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

_____

Sudheer Vemula

Certificate of Approval:

_____	_____
Vishwani D. Agrawal Professor Electrical and Computer Engineering	Charles E. Stroud, Chair Professor Electrical and Computer Engineering
_____	_____
Victor P. Nelson Professor Electrical and Computer Engineering	Stephen L. McFarland Acting Dean Graduate School

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

Sudheer Vemula

A Thesis

Submitted to

the Graudate Faculty of

Auburn University

in Partial Fulfilment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
August 7, 2006

BUILT-IN SELF-TEST FOR INPUT/OUTPUT CELLS IN FIELD

PROGRAMMABLE GATE ARRAYS

Sudheer Vemula

_____

Signature of Author

_____

Date of Graduation

VITA

Sudheer Vemula, son of Sudhakar and Sujana Rani Vemula, was born on June 22 1983 in Visakhapatnam, India. He graduated with Bachelor of Technology in Electronics and Communications Engineering degree in April 2004 from G. Pulla Reddy Engineering College affiliated to Sri Krishnadevaraya University, Anantapur, India with distinction. After completion of his undergraduate degree, he entered the graduate program in Electrical and Computer Engineering at Auburn University in August, 2004. While in pursuit of his Master of Science degree at Auburn University, he worked under the guidance of Dr. Charles E. Stroud as a graduate research assistant in the Electrical and Computer Engineering Department.

THESIS ABSTRACT

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

Sudheer Vemula

Master of Science, August 7, 2006
(B. Tech., Sri Krishnadevaraya University, Anantapur, India, 2004)

104 Typed Pages

Directed by Charles E. Stroud

Programmable Input/Output (I/O) cells are an integral part of any Field Programmable Gate Array (FPGA). The resources associated with the programmable I/O cells are increasing as newer architectures of FPGAs are being developed and this increases the importance of testing them. A general Built-In Self-Test (BIST) architecture to test the programmable I/O cells in FPGAs or associated with the FPGA core of System-on-Chip (SoC) implementations is proposed. The I/O cells are tested for various modes of operation along with their associated programmable routing resources.

The proposed BIST architecture has been implemented and verified on Atmel AT94K10 and AT94K40 SoCs. A total of 161 and 303 configuration downloads are required to test the I/O cells of AT94K10 and AT94K40 devices, respectively. The use of an embedded processor for dynamic partial reconfiguration reduced the number of

configuration downloads to three for both the AT94K10 and AT94K40 devices. The implementation of dynamic partial reconfiguration gave a speed up of 99.39 times in test time and a reduction in configuration memory storage requirements by 101 times for AT94K40 devices.

ACKNOWLEDGEMENTS

I would like to thank Dr. Stroud for his support and advice throughout my research at Auburn University. I would also like to thank Dr. Agrawal and Dr. Nelson for being on my graduate committee and for their contribution to my thesis. I would like to acknowledge my research colleagues Daniel, Lee, John, Jonathan, Sachin and Srinivas for their help during my research and for creating a great environment in the Auburn University Built-In Self-Test (AUBIST) lab. I would like to thank all my friends and mentors in US and back in India for everything they did for me. Finally, I would like to express my deepest gratitude to my parents and relatives whose love and encouragement is inspiring me to achieve my goals.

Style manual or journal used IEEE (Institute of Electrical and Electronics Engineers) Journal Style.

Computer software used Microsoft® Office Word. Plots were generated using Microsoft® Office Excel and Images were drawn using Microsoft® Office Visio® Professional.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

The feature sizes of a transistor on a Very Large Scale Integrated (VLSI) circuit are reducing by almost 10.5% every year, which results in an increase of transistor density by 22.1% [1]. Furthermore, an equal amount (around 22%) of increment in transistor density is provided by wafer and chip size improvements, along with other circuit design and process innovations [2]. This amounts to more than double the number of transistors on an Integrated Circuit (IC) every two years, which agrees closely with Moore's law of doubling the number of transistors on an IC every 18 months [3]. As the number of transistors on a chip increases, the probability that the chip contains at least one faulty transistor increases, which in turn increases the probability of the whole chip being faulty [1]. The defects which cause the faults may be due to impurities in the original silicon or in the manufacturing process [4]. As the feature size keeps decreasing, the defects that can occur during the fabrication process increase [5]. As the number of defects during the fabrication process increases, the importance of testing a chip increases. Currently, Field Programmable Gate Arrays (FPGAs) and System on Chips (SoCs) are among the devices with the highest transistor integration and are more prone to defects than other VLSI chips.

With the advent of VLSI technologies, the complexity and functionality of digital circuits increased dramatically, allowing more and more circuitry to be packed onto a

single chip. The increase in package density reduces the circuit costs, but increases the testing costs due to the reduction in controllability and observability (lack of access to the innermost resources) of the VLSI chips. Also, surface mount technology, in which components are mounted on both sides of the board, makes bed-of-nails testing either too expensive or unfeasible [6]. Bed-of-nails testers make contact with the solder joints of the Printed Circuit Board (PCB) to test the components soldered to it. The advantages of VLSI, namely reduced system cost, better performance, and greater reliability, are being offset by high system testing costs. Testing costs may be as high as 55% of the total cost of a complex IC and it is likely to increase proportionately with the increase in complexity of VLSI chips [7].

1.1 Overview of FPGAs

FPGAs are devices which can be programmed by a user [8]. The user can describe the circuit design using a Hardware Description Language (HDL) and it can be synthesized and implemented easily on an FPGA with just a computer and some cables. FPGAs have two important benefits: they have lower Non-Recurring Engineering costs and faster time to market compared to Application Specific Integrated Circuits (ASICs) [9].

The FPGA is an array of Programmable Logic Blocks (PLBs) with programmable routing resources used to interconnect PLBs and programmable Input/Output (I/O) cells [9], as shown in Figure 1.1. The programmability, nowadays, is mostly implemented by Static Random Access Memory (SRAM), requiring FPGAs to be configured (programmed) every time the circuit powers up [10]. A typical PLB in an FPGA has a

set of combinatorial and sequential logic resources that can be programmed to operate in various modes. The combinatorial logic part generally consists of RAM-based Look-Up Tables (LUTs) and the sequential logic part consists of latches and/or flip-flops [10]. PLBs generally incorporate three or four-input LUTs to implement three or four-input combinatorial logic functions, respectively. The small logic functions implemented by a PLB are expanded using the programmable routing or interconnect network to implement bigger logic functions.

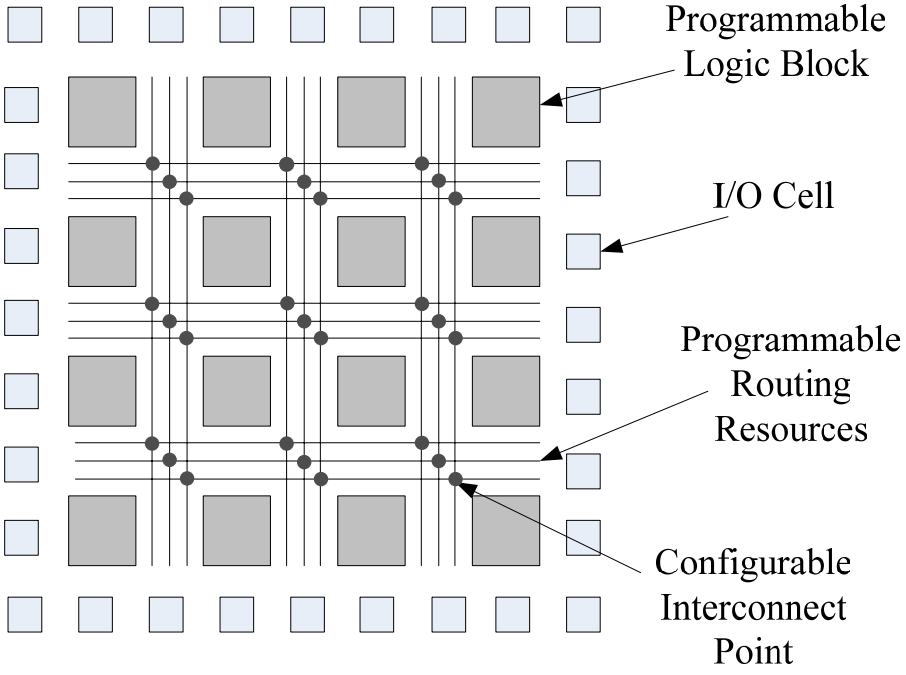


Figure 1.1: Basic FPGA Architecture

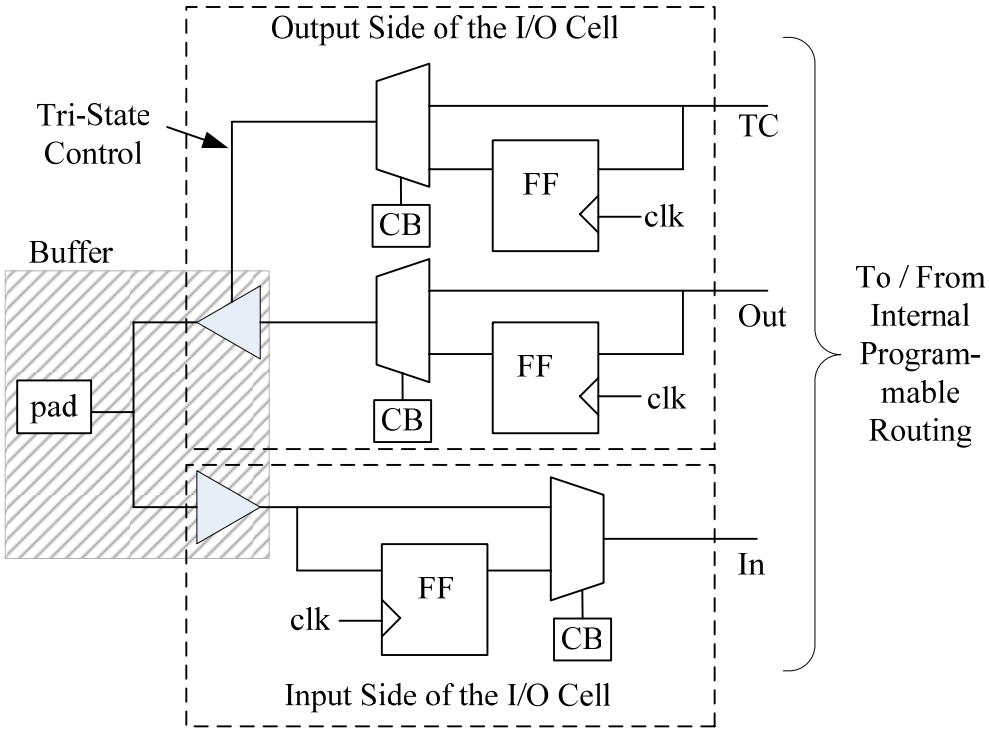
The programmable routing network is used to connect a PLB with other PLBs or with the I/O cells. The programmable routing consists of wire segments that can be connected or disconnected by Configurable Interconnect Points (CIPs) (also referred to as Programmable Interconnect Points, or PIPs) [11].

Reconfigurable cores like FPGAs are now being integrated into SoCs. The improvements in fabrication technology have created the ability to place all the system functions that were being placed on a single PCB onto a single chip, known as a SoC [12]. SoCs which have a reconfigurable core, like a FPGA, are called Configurable SoCs (CSoCs). Recently, FPGAs containing processor(s) are also being fabricated [13]. In CSoCs and also the FPGAs with processors, the processors have the capability to configure or reconfigure the FPGA [13]. CSoCs and FPGAs with processor cores have become highly popular because of the availability of the processor and also due to the advantages provided by reconfiguring an FPGA using the processor. All FPGAs and CSoCs communicate with other devices through programmable I/O cells to transmit and receive data. So, programmable I/O cells are an important part of any FPGA or SoC.

1.2 Overview of Programmable I/O Cell

The architecture of an example programmable I/O cell is shown in Figure 1.2. The programmable I/O cell consists of a bi-directional buffer, logic circuitry like flip-flops or multiplexers, and routing resources. The I/O buffer, shown as the shaded portion in Figure 1.2, constitutes the pad and some analog circuitry, like pull-up or pull-down transistors, delay elements and Schmitt trigger. The I/O buffers are used to communicate with other devices present on the PCB. The bonding pad is the interface between the die and the package [14]. A programmable buffer can be configured as an input buffer, which uses the input side of the I/O cell, as an output buffer, which uses the output side of an I/O cell, or as a bi-directional buffer, which uses both the output and input sides of the I/O cell. The tri-state signal is activated (put in a high impedance state) mainly when

the I/O cell is being used as an input cell. Configuration bits are used to program the logic and routing resources of an I/O cell. For example, the configuration bits, CBs shown in Figure 1.2, are used to activate registered or non-registered input/output by programming the multiplexer.



CB – Configuration Bit

Figure 1.2: Example of a Programmable I/O Cell

The I/O cells can be classified as bonded I/O cells and unbonded I/O cells. If the pads of the I/O cells are bonded to the pins of the package then the I/O cells are called bonded I/O cells; otherwise they are called unbonded I/O cells. The unbonded I/O cells cannot exchange information with the outside world [15]. For a given sized FPGA with a fixed number of I/O cells, different package sizes have different number of bonded and unbonded I/O cells. The buffer part of the I/O cell alters the characteristics of signals in

such a way that the signal characteristics become compatible with other devices connected to the chip when there is any information exchange between the chip and the outside world. As the performance of chips continues to increase, the I/O cells play a key role in maintaining high speed data transfer between packaged devices on a PCB or between different PCBs [16].

The I/O cells present in a FPGA are gradually increasing in number and also complexity as they integrate new functions [13]. As the number of functions in an I/O cell increases, the logic and routing resources associated with each I/O cell also increase and they become more prone to defects. So, testing of I/O cells is becoming an important issue.

1.3 Overview of the Prior Work in I/O Cell Testing

While a number of Built-In Self-Test (BIST) approaches have been developed for testing the programmable logic and routing resources in the FPGA core, they have neglected testing the I/O cells and the routing resources associated with them. BIST has been implemented to test the speed of I/O in [17]. Additional circuitry, including a Delay Locked Loop (DLL), test registers and comparators, was included for each register under test. This BIST circuit was developed for implementation in an ASIC to test the setup and hold time of the registers in the I/O cell. But other resources present in the I/O cell are not tested by this method [17].

A quiescent current-based (IDDQ) testing approach for I/O cells in an FPGA was proposed in [18]. In the steady state of CMOS circuits there is no direct connection between power supply (VDD) and ground, so the steady state current, or quiescent

current, should be zero. But the presence of defects can provide a path for flow of current, thus defects can be detected by measuring IDDQ. The technique proposed in [18] is an external test approach. In this technique the input and output sides of the I/O cell are tested separately. The input side of the I/O cell is tested by applying the inputs externally and the output side of the I/O cell is tested by monitoring the output signals externally, where the input side of one I/O cell and output side of another I/O cell are tested at the same time. Until now there has been no work on implementing BIST for programmable I/O cells.

1.4 Overview of Built-In Self-Test

Design-for-Testability (DFT) techniques were developed to keep the testing costs low. BIST is a DFT technique in which test pattern generation and output response analysis is done by on-chip circuitry [6]. The basic idea of BIST is to incorporate test circuitry along with the normal system circuitry to verify the proper functionality of the system. The BIST circuitry consists of a Test Patter Generator (TPG), which sources the test patterns to the Circuit Under Test (CUT), an Output Response Analyzer (ORA), which analyzes the output responses of the CUT, and a controller, which controls the test procedure [5]. BIST circuitry must be able to test the system quickly and should provide high fault coverage [1]. An example BIST architecture for a system is shown in Figure 1.3.

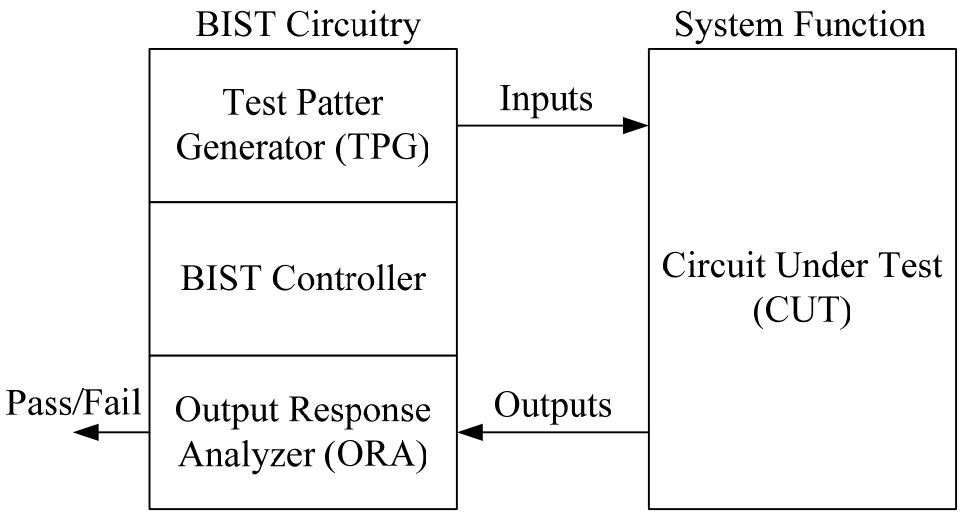


Figure 1.3: Basic BIST Architecture

The advantages of BIST are vertical testability, i.e., same test circuitry can be used at wafer level (before packaging) testing, device level (after packaging) testing and also at system-level testing, feasibility of at-speed test and often the overall test cost is reduced. The disadvantages of BIST include area overhead due to additional circuitry, longer design time and sometimes the fault coverage from BIST may be less than the fault coverage obtained from external tests [5]. The inclusion of BIST certainly increases the initial cost of the system. So, BIST feasibility for a system must be evaluated using benefit cost analysis, assessing the total life cycle costs compared to the initial cost [19].

Unlike conventional BIST, FPGA BIST does not include any area overhead or performance degradation. As an FPGA can be reconfigured any number of times, it is tested by reconfiguring logic and routing resources in different modes to test all the FPGA resources. The only requirement is additional memory for storing the configuration data to be written to the configuration memory of an FPGA to configure (program) it for BIST [19].

1.5 Thesis Statement

As the size of FPGAs (in terms of the number of transistors, the amount of logic that can be emulated and the number of I/O cells) increases, the problem of testing them also increases. Previously, BIST has been applied to test the PLBs and routing resources present inside an FPGA, but no work has been done in testing the logic and routing resources associated with the I/O cells of an FPGA using BIST. The goal of this thesis is to propose a BIST approach to test programmable I/O cells and to develop and implement BIST configurations for the Atmel FPGAs and SoCs, which can be used to test all the routing and logic resources associated with an I/O cell.

