -nodrc io65
xdl -xdl2ncd -force -nodrc io66
xdl -xdl2ncd -force -nodrc io67
xdl -xdl2ncd -force -nodrc io68
xdl -xdl2ncd -force -nodrc io69
xdl -xdl2ncd -force -nodrc io70
xdl -xdl2ncd -force -nodrc io71
xdl -xdl2ncd -force -nodrc io72
xdl -xdl2ncd -force -nodrc io73
xdl -xdl2ncd -force -nodrc io74
xdl -xdl2ncd -force -nodrc io75
xdl -xdl2ncd -force -nodrc io76
xdl -xdl2ncd -force -nodrc io77
xdl -xdl2ncd -force -nodrc io78


echo ********************************************
echo Generate BIST configurations, -j must be removed to later generate partials
echo  Only have to generate a few here for partials
bitgen -d -w io1
bitgen -d -w io2
bitgen -d -w io3
bitgen -d -w io4
bitgen -d -w io5
bitgen -d -w io6
bitgen -d -w io7
bitgen -d -w io8
bitgen -d -w io9
bitgen -d -w io10
```

```
bitgen -d -w io11
bitgen -d -w io12
bitgen -d -w io13
bitgen -d -w io14
bitgen -d -w io15
bitgen -d -w io16
bitgen -d -w io17
bitgen -d -w io18
bitgen -d -w io19
bitgen -d -w io20
bitgen -d -w io21
bitgen -d -w io22
bitgen -d -w io23
bitgen -d -w io24
bitgen -d -w io25
bitgen -d -w io26
bitgen -d -w io27
bitgen -d -w io28
bitgen -d -w io29
bitgen -d -w io30
bitgen -d -w io31
bitgen -d -w io32
bitgen -d -w io33
bitgen -d -w io34
bitgen -d -w io35
bitgen -d -w io36
bitgen -d -w io37
bitgen -d -w io38
bitgen -d -w io39
bitgen -d -w io40
bitgen -d -w io41
bitgen -d -w io42
bitgen -d -w io43
bitgen -d -w io44
bitgen -d -w io45
bitgen -d -w io46
bitgen -d -w io47
bitgen -d -w io48
bitgen -d -w io49
bitgen -d -w io50
bitgen -d -w io51
bitgen -d -w io52
bitgen -d -w io53
bitgen -d -w io54
bitgen -d -w io55
```

```
bitgen -d -w io56
bitgen -d -w io57
bitgen -d -w io58
bitgen -d -w io59
bitgen -d -w io60
bitgen -d -w io61
bitgen -d -w io62
bitgen -d -w io63
bitgen -d -w io64
bitgen -d -w io65
bitgen -d -w io66
bitgen -d -w io67
bitgen -d -w io68
bitgen -d -w io69
bitgen -d -w io70
bitgen -d -w io71
bitgen -d -w io72
bitgen -d -w io73
bitgen -d -w io74
bitgen -d -w io75
bitgen -d -w io76
bitgen -d -w io77
bitgen -d -w io78


echo ********************************************
echo Generate partial BIST configurations for faster test time
bitgen -d -j -b -w -r io1.bit io2.ncd io2p
bitgen -d -j -b -w -r io2.bit io3.ncd io3p
bitgen -d -j -b -w -r io3.bit io4.ncd io4p
bitgen -d -j -b -w -r io4.bit io5.ncd io5p
bitgen -d -j -b -w -r io5.bit io6.ncd io6p
bitgen -d -j -b -w -r io6.bit io7.ncd io7p
bitgen -d -j -b -w -r io7.bit io8.ncd io8p
bitgen -d -j -b -w -r io8.bit io9.ncd io9p
bitgen -d -j -b -w -r io9.bit io10.ncd io10p
bitgen -d -j -b -w -r io10.bit io11.ncd io11p
bitgen -d -j -b -w -r io11.bit io12.ncd io12p
bitgen -d -j -b -w -r io12.bit io13.ncd io13p
bitgen -d -j -b -w -r io13.bit io14.ncd io14p
bitgen -d -j -b -w -r io14.bit io15.ncd io15p
bitgen -d -j -b -w -r io15.bit io16.ncd io16p
bitgen -d -j -b -w -r io16.bit io17.ncd io17p
bitgen -d -j -b -w -r io17.bit io18.ncd io18p
```

```
bitgen -d -j -b -w -r io18.bit io19.ncd io19p
bitgen -d -j -b -w -r io19.bit io20.ncd io20p
bitgen -d -j -b -w -r io20.bit io21.ncd io21p
bitgen -d -j -b -w -r io21.bit io22.ncd io22p
bitgen -d -j -b -w -r io22.bit io23.ncd io23p
bitgen -d -j -b -w -r io23.bit io24.ncd io24p
bitgen -d -j -b -w -r io24.bit io25.ncd io25p
bitgen -d -j -b -w -r io25.bit io26.ncd io26p
bitgen -d -j -b -w -r io26.bit io27.ncd io27p
bitgen -d -j -b -w -r io27.bit io28.ncd io28p
bitgen -d -j -b -w -r io28.bit io29.ncd io29p
bitgen -d -j -b -w -r io29.bit io30.ncd io30p
bitgen -d -j -b -w -r io30.bit io31.ncd io31p
bitgen -d -j -b -w -r io31.bit io32.ncd io32p
bitgen -d -j -b -w -r io32.bit io33.ncd io33p
bitgen -d -j -b -w -r io33.bit io34.ncd io34p
bitgen -d -j -b -w -r io34.bit io35.ncd io35p
bitgen -d -j -b -w -r io35.bit io36.ncd io36p
bitgen -d -j -b -w -r io36.bit io37.ncd io37p
bitgen -d -j -b -w -r io37.bit io38.ncd io38p
bitgen -d -j -b -w -r io38.bit io39.ncd io39p
bitgen -d -j -b -w -r io39.bit io40.ncd io40p
bitgen -d -j -b -w -r io40.bit io41.ncd io41p
bitgen -d -j -b -w -r io41.bit io42.ncd io42p
bitgen -d -j -b -w -r io42.bit io43.ncd io43p
bitgen -d -j -b -w -r io43.bit io44.ncd io44p
bitgen -d -j -b -w -r io44.bit io45.ncd io45p
bitgen -d -j -b -w -r io45.bit io46.ncd io46p
bitgen -d -j -b -w -r io46.bit io47.ncd io47p
bitgen -d -j -b -w -r io47.bit io48.ncd io48p
bitgen -d -j -b -w -r io48.bit io49.ncd io49p
bitgen -d -j -b -w -r io49.bit io50.ncd io50p
bitgen -d -j -b -w -r io50.bit io51.ncd io51p
bitgen -d -j -b -w -r io51.bit io52.ncd io52p
bitgen -d -j -b -w -r io52.bit io53.ncd io53p
bitgen -d -j -b -w -r io53.bit io54.ncd io54p
bitgen -d -j -b -w -r io54.bit io55.ncd io55p
bitgen -d -j -b -w -r io55.bit io56.ncd io56p
bitgen -d -j -b -w -r io56.bit io57.ncd io57p
bitgen -d -j -b -w -r io57.bit io58.ncd io58p
bitgen -d -j -b -w -r io58.bit io59.ncd io59p
bitgen -d -j -b -w -r io59.bit io60.ncd io60p
bitgen -d -j -b -w -r io60.bit io61.ncd io61p
bitgen -d -j -b -w -r io61.bit io62.ncd io62p
bitgen -d -j -b -w -r io62.bit io63.ncd io63p
```

```
bitgen -d -j -b -w -r io63.bit io64.ncd io64p
bitgen -d -j -b -w -r io64.bit io65.ncd io65p
bitgen -d -j -b -w -r io65.bit io66.ncd io66p
bitgen -d -j -b -w -r io66.bit io67.ncd io67p
bitgen -d -j -b -w -r io67.bit io68.ncd io68p
bitgen -d -j -b -w -r io68.bit io69.ncd io69p
bitgen -d -j -b -w -r io69.bit io70.ncd io70p
bitgen -d -j -b -w -r io70.bit io71.ncd io71p
bitgen -d -j -b -w -r io71.bit io72.ncd io72p
bitgen -d -j -b -w -r io72.bit io73.ncd io73p
bitgen -d -j -b -w -r io73.bit io74.ncd io74p
bitgen -d -j -b -w -r io74.bit io75.ncd io75p
bitgen -d -j -b -w -r io75.bit io76.ncd io76p
bitgen -d -j -b -w -r io76.bit io77.ncd io77p
bitgen -d -j -b -w -r io77.bit io78.ncd io78p