The thesis is organized as follows: In Chapter 2, the general architecture of FPGAs and I/O cells, as well as the architecture of Atmel FPGAs and SoCs, is described along with the previous work in BIST for FPGAs and I/O cell testing. In Chapter 3 the BIST architectures used for testing the I/O cells of Atmel FPGAs will be described. In Chapter 4, the use of an embedded processor present in the Atmel SoCs will be described for BIST execution and partial reconfiguration, which reduces the test time. Finally, the thesis will be summarized and concluded in the Chapter 5, describing the major contributions and possible future work. A list of acronyms used in this thesis is included in Appendix A.

Chapter 2

Background

This chapter begins with a general architectural description of FPGAs and programmable I/O cells. Then the architecture of the Atmel AT94K Field Programmable System Level Integrated Circuits (FPSLIC) and their resources is described. This chapter also describes the prior work in I/O cell testing and BIST architectures for testing logic, routing and RAM cores in FPGAs. Finally, this chapter concludes with a re-statement of the thesis goals.

2.1 General Overview of FPGAs

Programmability in FPGAs can be implemented with anti-fuses, Erasable Programmable Read-Only Memories (EPROM), Electrically Erasable PROMs (EEPROM) or SRAMs [20]. Most of the current FPGAs on the market use SRAM technology. SRAM cells are volatile, so the FPGA has to be configured each time the power is supplied. Programming an SRAM-based FPGA consists of writing bits into the SRAM to connect the required wire segments and define logic functions. The bits written into the SRAM to configure the FPGA are called *configuration bits* and SRAM memory which stores the *configuration bits* is called the *configuration memory*.

From this point on, the architecture of the SRAM-based FPGAs is discussed. FPGAs consist of an array of PLBs interconnected using a programmable routing

network and is surrounded by programmable I/O cells. The architectures of a typical PLB, routing network and I/O cell are discussed in next sections.

2.1.1 Programmable Logic Blocks

PLBs generally consist of LUTs, D-type flip-flop(s), multiplexers and other logic resources. The basic architecture of an example PLB in its simplest form is shown in Figure 2.1. The implementation of a two input LUT is shown in Figure 2.2a, where the configuration memory can contain any combination of 1's and 0's. As each configuration bit can contain a logic '1' or '0', using 4 bits a total of $2^4 = 16$ logic combinations can be formed using a two input LUT. The implementation of a LUT for a two input OR gate is shown in Figure 2.2b.

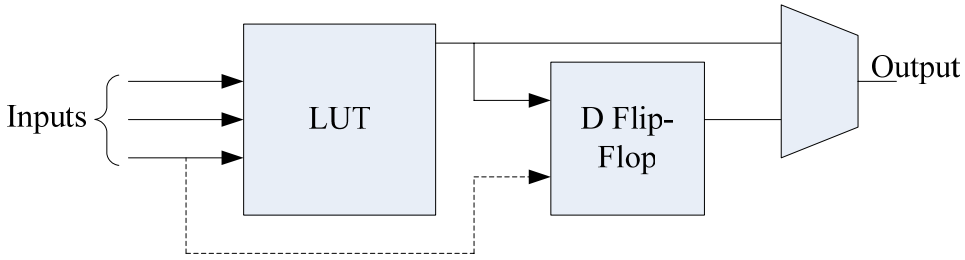
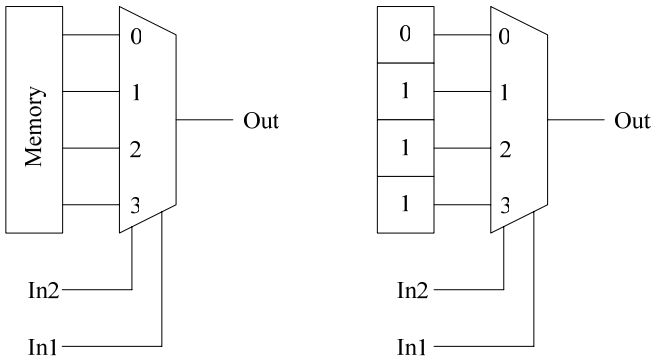


Figure 2.1:General Architecture of a Programmable Logic Block

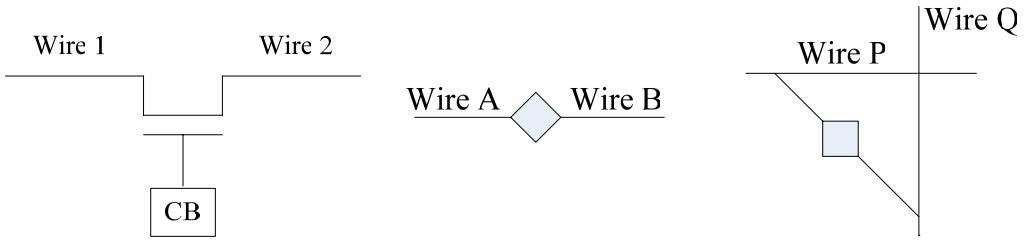


(a) General (b) OR Gate

Figure 2.2: LUT Implementation

2.1.2 Routing in FPGAs

The programmable interconnect network consists of wire segments that can be connected or disconnected by CIPs [5]. The basic CIP structure consists of a transmission gate controlled by a configuration memory bit, as shown in Figure 2.3a. Depending on the logic value of the configuration memory bit, the two wire segments may be connected or disconnected. There are four basic types of CIPs - cross point CIP, break point CIP, multiplexer CIP and compound CIP [5]. A break point CIP connects two wire segments which are in the same plane, as shown in Figure 2.3b. A cross point CIP is used to connect two wire segments which are in different planes. For example, a horizontal Wire P can be connected to a vertical Wire Q using a cross point CIP, where Wire P has to be above or below Wire Q, so they are in different planes. Multiplexer CIPs select one of the several inputs and connect it to the output. A compound CIP is a combination of four cross point CIPs and two break point CIPs [5]. The flexibility in routing a design can be improved by increasing the number of interconnects. So, modern FPGAs include large amounts of routing resources to improve the flexibility in implementing a design.



(a) General Structure (b) Break Point CIP (c) Cross Point CIP

Figure 2.3: Configurable Interconnect Point [5]

2.1.3 Programmable I/O Cells

The example architecture of an I/O cell was shown in Figure 1.2 and the introduction was given in Section 1.2. In this section a detailed discussion of the resources in the I/O cells will be presented. To transmit the signals correctly, a Direct Current (DC) output signal should supply enough voltage, current, power or energy to drive the loads connected to it. To receive the input signal, the voltage level should be interpreted correctly [22]. These output and input signal properties are generally managed by the I/O buffer.

The general programmable features of an I/O buffer are:

Output drive capability: Usually the output drive current can be programmed to different values and its value is generally chosen on the basis of power dissipation and loading considerations.

Pull-up/Pull-down: On activating either the programmable pull-up or pull-down transistor, a weak logic value of '1' or '0' will be supplied to the pad when all the other drivers are off. This helps in maintaining known voltage levels when the output is tri-stated [15].

I/O standards: The logic voltage threshold level of the pad can be set to be compatible with I/O standards like TTL or CMOS or with any of the available I/O standards.

Schmitt Trigger: Schmitt trigger is a regenerative comparator that adds hysteresis to the incoming signal, which improves its rise and fall times. It also helps in filtering out the noise and removing the glitches due to switch bounces [22]. The Schmitt trigger circuit is generally present on the input side of the I/O buffer.

Delay: The input signal can be programmed to have different intrinsic delays [15][24]. The delay helps in meeting the hold time requirements of the incoming signal.

Tri-State: Most I/O buffers have a programmable tri-state select signal on the output side of the I/O cell. Having a tri-state control signal allows using the same pad for input and output so that the tri-state is always activated (put in a high impedance state) when used as an input buffer. The tri-state select signal can be activated or deactivated permanently, or can be controlled by an internal logic signal. Two different implementations of tri-state buffer are shown in Figure 2.4 [23]. When the I/O cell is acting as input cell, the output enable is made low and data is taken from the pad through 'Data In'. When acting as an output cell, the output enable is made high and depending on the value of 'Data Out', the pad is connected to either VDD or ground.

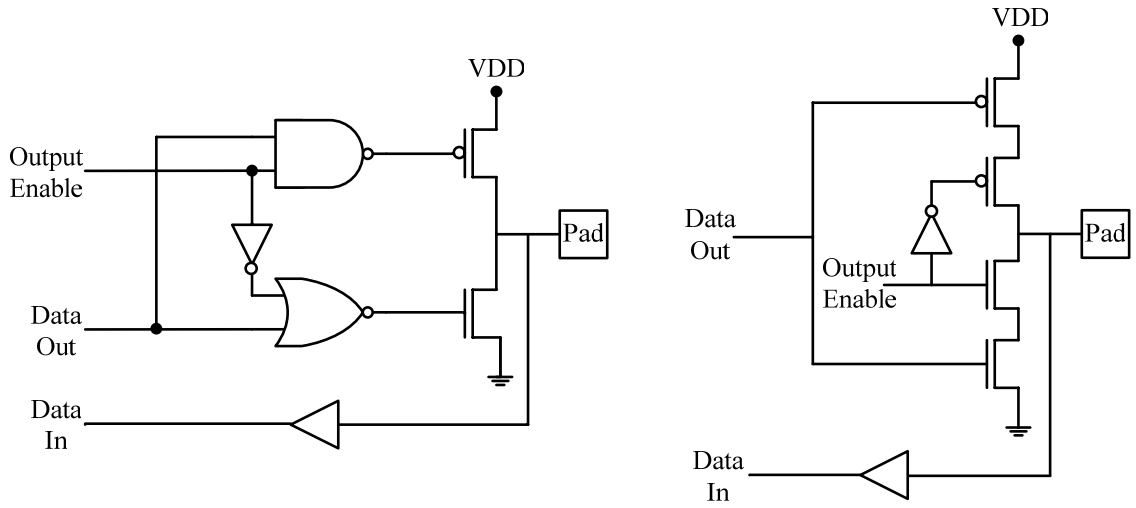


Figure 2.4: Implementation of Tri-state Buffer

These are some of the important programmable features of an I/O buffer present in most of the FPGAs. Other than the buffer, the I/O cell has logic components like flip-flops for providing registered inputs and outputs, multiplexers for selecting signals and

inverters to invert the signal values. The logic resources associated with the I/O cells are increasing continuously as newer architectures are being developed to allow high speed data transfer. The Xilinx Virtex-4 FPGA has 32 multiplexers and 10 flip-flops in its I/O cell [24]. Table 2.1 shows the gradual increase in the number of I/O cells and registers in the I/O cells in different architectures of Atmel and Xilinx FPGAs.

Table 2.1: I/O Cell Resources in Atmel and Xilinx FPGAs/SoCs

FPGA/Soc	Number of Registers per I/O Cell	Year	Maximum Number of I/O Cells	Device
Atmel AT40K	0	1999	384	AT40K40
Atmel AT94K40	2	2002	288	AT94K40
Xilinx Virtex	3	2002	512	XCV1000
Xilinx Virtex E	3	2002	804	XCV3200E
Xilinx Virtex II PRO	6	2003	1164	XC2VP70/100/ XC2VPX70
Xilinx Spartan 3	6	2005	784	XC3S5000
Xilinx Virtex 4	10	2005	960	XC4VLX200
Xilinx Virtex 5	10	2006	1200	XC5VLX330

For the PLBs to communicate with the I/O cells in the FPGA, some dedicated routing resources are associated with the I/O cells in every FPGA. But the routing architecture is dependent on the FPGA manufacturer and it has considerable variation from one manufacturer to another. Xilinx FPGAs use an array of compound cross-point CIPs for routing whereas Atmel FPGAs use multiplexer CIPs which will be discussed in section 2.2.1.

2.2 Atmel AT94K Architecture

Atmel AT94K series devices are called FPSLICs and they are a family of configurable SoCs. They have an AT40K SRAM-based FPGA core, an 8-bit Advanced Virtual Reduced Instruction Set Computer (RISC) processor core, referred to as the AVR, and some RAM cores [15]. The architecture of the FPGA and AVR cores in the AT94K FPSLICs will be described in this section.

2.2.1 FPGA Core Architecture

The FPGA core has an *N*x*N* array of identical PLBs arranged in a symmetrical fashion. The value of $N = 24$ for AT94K10 series FPGAs and $N = 48$ for AT94K40 series FPGAs. The FPGA core has bonded I/O cells on three sides and the fourth side has the interface with the AVR microprocessor, along with some unbonded I/O cells [15]. The architecture of the FPGA core for an *8*x*8* PLB array is shown in Figure 2.5. A block consisting of a *4*x*4* array of PLBs with a set of horizontal and vertical repeaters and a RAM core is repeated over the FPGA architecture. Each PLB consists of two 3-input LUTs, which can be combined using a multiplexer to implement any 4-input

combinational logic function, a D flip-flop and some multiplexers to provide routing and logic flexibility [15]. A part of the PLB is shown in Figure 2.6.

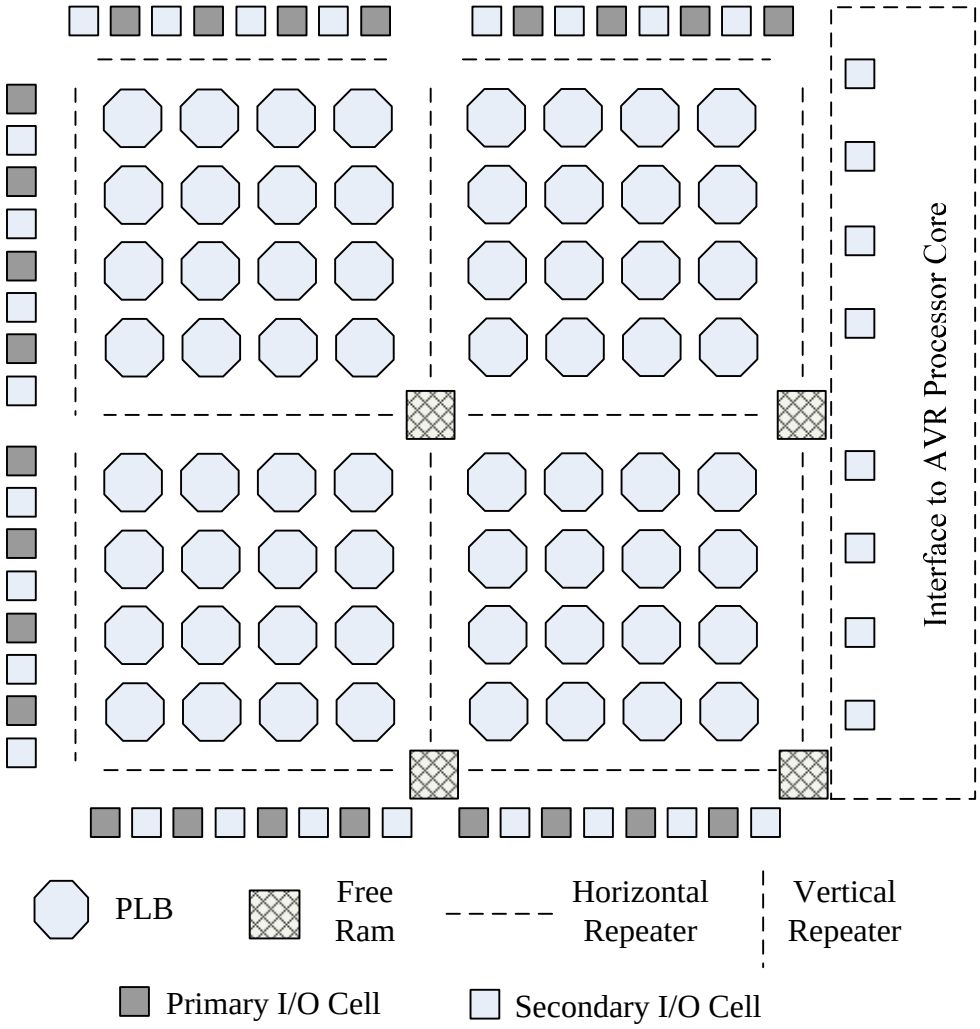


Figure 2.5: Architecture of AT94K FPSLIC

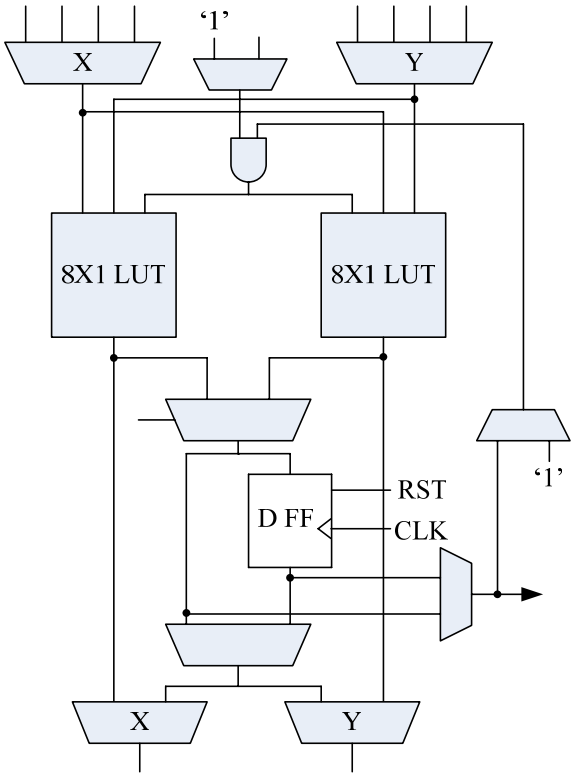
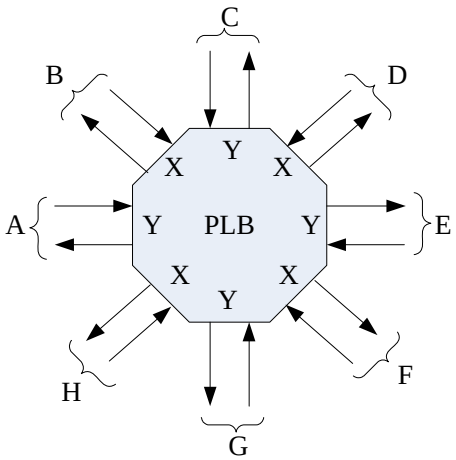


Figure 2.6: AT94K PLB

Every PLB has direct routing connections with all other surrounding PLBs and the PLBs located at the edges have direct connections with the I/O cells. The direct routing connections of a PLB with surrounding PLBs are shown in Figure 2.7.