echo *********************************************
echo Generate DRC errors reports for review
drc -s -v -o io1.tdr io1.ncd
drc -s -v -o io2.tdr io2.ncd
drc -s -v -o io3.tdr io3.ncd
drc -s -v -o io4.tdr io4.ncd
drc -s -v -o io5.tdr io5.ncd
drc -s -v -o io6.tdr io6.ncd
drc -s -v -o io7.tdr io7.ncd
drc -s -v -o io8.tdr io8.ncd
drc -s -v -o io9.tdr io9.ncd
drc -s -v -o io10.tdr io10.ncd
drc -s -v -o io11.tdr io11.ncd
drc -s -v -o io12.tdr io12.ncd
drc -s -v -o io13.tdr io13.ncd
drc -s -v -o io14.tdr io14.ncd
drc -s -v -o io15.tdr io15.ncd
drc -s -v -o io16.tdr io16.ncd
drc -s -v -o io17.tdr io17.ncd
drc -s -v -o io18.tdr io18.ncd
drc -s -v -o io19.tdr io19.ncd
drc -s -v -o io20.tdr io20.ncd
drc -s -v -o io21.tdr io21.ncd
drc -s -v -o io22.tdr io22.ncd
drc -s -v -o io23.tdr io23.ncd
drc -s -v -o io24.tdr io24.ncd
drc -s -v -o io25.tdr io25.ncd
```

```
drc -s -v -o io26.tdr io26.ncd
drc -s -v -o io27.tdr io27.ncd
drc -s -v -o io28.tdr io28.ncd
drc -s -v -o io29.tdr io29.ncd
drc -s -v -o io30.tdr io30.ncd
drc -s -v -o io31.tdr io31.ncd
drc -s -v -o io32.tdr io32.ncd
drc -s -v -o io33.tdr io33.ncd
drc -s -v -o io34.tdr io34.ncd
drc -s -v -o io35.tdr io35.ncd
drc -s -v -o io36.tdr io36.ncd
drc -s -v -o io37.tdr io37.ncd
drc -s -v -o io38.tdr io38.ncd
drc -s -v -o io39.tdr io39.ncd
drc -s -v -o io40.tdr io40.ncd
drc -s -v -o io41.tdr io41.ncd
drc -s -v -o io42.tdr io42.ncd
drc -s -v -o io43.tdr io43.ncd
drc -s -v -o io44.tdr io44.ncd
drc -s -v -o io45.tdr io45.ncd
drc -s -v -o io46.tdr io46.ncd
drc -s -v -o io47.tdr io47.ncd
drc -s -v -o io48.tdr io48.ncd
drc -s -v -o io49.tdr io49.ncd
drc -s -v -o io50.tdr io50.ncd
drc -s -v -o io51.tdr io51.ncd
drc -s -v -o io52.tdr io52.ncd
drc -s -v -o io53.tdr io53.ncd
drc -s -v -o io54.tdr io54.ncd
drc -s -v -o io55.tdr io55.ncd
drc -s -v -o io56.tdr io56.ncd
drc -s -v -o io57.tdr io57.ncd
drc -s -v -o io58.tdr io58.ncd
drc -s -v -o io59.tdr io59.ncd
drc -s -v -o io60.tdr io60.ncd
drc -s -v -o io61.tdr io61.ncd
drc -s -v -o io62.tdr io62.ncd
drc -s -v -o io63.tdr io63.ncd
drc -s -v -o io64.tdr io64.ncd
drc -s -v -o io65.tdr io65.ncd
drc -s -v -o io66.tdr io66.ncd
drc -s -v -o io67.tdr io67.ncd
drc -s -v -o io68.tdr io68.ncd
drc -s -v -o io69.tdr io69.ncd
drc -s -v -o io70.tdr io70.ncd
```

```
drc -s -v -o io71.tdr io71.ncd
drc -s -v -o io72.tdr io72.ncd
drc -s -v -o io73.tdr io73.ncd
drc -s -v -o io74.tdr io74.ncd
drc -s -v -o io75.tdr io75.ncd
drc -s -v -o io76.tdr io76.ncd
drc -s -v -o io77.tdr io77.ncd
drc -s -v -o io78.tdr io78.ncd

echo ******************************************************************
echo End of BIST configuration batch file for SX 35
echo ******************************************************************