A, E – Direct connections to/from horizontally adjacent PLBs

C, G – Direct connections to/from vertically adjacent PLBs

B, D, F, H – Direct connections to/from diagonally adjacent PLBs

Figure 2.7: Direct Routing Connections for a PLB

All the PLBs at the periphery have three sets of direct connections with the I/O cells. For example, a PLB which is located at the bottom edge of the FPGA, the direct connections F, G and H would be connected to or from I/O cells. In Figure 2.7, the horizontal and vertical connections of a PLB are denoted as 'Y' and the diagonal connections are denoted as 'X'. Along with these dedicated local connections, the inputs to the PLB can also be sourced from global routing connections.

The global routing connections are present around every PLB. Figure 2.8 shows the global routing connections associated with a single PLB. There are five horizontal and vertical bussing planes and each plane has two express buses and one local bus. The local buses span a length of 4 PLBs, whereas the express buses span a length of 8 PLBs [15]. The LUTs can get inputs from any of the local buses of the routing resources or from the direct routing connections of the adjacent PLBs. The output of the PLB can be sent onto any local bus or to an adjacent PLB by using global routing resources and direct connections, respectively.

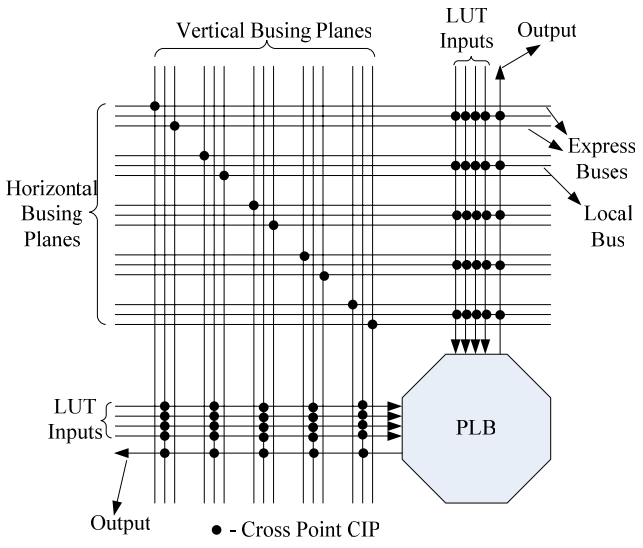


Figure 2.8: Global Routing Resources Associated with a PLB

As the local and express buses span a length of 4 and 8 PLBs, the logic signals are buffered using the repeaters. The repeaters are also used to provide routing flexibility. A horizontal repeater with possible interconnections is shown in Figure 2.9. A repeater makes connections between express and local buses. The repeater internally consists of four 3-input multiplexer CIPs to enable connections between any two buses. Using the repeater, any input to it can be connected to the output of the other three lines. It can also be used to make more than one connection between the buses if there is no conflict in the utilization of the multiplexer resources inside the repeater [15].

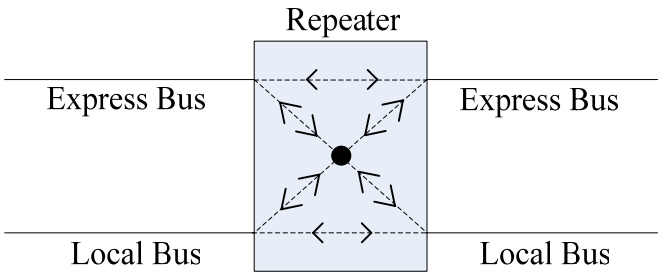


Figure 2.9: Simplified View of a Repeater

2.2.2 Data and Program Memory

A 36KByte SRAM memory core is partitioned into data memory and program memory. The data memory can be accessed by both FPGA and AVR, whereas the program memory can be accessed only by AVR. There is a 20KByte fixed program memory, a 4KByte fixed data memory, and the remaining 12KBytes is partitioned into three 4KByte blocks which can be configured to be used as either program memory or data memory. The AVR instructions to be fetched during execution are stored in the program memory [15].

The interface between the FPGA core, data SRAM core and the embedded AVR microcontroller is shown in Figure 2.10 [15][21]. The FPGA core can be directly accessed by the AVR core. Up to 16 decoded address lines are available from the AVR to the FPGA interface. Also there are 16 interrupts from the FPGA to the AVR with different priority levels. The FPGAIORE and FPGAIOWE signals are activated for one AVR clock cycle whenever data is read from the AVR data bus or written to the AVR data bus, respectively [15].

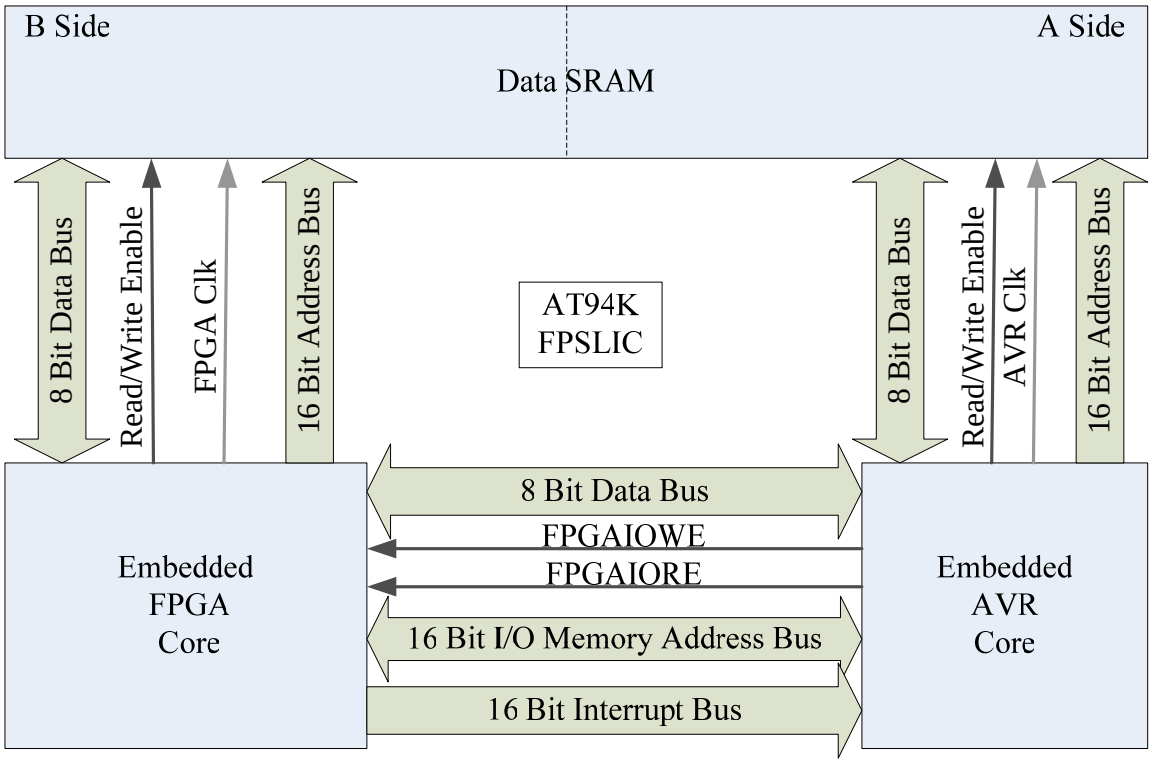


Figure 2.10: AVR, FPGA and SRAM Interface

2.2.3 Architecture of the Embedded AVR Microcontroller

The embedded AVR microcontroller is an 8-bit RISC architecture based microcontroller. The AVR uses Harvard architecture and has separate memories and

buses for program and data. The program memory has a single level pipeline such that next instruction can be fetched from memory while an instruction is being executed. During interrupts or subroutine calls, the return address of the program counter is stored in a stack, which is also a part of data SRAM. As a result, the size of the stack is limited by the size of the data SRAM [15].

2.2.4 AVR Write to FPGA Configuration Memory

The AVR is capable of writing to the configuration memory of the FPGA. There is an 8-bit data bus which is used by the AVR to write to the configuration memory of the FPGA core. The AVR can do dynamic full or partial reconfiguration of the FPGA without any loss of data [15]. The registers *FPGAX*, *FPGAY* and *FPGAZ* control the address of the configuration SRAM to which the data is to be written and *FPGAD* controls the data to be written. Each register is 8-bits wide and the configuration memory is byte addressable. The *FPGAX* and *FPGAY* describe the horizontal and vertical co-ordinates of the resources in the FPGA and *FPGAZ* describes the resources of the FPGA to be reconfigured like PLB, RAM, I/O cell or routing resources [15]. Figure 2.11 shows the FPGA access scheme. The Atmel FPSLIC devices do not have the capability for the AVR to read the configuration memory [15]. As the configuration memory is not bit addressable and as each byte may contain combination of bits related to different resources in the FPGA, the lack of configuration memory read back will become a stumbling block when only specific resources are need to be reconfigured (partial reconfiguration) without affecting the configuration of other resources.

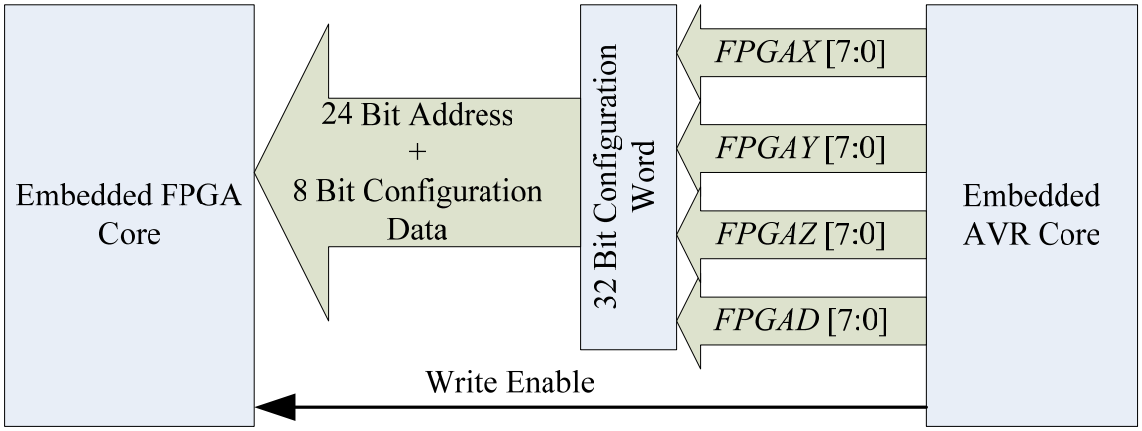


Figure 2.11: Internal FPGA Configuration Access

2.2.5 Architecture of Atmel AT94K I/O Cells

In this section, the architecture, characteristics and special features of the Atmel I/O cells, which are the target of the BIST approach in this thesis, are discussed. In Atmel FPGAs the I/O cells are mainly classified into two types, namely primary I/O cells and secondary I/O cells, based on their position with respect to the nearest PLB. In Figure 2.12, I/O1, I/O3 and I/O5 are primary I/O cells and I/O2, I/O4 and I/O6 are secondary I/O cells. Each primary I/O cell has direct access to one PLB and each secondary I/O cell has direct access to two PLBs, whereas the connections for the I/O cells at the corners are slightly different. The connections between PLBs and I/O cells shown in Figure 2.12 are direct connections. The corner I/O cells have access to only one PLB as shown for I/O2 cell in Figure 2.12. The corner I/O cells are almost the same as primary or secondary I/O cells, the only difference being the number of direct connections for secondary I/O cells [15].

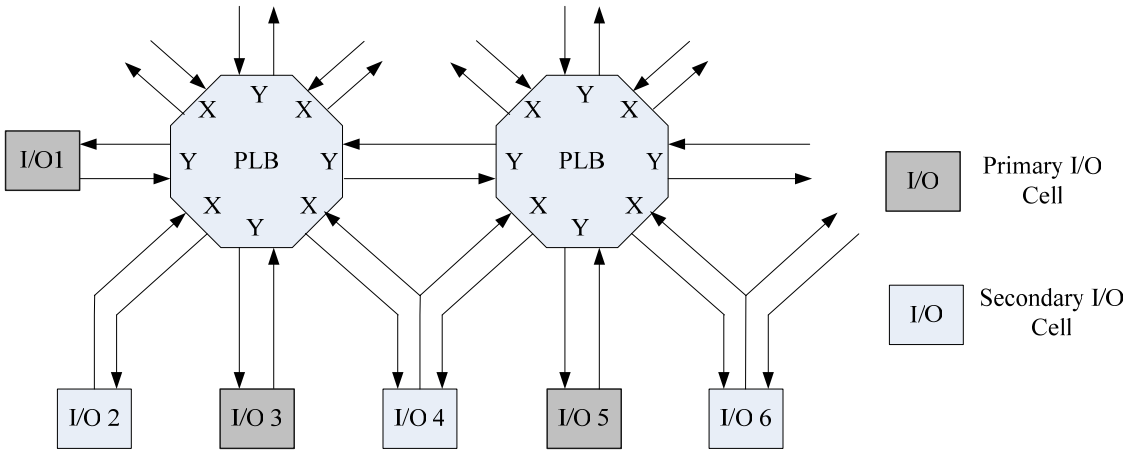


Figure 2.12: Location of Primary and Secondary I/O cells

2.2.5.1 Resources in I/O Cell

The output drive capability of the buffer can be programmed to be fast, medium or slow in terms of the slew rates of the I/O buffer, where slew rate is the rate of change in the output voltage [14]. The Fast slew rate option produces a drive capability of 20mA, medium produces 14mA and slow produces 6mA of drive current. The I/O buffer has programmable pull-up and pull-down transistors. The input threshold level can be programmed to be compatible with either TTL or CMOS. A Schmitt trigger circuit can be enabled on the input side. Also, the input buffer can be programmed to have four different intrinsic delays of approximately 0ns, 1ns, 3ns and 5ns. These are the programmable resources present in the I/O buffer [15].

The logic resources in the I/O cell consist of two positive-edge-triggered D flip-flops with asynchronous reset signal. The asynchronous reset signal can be programmed to be active high or active low and is sourced from an I/O cell. All the I/O cells have a break point CIP connection to the global reset (GRST in Figures 2.13 and 2.15), so any I/O cell can be configured as a global reset input pin. The flip-flops are present only on

the input and output data signals and the tri-state control signal does not have any flip-flop to provide a registered value. Each I/O cell has four multiplexers. The data multiplexer selects the data to be sent to the I/O pad and the tri-state multiplexer selects the control signal to the I/O buffer. The other two multiplexers select either registered or non-registered data at the input and output of the I/O cell. A total of 144 and 288 general purpose I/O cells are present in AT94K10 and AT94K40 FPSLICs, respectively. The number of I/O pads connected to the package pins varies from package to package as shown in Table 2.1. In addition to the logic resources, the I/O cells have transmission gates to provide routing flexibility. These are the general features of I/O cells in the Atmel FPGA core [15]. The logic and routing resources of the unbonded I/O cells are quite often utilized by the FPGA Computer Aided Design (CAD) tools for very dense designs.

Table 2.2: Number of Package Pins (Bonded I/O Cells) in Different Packages

Device	Maximum Number of I/O Cells	Package			
		AJ	AQ	BQ	DQ
AT94K10	144	46	58	84	116
AT94K40	280	Not Available	Not Available	84	120

2.2.5.2 Primary I/O Cells

The primary I/O cells, shown in Figure 2.13, are located orthogonal to the PLBs at the periphery of the FPGA core. They are present on three sides of the FPGA core, other than the side which interfaces to the AVR. Each primary I/O cell has seven inputs

to the data multiplexer and eight inputs to the tri-state multiplexer as shown in Figure 2.13. Of the seven inputs to the data multiplexer, IOOD is a direct connection from the orthogonal PLB, IOOC is clock-wise connection from global routing, IOOCC is counter clock-wise connection from global routing, IOOX is connected from the express bus of the global routing and IOOL is connected from the local bus of the global routing resources. Two other inputs, hardwired to logic '1' and '0', provide constant values at the output of the I/O cell. The same seven inputs are connected to the tri-state multiplexer as well and their naming convention is shown in Figure 2.13. In addition to these inputs, the tri-state multiplexer has a banked tri-state input which is common for a set of eight adjacent I/O cells, of which four are primary and four are secondary [15].

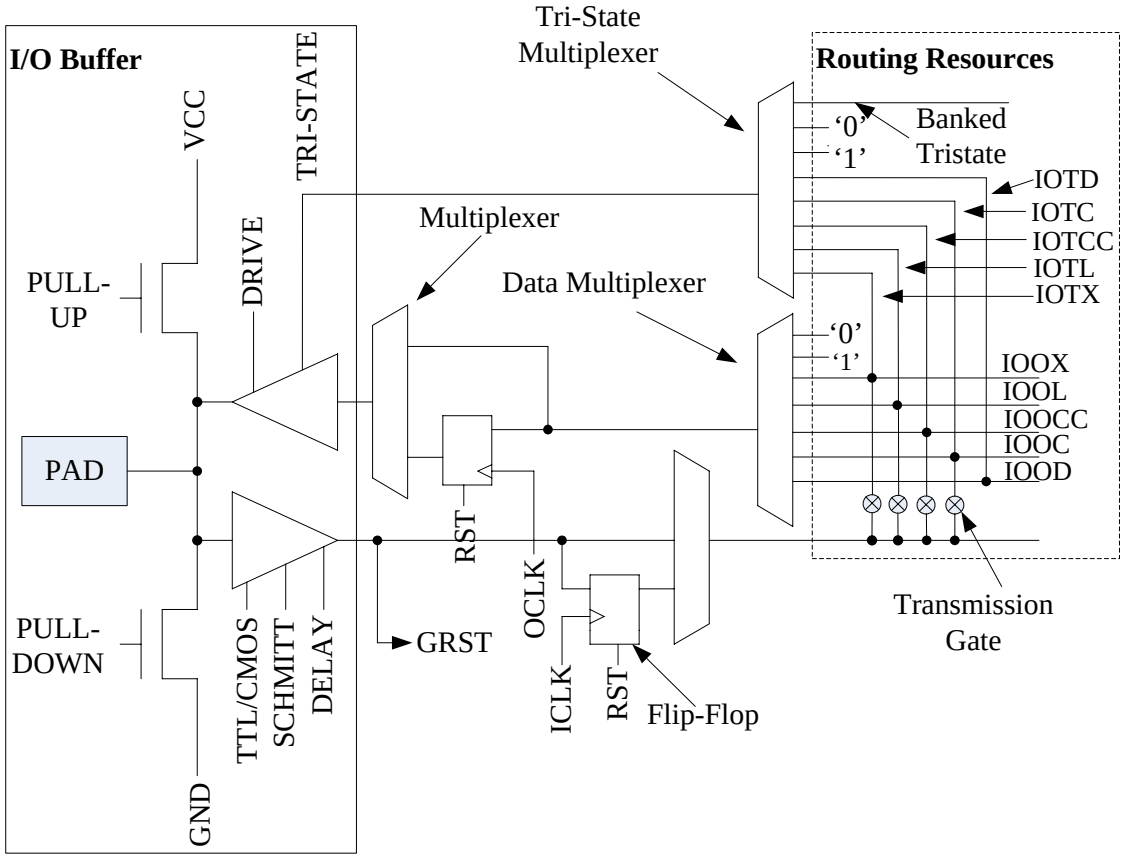


Figure 2.13: Primary I/O Cell of ATMEL AT94K FPSLIC Devices

The four inputs to the multiplexers that come from the global routing resources of the FPGA have transmission gates connected to them as shown in Figure 2.13, with their actual implementation shown in Figure 2.14. The transmission gates are mainly used to provide routing flexibility when the I/O cells are configured as input cells. The input from the I/O cell can be directly connected to a PLB at the periphery using the direct connection. To make connections with other PLBs, the transmission gates are used to route the input signals to the global routing resources of the FPGA. The two paths 'A' and 'B' that can be used to route the input signals to the routing resources are shown in Figure 2.14.