echo ******************************************************************
echo * The following code links this batch file to a similar but separate batch file used to
echo * generate all 78 I/O tile BIST configurations for a Virtex-4 SX 55 FF1147.  This
echo * method can be used to generate I/O tile BIST configurations for all Virtex-4
echo * device and package combinations.
cd..
cd sx55ff1148
sx55ff1148.bat
```

APPENDIX B

REDUCED CONFIGURATION FOR ILOGIC, OLOGIC, AND I/O BUFFERS

Table B.1: Proposed Configuration Modes of ILOGIC Components

Cfg. Bits	Cfg. 1	Cfg. 2	Cfg. 3	Cfg. 4	Cfg. 5
DELAYMUX	OFF	1	0	1	OFF
DELMUX	1	0	0	0	OFF
D2OBYPSEL	OFF	1	OFF	OFF	GND
IMUX	1	0	1	1	0
IFFDELMUX	1	0	OFF	0	OFF
D2OFFBSEL	GND	1	GND	1	OFF
IFFMUX	1	0	0	1	1
CE1INV	CE1	CE1_B	CE1	CE1_B	CE1
CLKINV	CLK	CLK_B	CLK	CLK_B	CLK
SRINV	SR	SR_B	SR	SR_B	SR
REVINV	REV	REV_B	REV	REV_B	REV
Q1MUX	IFF1	IFF3	IFF1	IFF1	IFF3
Q2MUX	IFF2	IFF4	IFF2	IFF2	IFF4
IFF1	FF	FF	FF	LATCH	FF
SRVAL[1:4]	1111	0000	0000	1100	0011
INIT[1:4]	1111	0011	0000	0000	1100
SRTYPE	Sync	Sync	Sync	Async	Sync
CLKDIVINV	OFF	OFF	0	1	OFF
DELAYVAL.	0	0	2	63	0
DELAY TYPE	Default	Default	Fixed	Variable	Default

Table B.2: Proposed Configuration Modes of OLOGIC Components

Cfg. Bits	**Cfg. 1**	**Cfg. 2**	**Cfg. 3**	**Cfg. 4**	**Cfg. 5**
T1INV	T1_B	T1	T1_B	T1	T1
T2INV	OFF	OFF	T2	T2_B	OFF
TCEINV	OFF	TCE	TCE_B	TCE	TCE
CLK1INV	OFF	CLK1	CLK1_B	CLK1	CLK
CLK2INV	OFF	CLK2	CLK2_B	CLK2_B	CLK
D1INV	D1_B	D1	D1_B	D1	D1
D2INV	OFF	OFF	D2	D2_B	OFF
OCEINV	OFF	OCE	OCE_B	OCE	OCE
SRINV	OFF	SR	SR_B	SR	SR
REVINV	OFF	REV	REV_B	REV_B	REV
TFF1	FF	FF	FF	FF	LATCH
OFF1	FF	FF	FF	FF	LATCH
TMUX	T1	TFF1	TFFDDRA	TFFDDRB	TFF1
OMUX	D1	OFF1	OFFDDRA	OFFDDRB	OFF1
SRVALOQ	0	1	0	1	0
SRVALTQ	0	1	0	1	0
INITOQ	0	0	0	1	0
INITTQ	0	0	0	1	0
SRTYPEOQ	Sync	Sync	Sync	Sync	Async
SRTYPETQ	Sync	Sync	Sync	Sync	Async

Table B.3: Proposed Configuration Modes of I/O Buffer Components

Cfg. Bits	**Cfg. 1**	**Cfg. 2**	**Cfg. 3**	**Cfg. 4**	**Cfg. 5**
PULL	Keeper	Down	Pullup	Keeper	Down
GTSATTR	Off	Disable	Disable	Off	Disable
SLEW	Slow	Fast	Slow	Off	Slow
DIFFTERM	NA	NA	NA	NA	NA
DRIVE	24	4	6	Off	2
DRIVE0MA	Off	Drive0	Drive0	Off	Drive0
IOATTR	LVTTL	LVCM-OS33	LVCM-OS25	PCI66-3	LVCM-OS15

APPENDIX C

LIST OF ACRONYMS

ATE – Automatic Test Equipment

BIST – Built-In Self-Test

BUT – Block Under Test

BRAM – Block Random Access Memory

CD – Complementary Differential

CLB – Configurable Logic Block

CUT – Circuit Under Test

DCI – Digitally Controlled Impedance

DCM – Digital Clock Manager

DDR – Double Data Rate

DFT – Design for Testability

DSP – Digital Signal Processor

FF – Flip-Flop

FPGA – Field Programmable Gate Array

I/O – Input/Output

IDELAY – Input Delay Module

ILOGIC – Input Logic

ISERDES – Input SERDES

LUT – Look-Up Table

OLOGIC – Output Logic

ORA – Output Response Analyzer

OSERDES – Output SERDES

PIP – Programmable Interconnect Point

PLB – Programmable Logic Block

PMV – Predicted Motion Vector

REV – Reset

SDR – Single Data Rate

SE – Single-Ended

SER – Single-Ended Requiring V_{REF}

SERDES – Serializer/Deserializer

SoC – System-on-Chip

SR – Set

SRAM – Static Random Access Memory

TPG – Test Pattern Generator

VLSI – Very Large Scale Integration

V_{REF} – Reference Voltage