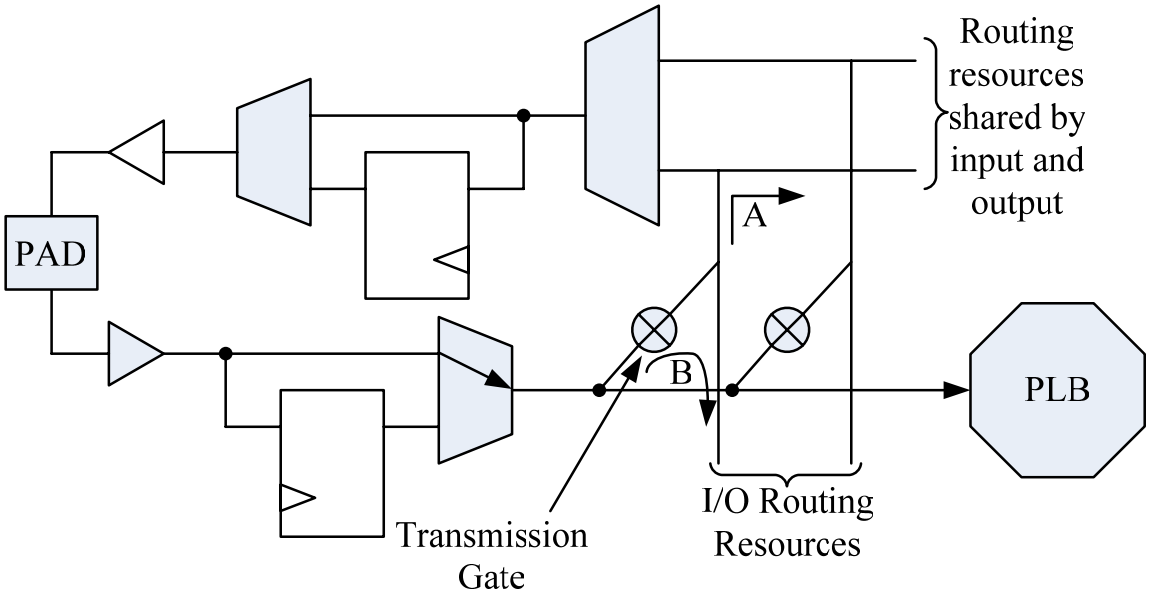


Figure 2.14: I/O Cell Configured as Input

2.2.5.3 Secondary I/O Cells

The secondary I/O cells, shown in Figure 2.12, are located diagonally to all the PLBs present at the periphery of the FPGA core adjacent to the primary I/O cells. The secondary I/O cells are also present on the side in which the AVR core is interfaced to the

FPGA core, all of these being unbonded I/O cells without flip-flops. The secondary I/O cells have six inputs to the data multiplexer and seven inputs to the tri-state multiplexer as shown in Figure 2.14. Of the six inputs, two are direct connections from two diagonal PLBs, two are from the global routing resources, and the other two are the hardwired '1' and '0'. The additional input present to the tri-state multiplexer is the banked tri-state signal which is same as the one for a primary I/O cell. Similar to the primary I/O cell, the two connections to the global routing resources have transmission gates [15].

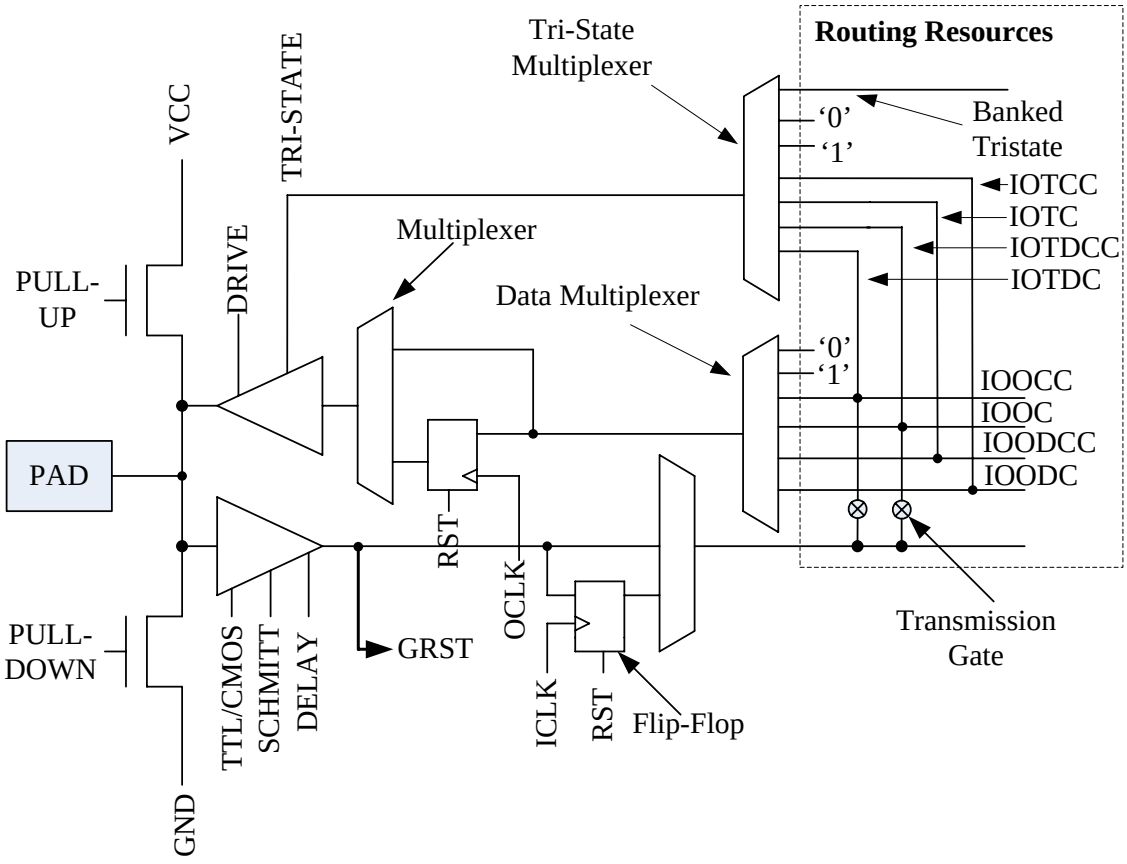


Figure 2.15: Secondary I/O Cell of Atmel AT94K FPSLIC Devices

2.2.5.4 Clock I/O Cells

Along with the primary and secondary I/O cells, clock I/O cells are also present to connect the clock input to the FPGA. There are six global clock I/O cells and all of them

are present in the corners of the FPGA. Depending on their location with respect to the primary or secondary I/O cells, they have the architectures of secondary or primary I/O cells, respectively. The differences between clock I/O cells and general I/O cells (primary and secondary) are that the clock I/O cells do not have any flip-flops and instead of a GRST connection to the global reset network, they have connection to the global clock network.

2.2.6 Macro Generation Language

Macro Generation Language (MGL) is a high level programming language specially designed by Atmel to allow the users to create their own design. MGL can be used to implement designs only on the Atmel FPGAs. MGL has the ability to place and route designs, which is not supported by other HDLs. Designs described using MGL can be edited, compiled and debugged using Figaro software provided by Atmel [25]. MGL can implement parameterized designs, so designs developed for smaller FPGAs can be easily extended to be implemented in larger FPGAs.

Using MGL, the routing interconnections can be specified exactly for every path and the signals can be routed to unconnected inputs of the multiplexers as well. MGL gives access to unbonded I/O cells so that the unbonded I/O cells can be activated and their logic resources can be utilized. Using MGL, the functionality of each PLB can be specified exactly. All the above mentioned features are not supported by other HDLs.

2.3 Previous Work in I/O Cell Testing

The prior work presented in Section 1.3 will be discussed in more detail in this section. In [17], a technique to test the setup and hold time of the flip-flops in the I/O cells is proposed. In this technique, a DLL is used to generate two clocks named *setup* and *hold* clocks with fixed delays with respect to the master clock. Two additional test flip-flops are present for each flip-flop in the I/O cell. The *setup* and *hold* clocks are applied to the two test flip-flops and the same input is applied to the test flip-flops and the flip-flop of the I/O cell. The data captured in the flip-flop of the I/O cell is compared with the data captured in the test flip-flops to test for setup and hold times of the flip-flops in the I/O cell. The additional circuitry used for this BIST technique consists of two flip-flops and two comparators for every flip-flop under test and a DLL to generate the *setup* and *hold* clocks. By using additional buffers to drive the setup and hold clocks generated, the same DLL can be used to test the flip-flops in all I/O cells. The high area overhead makes this suitable only for specific high performance circuits.

In the IDDQ test approach presented in [18], the I/O cells and the routing resources associated with the I/O cells are tested by configuring half of the I/O cells as input cells and the rest as output cells, as shown in Figure 2.16, in the first test phase. When configured as an input cell, the input side of the I/O cell is tested by externally applying test patterns and the results are externally monitored through another output cell, testing the output side of that I/O cell at the same time. In the second phase the roles of the input and output I/O cells are reversed. In two phases, both the input and output sides of the I/O cells are tested. Since the number of input signals is limited, extra test signals for testing the resources of the output cell like tri-state and clock enable are

generated using the internal resources. These tests are useful only for the manufacturing testing, since the resources associated with the unbonded I/O cells cannot be tested after packaging. So, the test approach presented in [18] is not a BIST approach and unbonded I/O cells can be tested only during the manufacturing tests. Also to test the packaged FPGAs, the tests have to be developed separately for each package.

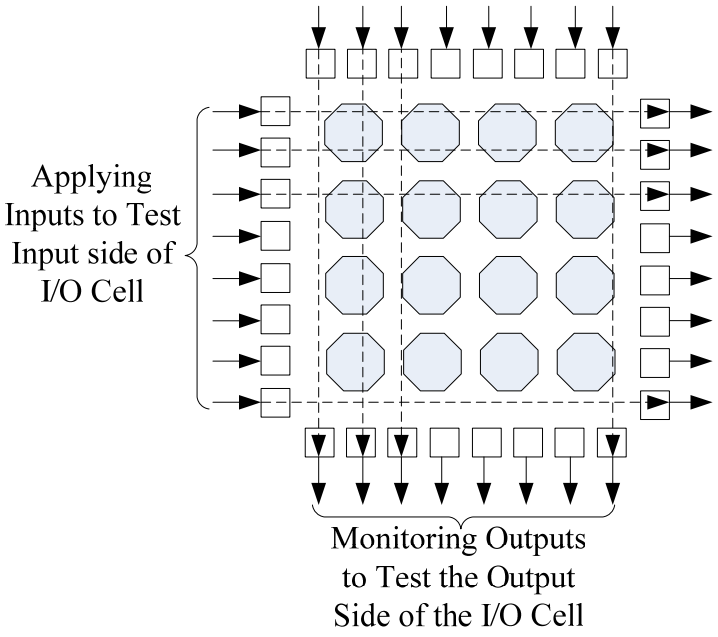


Figure 2.16: External Test Approach to Test I/O Cells of an FPGA

In [26] also, the same external test approach as used in [18] to configure some I/O cells as input cells and the others as output cells to perform the test has been described. In this paper, a technique is proposed to detect the stuck-at faults in the routing resources associated with the I/O cell. The routing resources are considered to be switch matrix based and the test techniques were proposed specifically. As this is also an external test technique, it also has the same disadvantages as described for [18].

2.4 BIST for FPGAs

In this section the general BIST techniques to test PLBs and routing resources of the FPGA cores will be described.

2.4.1 BIST for PLBs

The PLBs in FPGAs generally have a regular array of *M*x*N* blocks, *M* and *N* being even numbers in most cases. To test the PLBs, they are divided into 3 groups: TPGs, ORAs and Blocks Under Test (BUTs) as shown in Figure 2.17. The TPG may be either a counter or a Linear Feedback Shift Register (LFSR) and the structure of the ORAs will be discussed shortly. The PLBs configured as BUTs are tested with test patterns being supplied by two identical TPGs and the results being analyzed by ORAs [19]. Each TPG supplies test patterns to alternate column of BUTs. After performing the test in a particular BUT configuration, the BUTs are then reconfigured to be tested in a different mode. BUTs are repeatedly reconfigured until the logic resources are tested completely [5]. Each reconfiguration of the FPGA to test a different mode of PLB operation is referred to as a test phase. A test session is a collection of test phases that completely test the BUTs in all possible modes of operation. Once the BUTs are completely tested, the BIST architecture is flipped to reverse the roles of the PLBs as shown in Test Session 2 of Figure 2.17. So the PLBs previously configured as TPGs and ORAs now function as BUTs and the previous BUTs function as TPGs and ORAs [5]. After testing the PLBs in one test session, the results are retrieved before going to the next test session. The ORA results can be extracted by reading the contents of the ORA flip-flops from the configuration memory, if configuration memory readback is

supported, or by connecting the ORAs in the form of a scan chain as shown in Figure 2.17 to scan out the results through the FPGA I/O [19][27]. Some of the previous logic BIST approaches were described in [19][27] and the logic BIST for the embedded FPGA core in Atmel FPSLICs has been described in.[28]

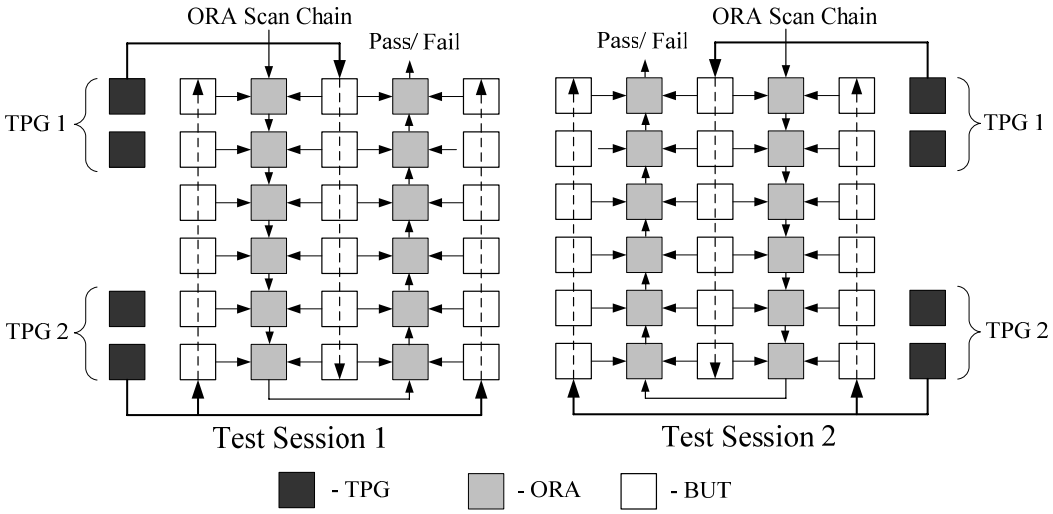


Figure 2.17: BIST Architectures of PLBs

To implement a comparison-based ORA with scan chain, a total of five inputs are required as shown in Figure 2.18. Since a PLB in Atmel has only four inputs, the ORA shown in Figure 2.18 would require two PLBs and this would increase the total number of configurations required to test all the PLBs. To implement an ORA with single PLB, the structure of the ORA is changed as shown in Figure 2.19. In the normal test mode the ORA is configured as shown in Figure 2.19a. The OR gate and the flip-flop combined will latch up any mismatches between the two inputs from BUTs. After performing the test, dynamic partial reconfiguration is performed such that the ORA architecture is changed as shown in Figure 2.19b [29]. When Shift Control is at logic ‘0’, the output of

the flip-flop is fed back to the input and ORA results are retained. When Shift Control is made logic '1', the ORAs are connected in the form of a scan chain and the results are scanned out.

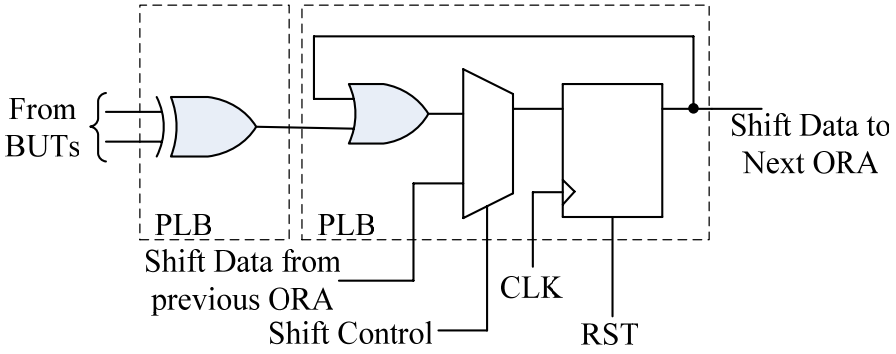
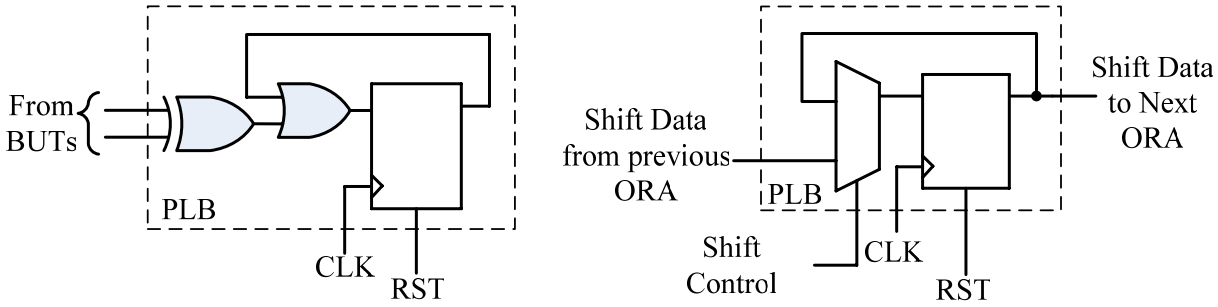


Figure 2.18: Comparison Based ORA with Scan Chain



(a) Test Mode (b) Scan Mode

Figure 2.19: ORA used for Logic BIST

2.4.2 Routing BIST

In routing BIST, a subset of routing resources is divided into two sets and the test patterns generated from the TPG PLBs are routed on those two sets of wires under test (WUTs). The outputs of the two sets of wires are analyzed by PLBs which are configured as comparison-based ORAs [30]. The sample setup for two similar sets of wires under test is shown in Figure 2.20. The TPG would be an *M*-bit counter and the value of *M* depends on the number of wires under test.

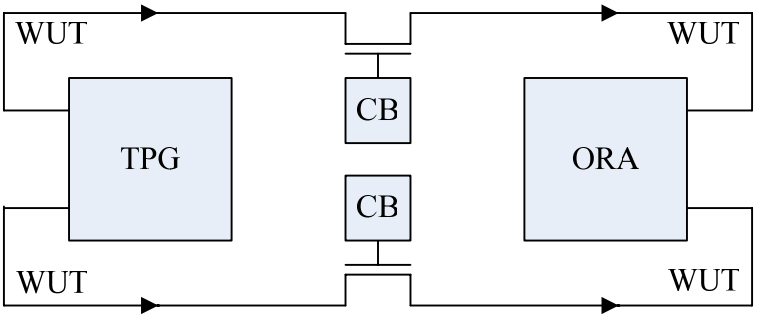


Figure 2.20: Routing BIST architecture

An alternate parity-based routing BIST approach is described in [31]. In this approach, instead of routing the same set of test patterns on two sets of wires, the TPG generates a parity bit which would be transmitted on a single wire. Now the ORA has to be changed to parity-based ORA, which would have a parity decoding circuit and a comparator [31]. Some of the previous routing BIST approaches were described in [30][31][32] and the routing BIST approach for the embedded FPGA core in Atmel FPSLICs has been described in [28].

2.4.3 BIST for RAM Cores

In the logic BIST architecture shown in Figure 2.17, the BUTs in the middle column are compared by two ORAs whereas the BUTs at the edges are compared by only one ORA. The comparison of a BUT by a single ORA causes a loss in diagnostic resolution. To test the RAMs in Atmel FPGAs there are sufficient PLBs to implement a two PLB based ORA with scan. So, the ORA in Figure 2.18 has been used in BIST for RAMs without any need for partial reconfiguration [21]. The output of the first BUT is compared with the output of the last BUT, this implementation is known as circular comparison. A diagnostic procedure based on multiple faulty cell locator

(MULTICELLO) has already been developed in [19]. The algorithm has been extended to diagnose multiple faulty PLBs when circular comparison based BIST is implemented. Overall, circular comparison provides better fault diagnosis [33].

2.5 Thesis Re-statement

The increase in the logic resources of the programmable I/O cell and the importance of the I/O cell as the interface for the FPGA core are the main reasons for developing BIST for I/O cells. The implementation of I/O cell BIST should also provide near 100% fault coverage for the FPGA as the BIST approaches to test the other resources in a FPGA have already been proposed. In this thesis, a general BIST approach to test the logic and routing resources associated with the programmable I/O cells of any FPGA will be described. The BIST approach and its implementation on the I/O cells of the Atmel AT94K FPSLICs is described in Chapter 3. After general implementation, the use of the embedded AVR microcontroller in reducing the test time by performing dynamic partial reconfiguration will be described. The implementation results of the proposed BIST approach for Atmel AT94K FPSLIC devices, along with the limitations of the approach will also be described in Chapter 4. The proposed BIST approaches can be used for manufacturing testing as well as for system-level testing by the end user without any additional test equipment.

Chapter 3

BIST for I/O Cells

This chapter discusses the BIST architectures used to test the logic and routing resources associated with the I/O cells in the Atmel AT94K FPSLICs. First, a general BIST architecture that can be applied to test the logic and routing resources of programmable I/O cells in any FPGA will be discussed. Then the BIST architectures used to test the resources specific to Atmel FPGAs will be discussed. Finally the number of test configurations and the test time of BIST for I/O cells is compared with the BIST for logic, routing and RAMs.

3.1 BIST Architecture

Figure 3.1 shows the I/O cell BIST architecture. This general BIST architecture can be applied to test the programmable I/O cells of any FPGA. The basic approach used in implementing the BIST for I/O cells in the FPGAs is:

- Some of the PLBs are configured as a TPG and some of them are configured as comparison-based ORAs.
- The I/O cells are configured as bidirectional cells by activating both the input and output sides of the I/O cells.
- The test patterns are supplied by the PLBs configured as a TPG to the output side of the I/O cell.

- The output responses of the I/O cells are looped back into the FPGA from the pad through the input side.
- The output responses of the I/O cells are analyzed by comparing with the responses of other identically configured I/O cells, forming a circular comparison.

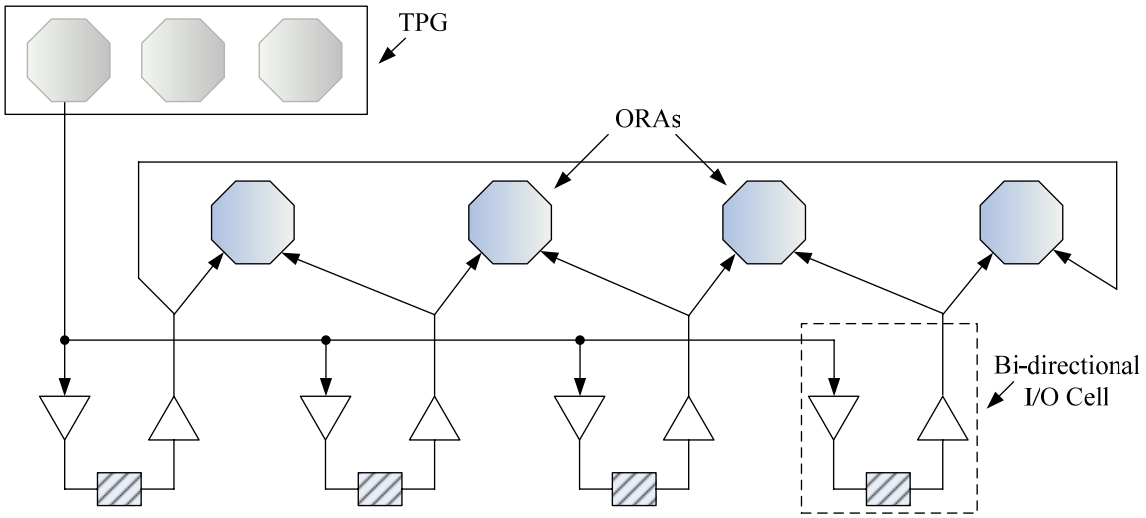


Figure 3.1: I/O BIST Architecture

This BIST architecture enables the testing of all bonded and un-bonded I/O cells, so it is package independent. The PLBs and their associated routing resources present in the FPGA are assumed to be already tested with the logic and routing BIST approaches [28], so a single TPG can be used to source the test patterns, as it is supposed to be fault-free. The response of every I/O cell is compared with the responses of two other I/O cells, thus circular comparison is achieved. If there are *N* I/O cells under test, then there would be *N* ORAs for circular comparison, whereas there would only be *N-1* ORAs for a non-circular comparison based approach.

For Atmel FPGAs the maximum number of routing resources common to the data and tri-state (excluding banked input) multiplexers is five for primary I/O cells. So, the

TPG is designed to be a 6-bit counter with the five Least Significant Bits (LSBs) of the counter being applied as inputs to the data and tri-state multiplexers, shown in Figures 2.13 and 2.15. The Most Significant Bit (MSB) of the counter is used to drive the reset signal to the flip-flops of the I/O cell. The I/O cells are repeatedly configured in various modes of operation, such as activating pull-up or pull-down, selecting different multiplexer inputs, selecting registered or non-registered inputs and outputs, etc. Similarly, the routing resources associated with the I/O cells are also tested by repeated reconfiguration of the connections made from the TPG to the I/O cells and from the I/O cells to the ORAs. The minimum number of BIST configurations required to test the I/O cells and the associated routing resources is usually a function of the number of possible signal paths to the output side of the I/O cell. For Atmel FPGAs, the minimum number of configurations is determined by the number of inputs to the largest multiplexer in the I/O cell. Since the tri-state multiplexer has the highest number of inputs it will determine the minimum number of configurations required.

In Atmel FPGAs, dedicated X and Y connections are present between the PLBs on the periphery of the FPGA and the I/O cells. So, the PLBs at the periphery are just used for routing the TPG and ORA signals, as shown in Figures 3.3 and 3.6. For every PLB in the periphery there are three I/O cells, since each PLB can be associated with a primary and two secondary I/O cells. For a FPGA with a 24x24 array of PLBs, 48 I/O cells are present on all three sides and 24 I/O cells on the side interfaced with the AVR. Each PLB can have a maximum of three outputs, one X, one Y and one to local routing resources, shown in Figure 2.8. If two outputs are used to route the TPG signals to the primary and secondary I/O cells through the dedicated routing resources, the responses of

both the I/O cells cannot be routed back through the dedicated routing resources associated with the same PLB. This limitation in the number of outputs from a PLB requires the primary and the secondary I/O cells to be tested in separate configurations. The BIST approaches for primary and the secondary I/O cells are discussed in the next two sub sections.

3.1.1 BIST for Primary I/O Cells

The architecture of the primary I/O cells was described in Section 2.2.5.2. The signals from the TPG are routed to all the inputs of the data and tri-state multiplexers. Each primary I/O cell has four inputs from the routing resources and those resources are shared by two adjacent primary and secondary I/O cells as shown in Figure 3.2. All the inputs from the routing resources have to pass through repeaters at the edge to reach the I/O cells. The TPG outputs are named from T1 through T6, where T1 is the LSB of the counter and T6 is the MSB. When the same input of the multiplexer is selected in the two adjacent I/O cells, they have different TPG signals activated. In Figure 3.4, if the first input of the data multiplexer is activated then TPG signals T4, T1, T4, T1, and so on are selected by the multiplexers in adjacent primary I/O cells. As a result, all the alternate I/O cells are compared with each other instead of being compared with adjacent I/O cells. If adjacent ORAs are compared then T4 would be compared with T1, which would always cause a mismatch. So, ORAs are divided into two circular comparison loops, comparing alternate I/O cells.

The test patterns are routed in the same manner to all the primary I/O cells as shown in Figures 3.2 and 3.3. The PLBs at the periphery of the FPGA are used only for

routing the TPG and ORA signals, using the direct connections between PLBs and I/O cells. The routing connections of the periphery PLBs with the primary I/O cells are shown in Figure 3.3. Consider two inputs IOOC (clock-wise connection) and IOOCC (counter clock-wise connection), the input which behaves as the clock-wise connection for I/O cells, named as IOOC, present on the South becomes a counter clock-wise connection for the I/O cells present on North, named as IOOCC. Even though same signal is selected by the multiplexers present on North and South, they are sourced different test patterns on each side. For example, consider two multiplexers of I/O cells which are routed in the same manner on North and South sides, if the IOOC is selected by both the multiplexers, the multiplexer on the South may activate the T3 signal whereas the multiplexer on the North may activate T2. So, the responses from the I/O cells of different sides will be different, even though the same input IOOC has been selected by the multiplexers. Since the ORAs on one side of the FPGA cannot be compared with the ORAs on the other side, individual circular comparison loops are formed by the ORAs for I/O cells on each side. Two ORA circular comparison loops are present on each side as shown in Figure 3.5.

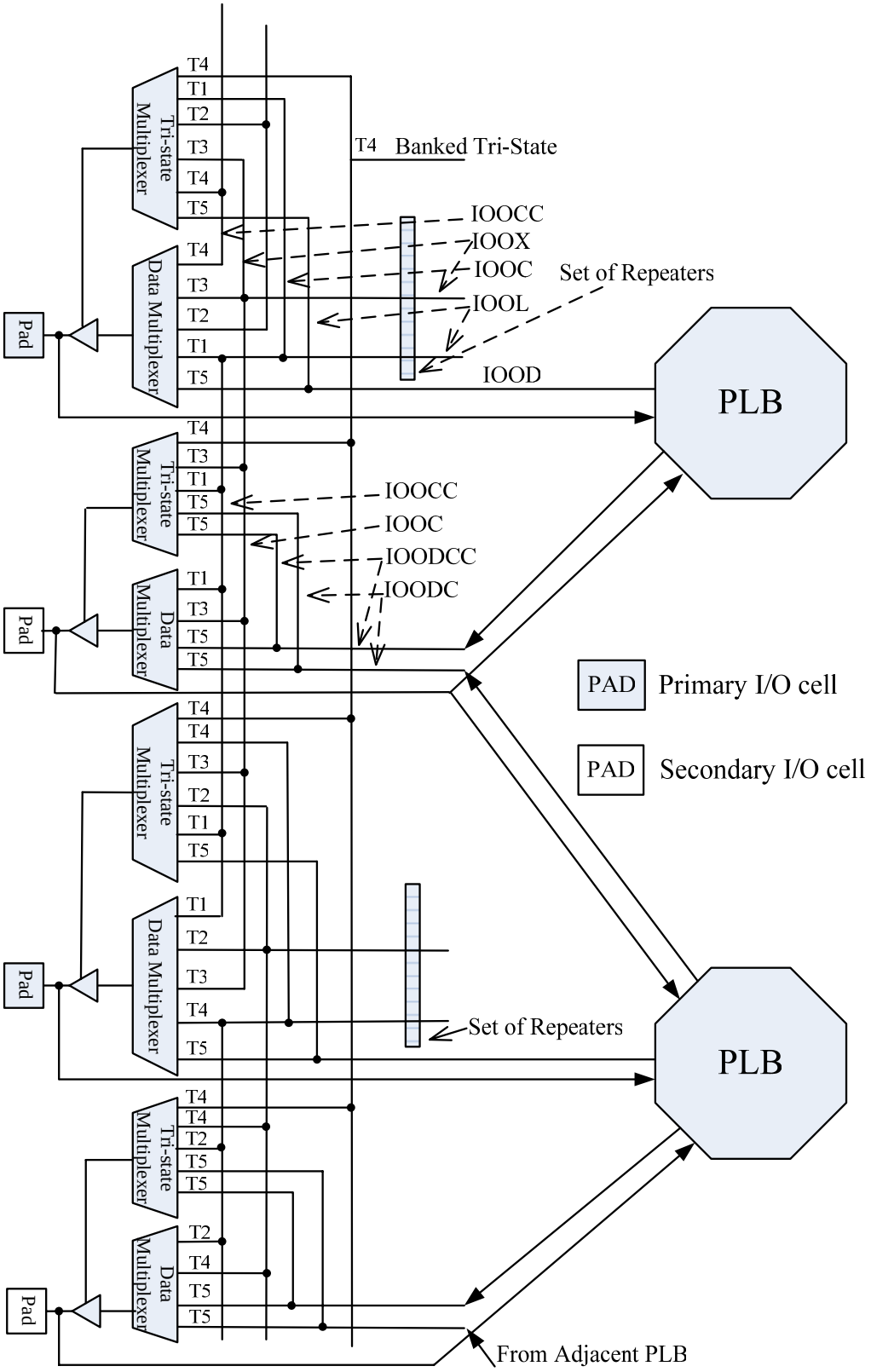


Figure 3.2: Routing Interconnections between the Primary and Secondary I/O Cells

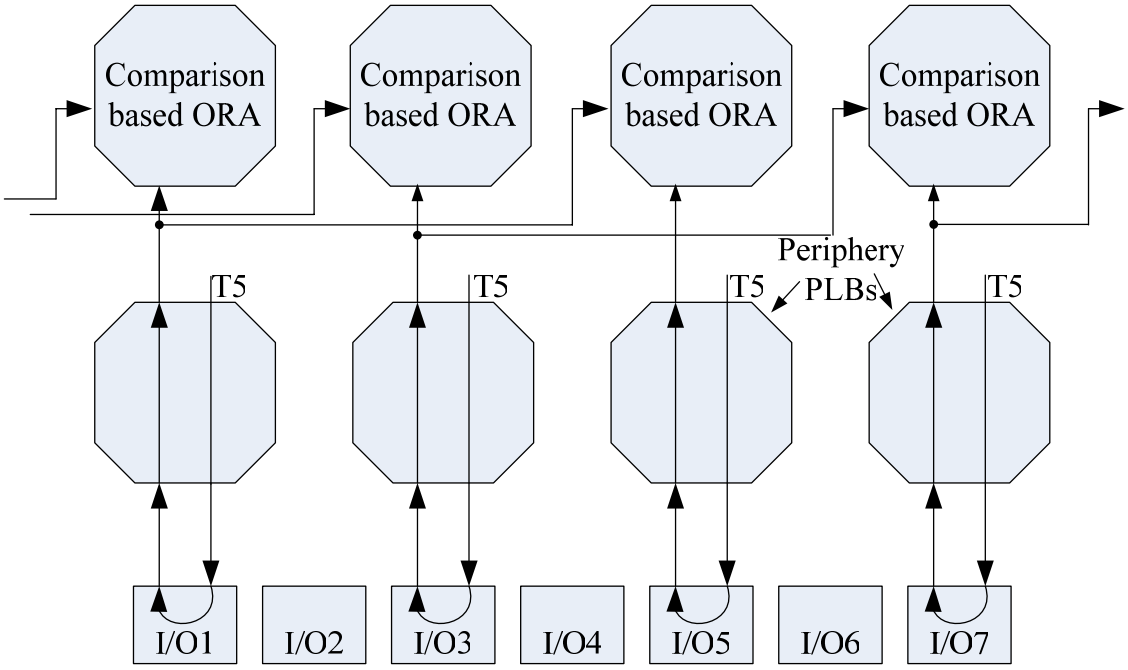


Figure 3.3: Direct Routing Connections from PLBs to Primary I/O cells

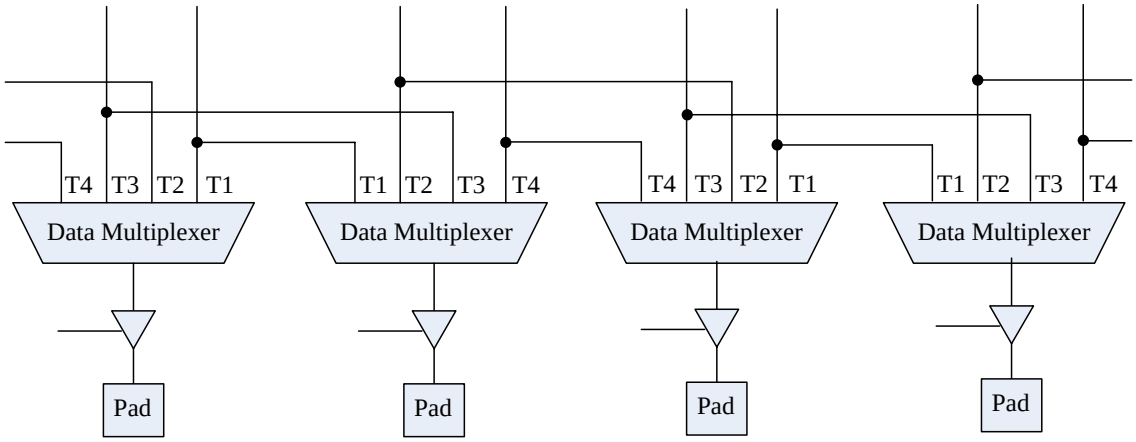


Figure 3.4: Routing Interconnections between the Primary I/O Cells

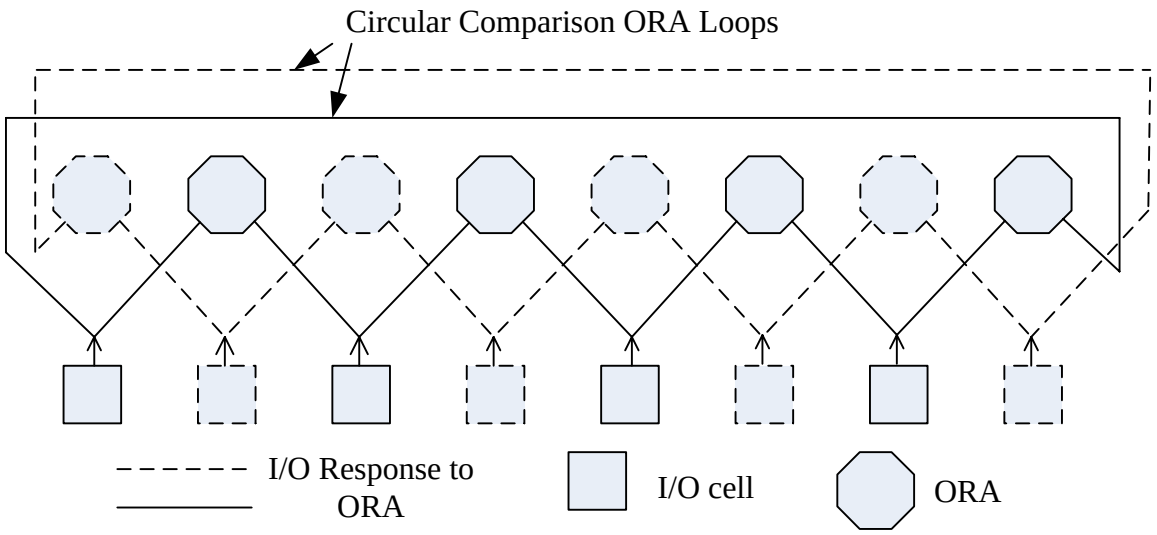


Figure 3.5: ORA Loops on Each Side

The TPG signals are routed to all the inputs of the data and tri-state multiplexers of primary I/O cells. The first four bits of the TPG are routed through the global routing resources as shown in Figure 3.4. The fifth bit of the TPG, T5, is routed through the direct input connection from the PLB to the multiplexer, shown in Figure 3.3, and the sixth bit of the TPG, T6, is routed to the reset signal of the flip-flops. If an additional output (such as T7) is routed to the banked tri-state input, there will be routing contentions with the already routed TPG signals (T1 to T4). As a result, TPG input T4 was routed to the banked tri-state input to remove the routing contentions.

The tri-state multiplexer consists of eight inputs, and a total of nine configurations are required to test all the multiplexer inputs of both data and tri-state multiplexers. If we exclude the hardwired ‘0’ of both the multiplexers, the remaining seven inputs of the tri-state multiplexer and six inputs of the data multiplexer can be tested in seven configurations. Separate configurations are needed to test hard wired ‘0’ inputs of the tri-state and data multiplexers, as testing a hard wired ‘0’ of one multiplexer would block

any fault affect on the other multiplexer output. So, a total of nine configurations are required to test the multiplexer inputs of the primary I/O cells. In the same nine configurations, the logic resources associated with the I/O cell are also tested by configuring them in all possible modes. The resources tested in each configuration are summarized in Table 3.1. Note that most of the logic resources get tested more than once as they can be tested in fewer than nine configurations.

Table 3.1: Configuration Modes of Primary I/O Cells

Config-urations	Data Mux	Tri-State Mux	Delay (ns)	Drive (mA)	I/O Flip-Flop	Pup/ Pdown	TTL/ CMOS	Schmitt
1	IOOX	IOTCC	2	6	Both	Pup	TTL	Active
2	IOOC	IOTL	5	14	Input	Pdown	CMOS	Inactive
3	IOOCC	IOTX	8	20	Output	Pup	CMOS	Inactive
4	IOOL	IOTC	0	6	None	Pdown	CMOS	Active
5	IOO1	IOTD	2	14	None	Pup	TTL	Inactive
6	IOO0	IOTD	5	20	None	Pup	TTL	Active
7	IOOCC	IOT0	8	6	None	Pdown	TTL	Active
8	IOOD	IOTB	0	14	None	Pdown	CMOS	Inactive
9	IOTD	IOT1	2	20	Both	Pup	TTL	Active

3.1.2 Secondary I/O cells

The architecture of the secondary I/O cells was described in Section 2.2.5.2. As TPG patterns are routed to all the inputs of the primary multiplexer, they get routed to the multiplexer inputs of the secondary I/O cell as shown in Figure 3.6, since the routes are

being shared with primary I/O cells. The sharing of the routing resources between the primary and secondary I/O cells is shown in Figure 3.2. Secondary I/O cells have two inputs from the global routing resources and two are direct connections from PLBs. As the TPG patterns are already routed to all the multiplexer inputs of the secondary I/O cell, the routing scheme of T1-T4 TPG signals for the secondary I/O cells is kept same as that of the primary I/O cells. For secondary I/O cells also, the PLBs at the periphery are just used for routing the TPG and ORA signals as shown in Figure 3.6. For secondary I/O cells also, two ORA loops also have been formed due to the regularity in multiplexer inputs, on each side of the FPGA as shown in Figure 3.7.

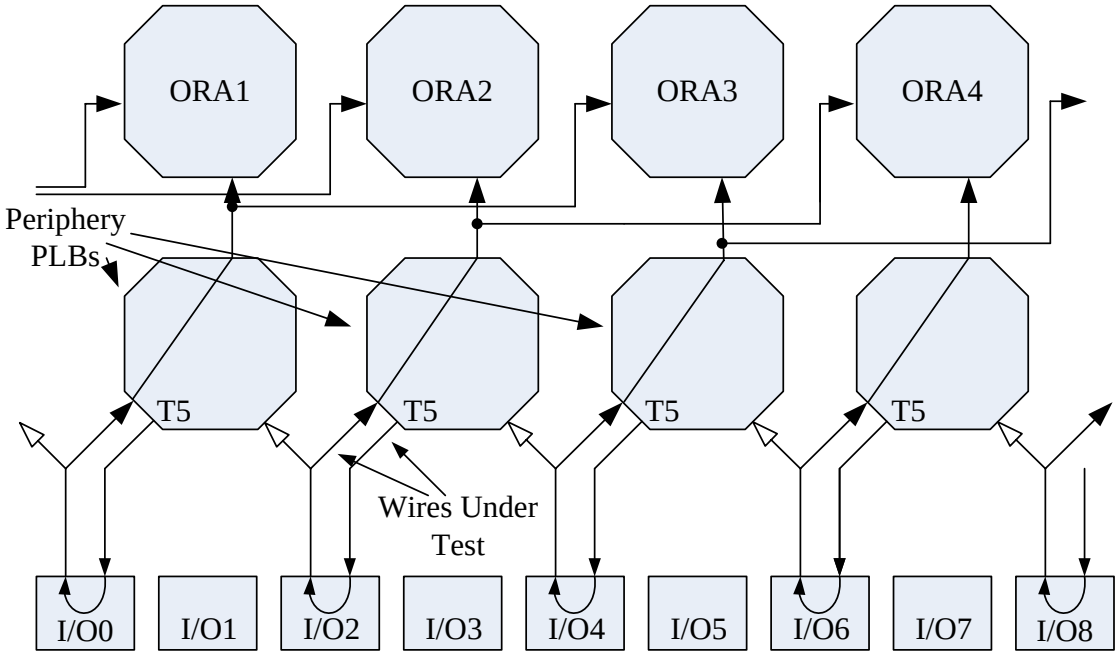


Figure 3.6: Direct Routing Connections from PLBs to Secondary I/O cells

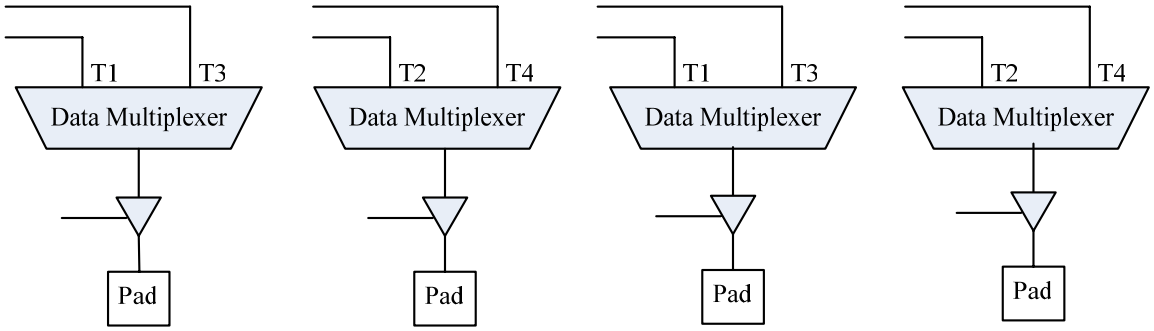


Figure 3.7: Routing Connections to the Secondary I/O Cells

The TPG signals are routed to all the inputs of the data and tri-state multiplexers of the secondary I/O cells. Two of the first four TPG bits are routed through the routing resources as shown in Figure 3.2. The fifth bit of the TPG, T5, is routed through the direct input connection from the PLBs to the multiplexer, shown in Figure 3.6, and the sixth bit of the TPG, T6, is routed to the reset signal of the flip-flops. As the banked tri-state is same for the set of four primary and secondary I/O cells, the banked tri-state input for secondary I/O cells is also connected to T4. The tri-state and data multiplexers consist of seven and six inputs, respectively, one input less than that of primary multiplexers. So, a total of eight configurations are required to test all the inputs (compared to nine for the primary). The resources tested in each configuration are shown in Table 3.2. In the same eight configurations, the logic resources associated with the I/O cell are also tested by configuring in all possible modes. In one of the eight configurations, the other direct connection to the PLB is also tested.

Table 3.2: Configuration Modes of Secondary I/O Cells

Config-urations	Data Mux	Tri-State Mux	Delay (ns)	Drive (mA)	I/O Flip-Flop	Pup/ Pdown	TTL/ CMOS	Schmitt
1	IOODCC	IOTCC	2	6	Both	Pup	TTL	Active
2	IOOC	IOTDCC	5	14	Input	Pdown	CMOS	Inactive
3	IOOCC	IOTDC	8	20	Output	Pup	CMOS	Inactive
4	IOO1	IOTC	0	6	None	Pdown	CMOS	Active
5	IOO0	IOTC	2	14	None	Pup	TTL	Inactive
6	IOODC	IOTB	5	20	None	Pdown	TTL	Active
7	IOOC	IOT0	8	6	None	Pdown	TTL	Active
8	IOODC	IOOT1	0	14	Both	Pdown	CMOS	Inactive

3.1.3 Testing Transmission Gates

The I/O cells have transmission gates that allow the input and output portions of the I/O cell to share the same programmable routing resources. Four transmission gates are associated with the primary I/O cells and two are associated with the secondary I/O cells. Two types of faults, known as stuck-on and stuck-off, are associated with the transmission gates. A Stuck-on fault means a gate is always turned on and stuck-off fault means it is always turned off, irrespective of its configuration bits. These transmission gates have already been tested for stuck-on faults while the multiplexer inputs are tested. If the transmission gates are stuck-on, they back drive the TPG inputs when the flip-flops are activated as shown in Figure 3.8. The output response of one I/O cell can back drive the TPG signal of an adjacent I/O cell as well.

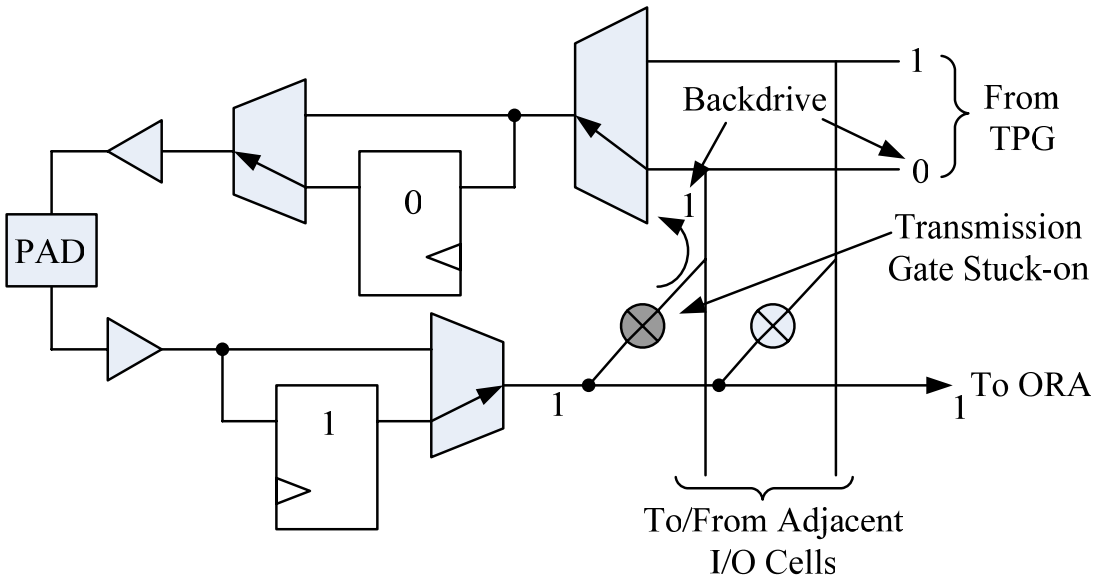


Figure 3.8: Transmission Gate Stuck-on Test

A slightly different approach is used to test the transmission gates for stuck-off faults to reduce the complexity of routing from the output of the input buffer to the ORA using different transmission gates. To route to the ORA using the transmission gates and paths ***a*** or ***b*** shown in Figure 3.9 is complex, so the feedback loop path and the direct connection to the PLB are used. When the configuration is downloaded into the FPGA, the flip-flops in the I/O cells are automatically initialized to complementing values with the input flip-flop initialized to '1'. In this approach there is no TPG, the flip-flops in the I/O cells themselves act as the TPG. The test configuration is as shown in Figure 3.9, a feedback loop is created between the input and output portions of the I/O cell. Now, as the flip-flops are clocked, the values stored in them will be toggled, as shown in Figure 3.10, and the output of the flip-flop present on the input side of the I/O cell will be monitored by the ORA. The flip-flops are clocked for few cycles to test for the stuck-off fault of the transmission gate. To test a different transmission gate, a new configuration

has to be downloaded. The first configuration of this approach is also generated by manipulating the bitstream generated by MGL. The actual implemented way of testing transmission gates is discussed in Chapter 4.

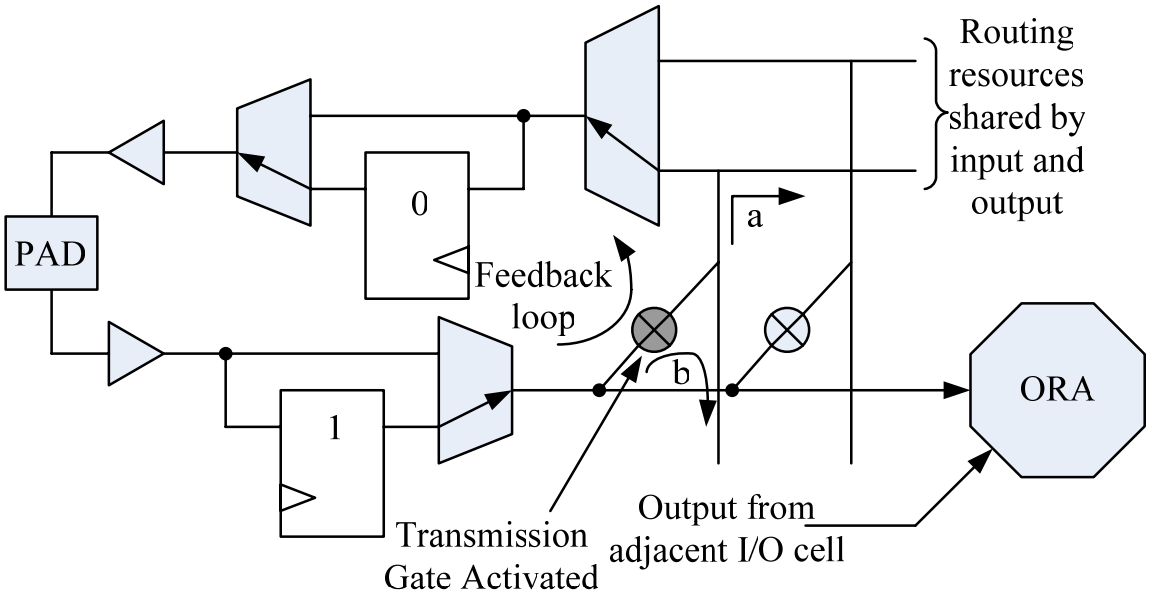


Figure 3.9: Transmission Gate Stuck-off BIST Configuration

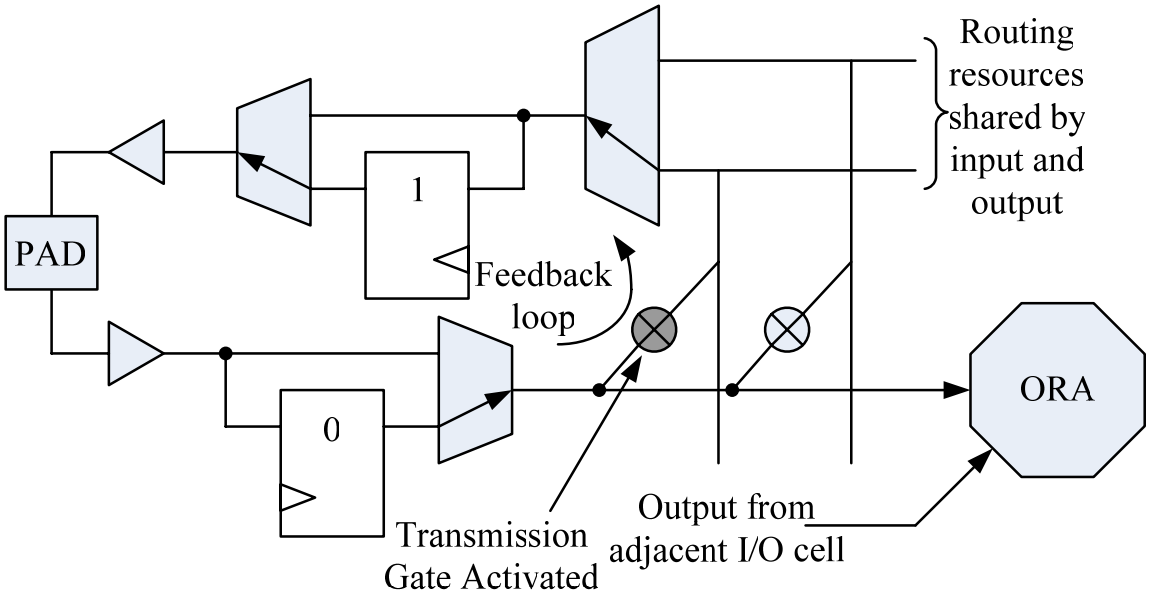


Figure 3.10: Transmission Gate Stuck-off BIST Configuratio

3.2 Testing the Global Reset CIP

As shown in Figures 2.13 and 2.15, all the primary and secondary I/O cells have access to the global reset network, except the secondary I/O cells present on the side interfaced to the AVR. By activating the global reset CIP, the I/O pin controls the global reset connection to all the flip-flops with reset activated. So, the global reset CIP of the every I/O cell has to be tested for stuck-on and stuck-off faults.

3.2.1 Stuck-On Test

To test the global reset CIP for a stuck-on fault an additional ORA is required to be added to the general BIST architecture and the test for stuck-off faults requires a totally different BIST architecture. Stuck-on faults of the global reset CIPs of all I/O cells are detected in parallel using the BIST architecture shown in Figure 3.1. The stuck-on fault can be detected by inserting a special ORA as shown in Figure 3.11. To detect the fault, both the flip-flops in the I/O cell have to be activated and reset prior to the BIST sequence. The tri-state control signal is made logic ‘1’ and the previously active global reset connection has to be deactivated before this test is performed. If none of the global reset CIPs are stuck-on, the TPG would source ‘1’s and ‘0’s which would reach the ORA with delay of two clock cycles and the ORA will latch up the first ‘1’ it observes. If any of the global reset CIPs is stuck-on, then the first ‘1’ from the TPG would be latched in the output flip-flop and the same value would also be sent to the global reset network. Since the reset is asynchronous, it will reset all the flip-flops, including the output flip-flop, and the input flip-flop and the ORA would never latch up a ‘1’. Hence, all I/O cells are tested in parallel for the global reset CIP stuck-on fault.

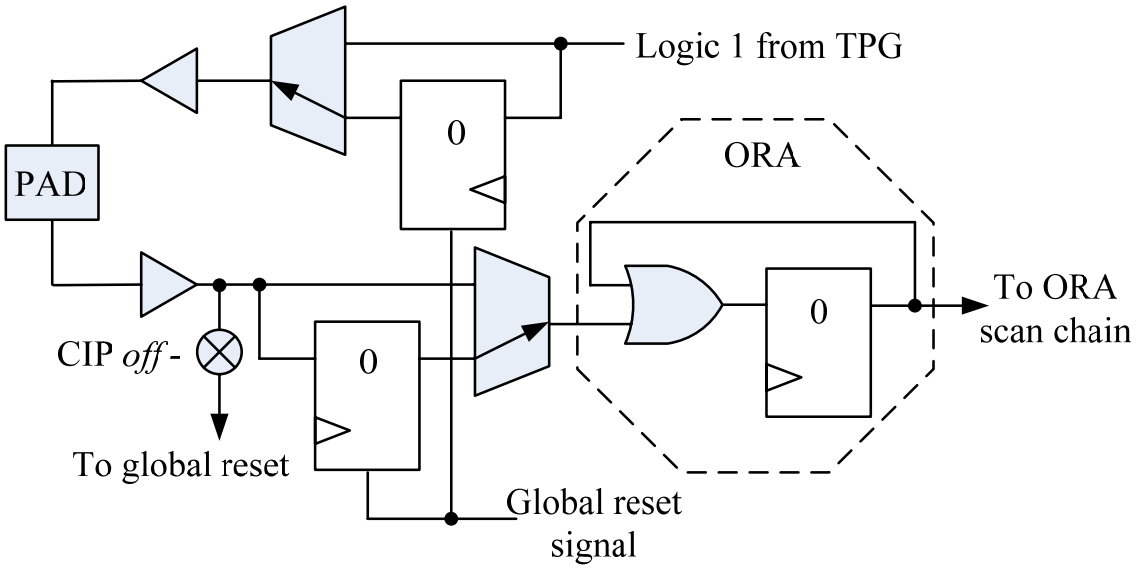


Figure 3.11: Global Reset CIP Stuck-on Test Configuration

3.2.2 Stuck-Off Test

To test for the global reset CIP stuck-off fault, the BIST architecture has to be completely changed. The global reset connection of each CIP has to be tested individually, so a separate BIST configuration is needed for each I/O cell to test its ability to drive the global reset. The BIST architecture for the global reset CIP stuck-off test is shown in Figure 3.12. Each CIP stuck-off fault is tested by making it the global reset signal and then a flip-flop with global reset connected is checked to see if it is being reset or not. During this BIST sequence, the TPG generates the expected output response of the flip-flop to be reset for the comparison in the ORA. The TPG is designed to be a 2-bit counter with the MSB (C1) being used as the input to the global reset and the complement of LSB (C0), being used as the input to the flip-flop being reset. Whenever ‘C1’ is logic ‘1’, the flip-flop is asynchronously reset, otherwise the ‘C0’ is clocked

through the flip-flop to the ORA. Here the flip-flop being reset can also be considered as a part of the ORA as it is being used to test the affect of global reset signal.

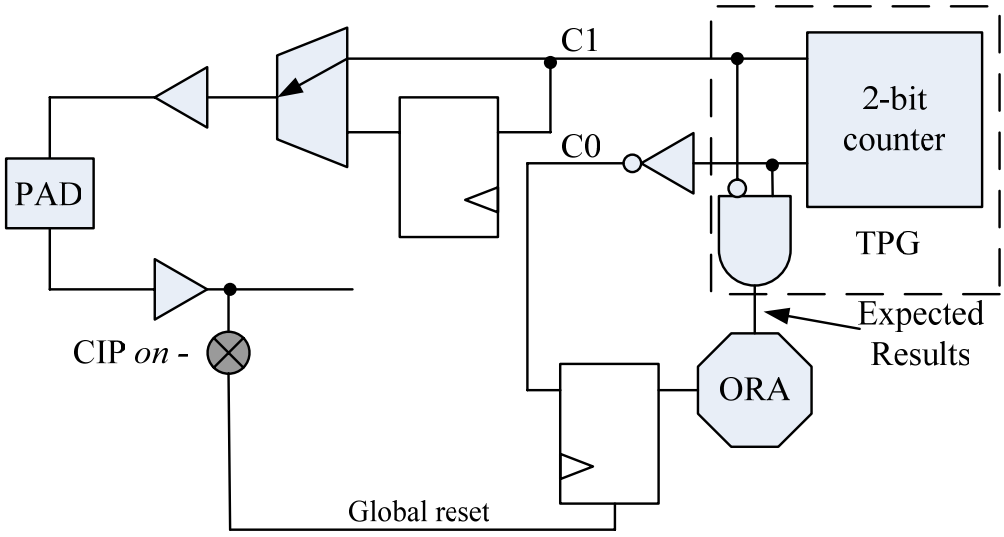


Figure 3.12: Global Reset CIP Stuck-off Test Configuration

A single ORA and a flip-flop is sufficient to test the functioning of the global reset. Since there are many unused resources in the FPGA, the number of ORAs has been increased to test most of the column-based resets. As shown in Figure 2.5, the Atmel FPGA has a *4x4* array of PLBs. Every column in the 4x4 array has a common reset connection. For example, if two flip-flops are activated in a column of a 4x4 array, then both the flip-flops should have an active or inactive reset connection. The column resets are also tested, along with the global reset CIP stuck-off faults, by having a flip-flop with an active reset connection in that column. Only half of the column resets can be tested because the remaining PLB columns are used to instantiate the TPG and ORAs, which should not have active reset connections to their flip-flops. Insertion of additional ORAs does not increase the test time but the scan out time of the results will be increased. The scan out time is negligible when compared with the download time. The column

reset connections are tested again, even though they are already tested during the BIST for logic resources.

3.3 BIST Configurations

To test the logic and routing resources associated with the primary I/O cells, excluding the global reset CIP stuck-off fault, a total of 13 configurations are required, nine for testing the multiplexer inputs along with the associated logic and routing resources and four for testing the transmission gates. Similarly, 10 configurations are required to test the logic and routing resources of the secondary I/O cells, eight to test the multiplexer inputs along with the associated logic and routing resources and two for testing the transmission gates. These 23 configurations are independent of the number of I/O cells present in the FPGA. The global reset CIP stuck-on test is performed during these 23 configurations. But the number of configurations required to test the global reset CIP stuck-off is dependent on the number of I/O cells. An Atmel AT94K10 FPSLIC has 138 general I/O cells which have connection to the global reset and an AT94K40 FPSLIC has 280 of them. The total number of BIST configurations has been tabulated in Table 3.3.

Table 3.3: Total Number of Configurations Required to Test the I/O Cells

FPSLIC MODEL	Primary I/O cells		Secondary I/O cells		Global Reset	Total Number of Configurations
	Multiplexer Inputs	Transmission Gates	Multiplexer Inputs	Transmission Gates		
AT94K10	9	4	8	2	138	161
AT94K40	9	4	8	2	280	303

A gate level model of the primary and secondary I/O cells was developed using Auburn Simulation Language and the fault simulation was performed using AUSIM [34]. The fault simulation results for primary and secondary I/O cells are shown in Figure 3.13. It can be seen that the last four configurations of primary I/O cells and the last two configurations of secondary I/O cells have constant fault coverage as only the transmission gate CIP connection is modified. Fault coverage of around 99.5% has been achieved for primary and secondary I/O cells from 13 and 10 configurations, respectively. 100% fault coverage is achieved after performing the global reset CIP stuck-off test.

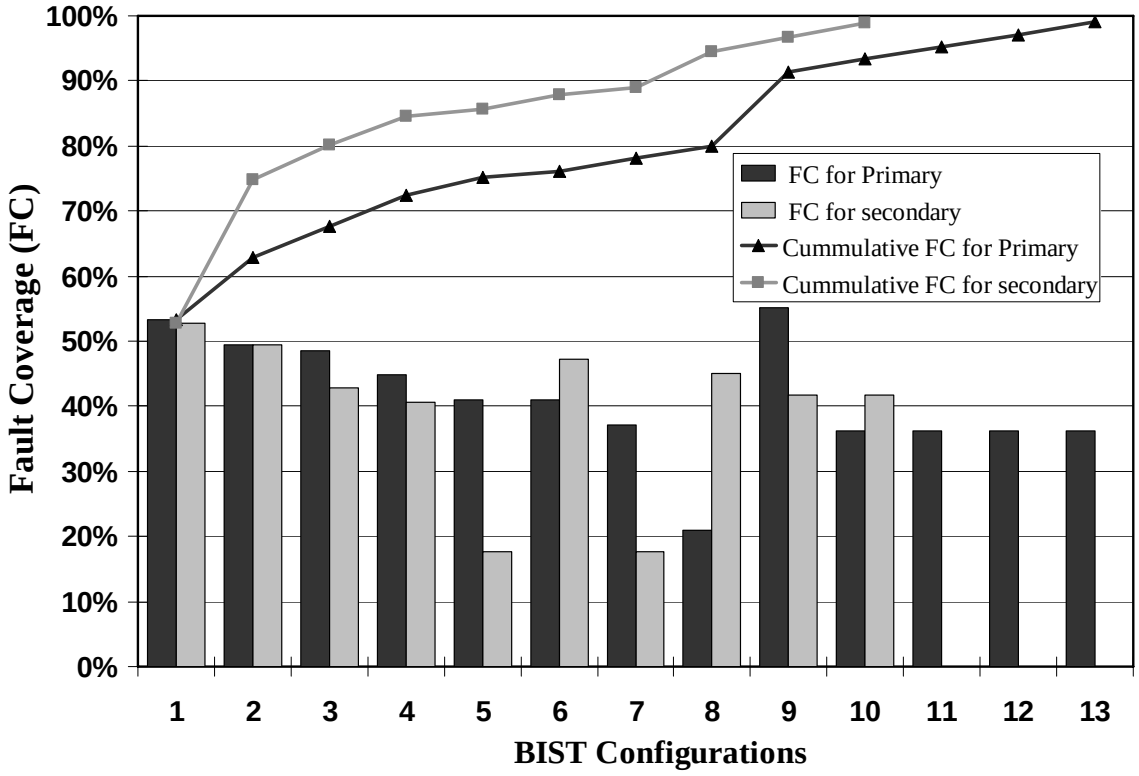


Figure 3.13: Individual and Cumulative Fault Coverage for Atmel AT94K I/O Cell BIST Configurations

Timing analysis was performed to find the paths with worst case delay. The maximum path delays in all three configurations for AT94K10 and AT94K40 FPSLICs is shown in Table 3.2. As the number of secondary I/O cells under test is high compared to the number of primary I/O cells, the load on the TPG is higher when secondary I/O cells are tested. So, the delay for the secondary I/O cells is higher than that of the primary I/O cells. As the TPG for the global reset signal drives all the primary and secondary I/O cells, it has the highest delay. But the delay from the reset signal to the ORA is less in this case, as it uses the global reset network.

Table 3.4: Timing Analysis for the Worst Case Path Delays

FPSLIC Model	Primary I/O cells	Secondary I/O cells	Global Reset
AT94K10	24.46ns	36.46ns	33.69ns
AT94K40	37.21ns	53.97ns	69.11ns

3.4 Automatic Configuration Generation Using MGL

The process of implementing the configurations using MGL is as follows:

- The TPG is described in MGL and is placed at some particular PLB locations.
- The I/O cells are then instantiated as bidirectional I/O cells and the TPG signals are routed to the data and tri-state multiplexers of the I/O cells. Using MGL, the unbonded I/O cells can be instantiated and the signals can be routed to unconnected multiplexer inputs which cannot be done using any HDL.
- The routing from the TPG to every I/O cell is described from the starting point of the TPG output, which comes from a PLB, to the end point of the multiplexer inputs of the I/O cell.
- After routing the TPG patterns to the I/O cells, the ORAs with the scan chain implementation are instantiated.
- The design described in MGL can be viewed in the Figaro window. Any unrouted nets can be routed using the Figaro tool itself. The tool will notify the user if there are any routing contentions.

- Routing contentions can be removed using the “optimize routing” function supported by Figaro. When “optimize routing” is used, it will modify some of the already described routes and it might remove some routes when it cannot route them.
- The design is then verified to see if all the required routing connections are still present after the routing has been optimized. If some of the inputs to the multiplexer have been removed, then the MGL routing description has to be changed.
- The bitstream for the described design is also generated using Figaro.

The bitstream will then be downloaded into the FPGA and the BIST clock is applied to execute the BIST sequence. At the end of the BIST sequence the results are stored in the ORAs and then the ORA results have to be scanned out. All the routes to the I/O cell have already been used by the TPG and therefore the ORA results cannot be scanned out through an I/O cell. So, the AVR interface is used to read the ORA results. The ORA results are first written into the AVR data memory and then read from the data memory. The ORA results are analyzed using a ‘C’ program. To verify the sanity of the ORA results, a string of 1’s is scanned out at the end of the scan chain. If there are no faults in the I/O cells then all the ORAs should have 0’s, with 1’s at the end of the scan chain. The absence of 1’s at the end of the scan chain indicates that the ORA results are not scanned out properly or that an ORA is faulty.

Three different MGL programs for testing primary I/O cells, secondary I/O cells and global reset CIP stuck-off were developed for each device, excluding the configurations for transmission gate stuck-off. First they were developed for AT94K10 FPSLICs and then for AT94K40 FPSLICs. The parameterized MGL programs developed for AT94K10 FPSLICs cannot be extended to AT94K40 FPSLICs just by

changing the array size. The routing contentions for AT94K40 FPSLICs were different from those of AT94K10 FPSLICs, so the routing has to be slightly modified. Also in AT94K40 FPSLICs, two fast *clock I/O cells* are present on west side of the FPGA core instead of general I/O cells, so these I/O cells are not included in the test. As a result, separate MGL programs were developed for the two different-sized devices. Therefore, a total of six MGL programs were developed for both the AT94K devices. The number of lines of non-commented source code for the MGL programs is shown in Table 3.5.

Table 3.5: Number of Lines of MGL Source Code for Different Master Configurations

FPSLIC Model	For Primary I/O Cells	For Secondary I/O Cells	For Global Reset CIP Stuck-off
AT94K10	484	573	308
AT94K40	470	574	450

The bitstreams for the three master BIST configurations are generated from the MGL programs using Figaro. The subsequent bitstreams are generated by modifying the master bitstreams. In the subsequent bitstreams, the TPG and the routing information of the FPGA remains the same and only the bits associated with the logic and routing of the I/O cells are modified and this can be done using a simple program. Thus all the bitstreams for the remaining configurations are generated.

The generated bitstreams are then downloaded into the FPGA and executed. The configurations were verified by injecting some faults. The faults were injected by modifying the configuration bits related to a particular I/O cell before being downloaded. The I/O cell which is configured differently from the other I/O cells would latch up 1’s in

two ORAs. The developed BIST configurations have been tested on actual faulty chips as well. The configurations have been applied on nine AT94K10 FPSLICs which were made faulty by zapping some wire segments inside them with lasers. The information on the zapped wires has not been disclosed. Of the nine FPGAs, four of them failed the I/O cell BIST configurations. None of them showed faults in the I/O cells but most of the faults were detected by not scanning out 1's at the end of the scan chain. One of the devices could not successfully download the bitstream, indicating the fault is with some other core of the FPSLIC rather than FPGA core.

When the test was performed on fault-free devices in configurations with flip-flops activated, some initialization faults in the registers of the I/O cells were detected. After the BIST configuration had been downloaded, it was observed that all the output flip-flops were initialized to logic '0' and all the input flip-flops were initialized to '1' in AT94K10 FPSLICs. Whereas in AT94K40 FPSLICs it was observed that one output flip-flop and one input flip-flop were not initialized correctly, they were initialized to their opposite states when compared with the initialization values of the registers in the other I/O cells. Another fault was identified when the secondary I/O cells of AT94K10 and AT94K40 devices were tested. It was observed that one of the I/O cell present along the side that interfaces with the AVR always failed, indicating that there was no loop back connection whereas it is shown in Figaro with a loop back connection. These results were consistent with all the FPGAs that were used, which indicate that these faults are actually minor design errors in the AT94K10 and AT94K40 FPSLICs.

3.5 Untested Resources

Even though 100% fault coverage is achieved, the resources which are not considered under fault simulation and the resources of the I/O cell which are not being tested are discussed in this section. All the routing resources from PLBs other than direct connections have to pass through the repeaters at the edges of the PLB array to reach the input of the data and tri-state multiplexers. The input signals to the I/O cells can also pass through the repeaters at the edge of the array (paths ***a*** and ***b*** in Figure 3.9) to reach the PLBs in the middle of the FPGA. The repeaters at the edges can be completely tested only when the signals are routed to or from the I/O cells. Only a few connections of the repeaters are tested, as the TPG signals are routed through the repeaters to the input side of the data and tri-state multiplexers.

The resources associated with the I/O clock cells described in Section 2.2.5.4 are also not tested. The I/O clock cells cannot be routed as normal I/O cells in bi-directional mode using MGL. To test the I/O clock cells, the TPG signals to the I/O cells and the ORA signals from the I/O cells to the ORAs have to be implemented by bitstream manipulation or processor reconfiguration, which is very difficult. So, the resources in the I/O clock cells and their connection to the global clock network are also not tested.

In some of the packages of the Atmel AT94K40 FPSLICs, two I/O cells are dedicated I/O fast clock cells. In other packages of the AT94K40 FPSLICs, the two dedicated I/O fast clock cells are replaced by normal I/O cells. But it is observed that the I/O cells that are replaced by fast clock I/O cells do not have flip-flops, similar to the clock I/O cells. Since their functionality would differ from the functionality of normal I/O cells, the I/O cells at the location of the fast clock I/O cells are also not tested.

3.6 Testing Time

The total test time includes the time required to download the BIST configuration, the test run time and results retrieval time. The total test time is dominated by the configuration download time, as the configuration memory is very large. The logic and routing resources present in the Atmel FPSLICs can be completely tested in 68 configurations [36], fewer than the 161 I/O cell BIST configurations for AT94K10 and 303 I/O cell BIST configurations for AT94K40 FPSLICs. The download time comparison is shown in Table 3.4.

Table 3.6: Download Time Comparison for Logic, Routing and RAM with I/O BIST [37]

FPSLIC Model	Time for one Download (ms)	Total Number of Downloads		Total Download Time (sec)	
		Logic, Routing and RAM	I/O cells	Logic, Routing and RAM	I/O cells
AT94K10	63	68	161	4.3	10.14
AT94K40	523	68	303	35.6	158.5

The download time of 158.5sec for testing the I/O cells is much higher when compared with the download time of 35.6sec for testing the rest of the resources in the FPGA core of AT94K40 FPSLICs. So, the test time for testing the resources associated with the I/O cells takes almost five times as much time than that required to test the rest of the resources in the FPGA core. This is primarily due to the number of configurations required for the global reset CIP stuck-off test.

In this chapter, three different BIST architectures, required to test all the resources of the programmable I/O cells in Atmel devices, have been discussed. It is observed that the download time and configuration memory storage requirements to test the I/O cells are high when compared with the download time and configuration memory requirements for Logic, Routing and RAM BIST approaches. In Chapter 4, some techniques to reduce the download time and configuration memory storage requirements, using the assistance of the embedded AVR processor, are discussed.

Chapter 4

Processor Assisted BIST for I/O cells

In the previous chapter it is observed that the download time for testing the I/O cells is much higher compared to testing the rest of the resources present in the FPGA. In this chapter, techniques to reduce the number of downloads by performing internal partial dynamic reconfiguration using the embedded processor are discussed along with their advantages and disadvantages. A new technique to generate the BIST clock signal using embedded processor reconfiguration will also be presented. Finally this chapter concludes with experimental results showing reduction in test time and configuration memory storage requirements.

4.1 Dynamic Partial Reconfiguration for BIST

The process of implementing and executing BIST through embedded processor reconfiguration has already been investigated for logic, routing and RAM resources in Atmel FPSLICs [21][28][34]. The maximum operating speed of the AVR processor is 25MHz, whereas the configuration memory download speed is only 1MHz. As the operation speed of the AVR processor is much faster, reconfiguration from the processor can reduce the test time. The main advantage of dynamic partial reconfiguration is that only selected bits can be modified while the rest of the configuration bits remain unmodified. In most of the BIST configurations the variation in configuration memory

bits from one configuration to the next is very small. In such cases, selected bits can be easily reconfigured using the AVR processor. This completely avoids the download required for the next BIST configuration and hence the download time for the next BIST configuration is eliminated.

To test the I/O cells by doing the reconfiguration using the embedded processor, the bitstream generated from the MGL is downloaded into the FPGA along with a program to be executed by the AVR. In this AVR program, the initial BIST configuration is executed under the control of the AVR through internal generation of the BIST clock. After executing the BIST sequence, instead of scanning out the ORA results and downloading the next BIST configuration, the I/O cells under test are reconfigured without scanning the ORA results. The I/O cells are reconfigured by the AVR processor according to the downloaded program and the BIST sequence is now executed for that BIST configuration. The successive BIST configurations are generated by the AVR processor unless the logic and routing resources associated with the FPGA have to be reconfigured.

This procedure is continued until all the multiplexer inputs and logic resources in the I/O cells are tested, in the first nine configurations. Next the transmission gates associated with the I/O cells are tested. For this, the I/O cells have to be configured in loop back mode, complementing values have to be stored in the input and output flip-flops of the I/O cells and all the TPG signals driving the multiplexer inputs have to be blocked. The feedback loop is activated by reconfiguring the transmission gates. Complementing values are stored in the flip-flops when the test is performed in the previous configuration (i.e., in the ninth configuration), and all the TPG signals are

blocked by reconfiguring the repeaters at the edge of the PLB array. The repeaters at the edges are sometimes used to route the ORA signals, so the repeaters which route the ORA signals are not reconfigured. After reconfiguring the repeaters the transmission gate test is ready to be performed by internally generating the BIST clock. To test the next transmission gate configuration, the currently active transmission gate has to be turned off and a new one has to be activated, and also the data multiplexer input has to be changed. Thus all the transmission gates are tested.

After testing the transmission gates, the ORA results are scanned out. So, the shift signal is made '1' and the output of the scan chain is connected to the AVR interface. As a single MGL program is used to generate the first BIST configuration for primary I/O cells, it is called the master MGL program for primary I/O cells.

When the primary I/O cells in Atmel devices are being tested, the bitstream generated by MGL is downloaded into the FPGA. The T4 bit of the TPG is routed to the tri-state control signal. But the tri-state multiplexer has two T4 signals, one from the banked tri-state and the other from the global routing resources of the FPGA core. One of the two inputs is selected by the Figaro routing tool and the selection is done randomly. To make sure that the same input is tested in all the I/O cells, the I/O cells are reconfigured even before they are tested in their first BIST configuration. The bitstream generated by MGL could be modified using a 'C' program before being downloaded but AVR reconfiguration is an easier process.

The configuration required to test the secondary I/O cells is different from the configuration developed to test the primary I/O cells, as the I/O cells along the AVR interface are also included in the test. This increases the number of ORAs to be

instantiated and the direct connections from the edge PLBs to the I/O cells also have to be changed. To instantiate and route the ORAs and also to modify the existing routing connections using the AVR processor reconfiguration only, requires considerable development time. Instead, a separate BIST configuration for testing the secondary I/O cells was developed using MGL, with a simpler program generated for AVR processor reconfiguration.

The secondary I/O cells are tested in a similar manner to primary I/O cells. First the bitstream generated by MGL is downloaded and the secondary I/O cells are reconfigured before getting tested in their first BIST configuration, for the same reason as mentioned for primary I/O cells. The remaining BIST configurations and the tests for the transmission gates are performed in a similar way to that of primary I/O cells.

After testing the primary and secondary I/O cells, the global reset CIP stuck-off test is performed. As the architecture to test the global reset CIP is also completely different, the initial bitstream to be downloaded into the FPGA is generated using MGL along with a program to be executed by the AVR for BIST reconfiguration and results retrieval. After downloading the bitstream and the AVR program, the test is performed in the first configuration to test the global reset CIP stuck-off fault of a single I/O cell. After the test is performed, the global reset CIP of the I/O cell which has been tested is deactivated and the global reset CIP of the next I/O cell to be tested is activated. The successive reconfigurations are obtained by just reconfiguring two CIPs instead of downloading the whole configuration. Partial reconfiguration saves a lot of download time and memory storage requirements when testing the global reset CIPs for stuck-off faults.

Three different AVR reconfiguration programs have been written for primary, secondary and global reset CIP stuck-off configurations of each FPGA. The programs required to communicate with the AVR were already developed in [21][29]. A total of six AVR reconfiguration programs have been developed, three for AT94K10 FPSLICs and three for AT94K40 FPSLICs. The number of lines of non-commented source code for the six programs is shown in Table 4.1. The programs are compiled using the AVR 'C' compiler. The program memory requirements of the AVR processor reconfiguration programs are shown in Table 4.2 and require less than 10% of the total program memory space available. The bitstream generated by MGL and the *hex* file generated by the AVR 'C' compiler are combined using Atmel System Designer tool and downloaded into the FPGA [37]. The configuration bitstream is downloaded into the FPGA and the reconfiguration procedure is stored in the AVR program memory.

Table 4.1: Number of Lines of AVR 'C' Code

Device Model	Primary	Secondary	Global Reset
AT94K10	638	673	339
AT94K40	682	687	354

Table 4.2: Memory Required for Storing the AVR Program

Device Model	Primary (Bytes)	Secondary (Bytes)	Global Reset (Bytes)
AT94K10	1120	2498	2228
AT94K40	1152	1366	1128

4.2 Retrieving BIST Results

If the BIST configurations are executed by downloading every configuration separately, then the ORA results are scanned out at the end of each configuration. If the BIST is executed by doing AVR reconfiguration instead of external downloads, then the BIST results are scanned out only at the end of all BIST configurations for that download. So, the BIST results are scanned out only three times, firstly after testing all the primary I/O cells, secondly after testing all the secondary I/O cells, and finally after testing all the global reset CIPs for their stuck-off faults. Scanning out the results only three times reduces the test time but the diagnosis results can only identify the faulty I/O cell instead of identifying the faulty resource in the I/O cell. Whereas the global reset CIP stuck-off tests will just indicate that one of the global reset CIPs is stuck-off and the faulty I/O cell cannot be identified.

If the ORA results are scanned out at the end of every configuration after performing the test, the ORA flip-flops have to be cleared before the test is performed in the next configuration, as a string of 1's is scanned into the ORA shift register. If more accuracy is required in diagnosing a faulty resource, then the ORA results can be scanned at the end of every BIST configuration, followed by clearing the ORA contents before executing next BIST configuration.

There is another disadvantage in testing the secondary I/O cells using processor-assisted BIST. In Figure 3.6, ORA3 compares the responses of I/O0 and I/O4. If the PLBs are reconfigured to test the other direct connection from the secondary I/O cell, the ORA3 now compares the responses of I/O6 and I/O2. In such a case, if a mismatch is latched up in one of the ORAs, the faulty I/O cell cannot be uniquely diagnosed. To

diagnose a faulty I/O cell exactly, only one direct connection from secondary I/O cell to the PLB can be tested.

4.3 Generating BIST Clock Cycles

In order to execute the BIST sequence, the clock must be generated from the AVR. The FPGAIORE signal from the AVR was previously used as the clock signal for Logic and Routing BIST approaches, which used reconfiguration from the AVR processor [21][29]. The FPGAIORE enable signal was routed to a clock I/O cell in order to connect to the clock network. The connection between the FPGAIORE and the clock I/O cell cannot be activated using MGL as MGL does not have access to the AVR processors' resources. The routing resources have to be reconfigured from the AVR processor to connect the FPGAIORE signal to the clock I/O cell. The FPGAIORE signal and the clock I/O cell are shown in Figure 4.1.

In the actual implementation, shown in Figure 4.1, the input to the clock I/O cell has been connected from a PLB close to FPGAIORE and this route is connected using MGL. Then the FPGAIORE signal is routed from the AVR to the PLB, which is easier when compared to routing the FPGAIORE directly to the I/O clock cell. To route the FPGAIORE signal from the AVR, the repeaters and the CIPs of the routing resources have to be programmed without causing conflicts with the other routing resources.

So, the routing from the AVR to the clock I/O cell has to be changed according to the routing of the other cells. Even small changes in design would change the routing, so the route from FPGAIORE has to be modified every time, accordingly. To remove this problem of routing from the AVR to the clock I/O cell, the clock can be generated from

the hard-wired ‘1’ and ‘0’ signals present at the data multiplexer of the I/O cell via partial reconfiguration by the AVR. A brief description of clock I/O cells was given in Section 2.2.5.4.

To generate the clock, the clock I/O cell is reconfigured in bidirectional mode using the AVR and the output of the data multiplexer is toggled between ‘1’ and ‘0’ by reconfiguring the data multiplexer selection signal. The two configurations of the clock I/O cell are shown in Figure 4.2. So, this procedure is more convenient to be implemented than the previous approach of routing and generating the clock from FPGAIORE.

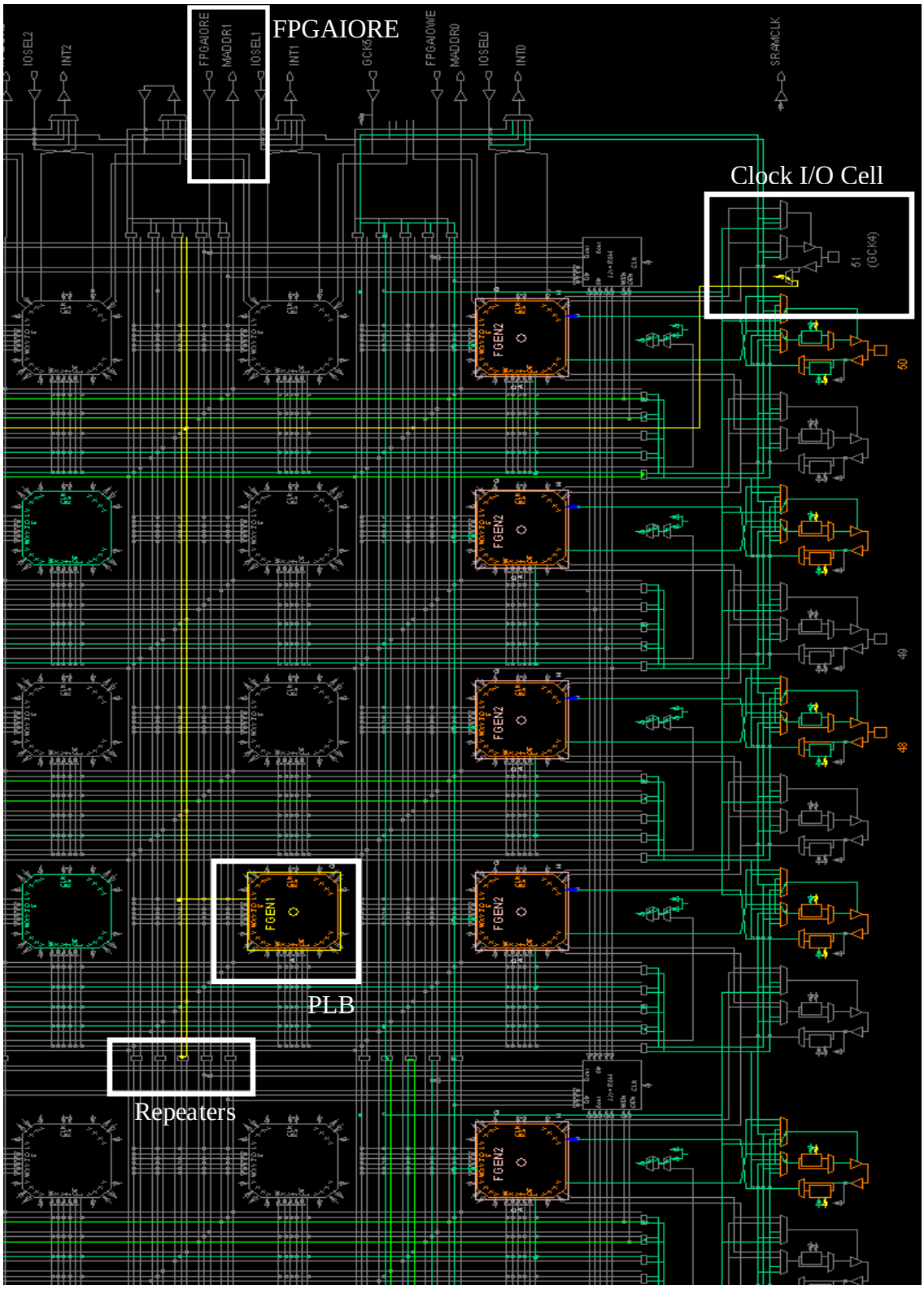


Figure 4.1: FPGAIORE and I/O Clock Cell Connection in BIST for Primary I/O Cells

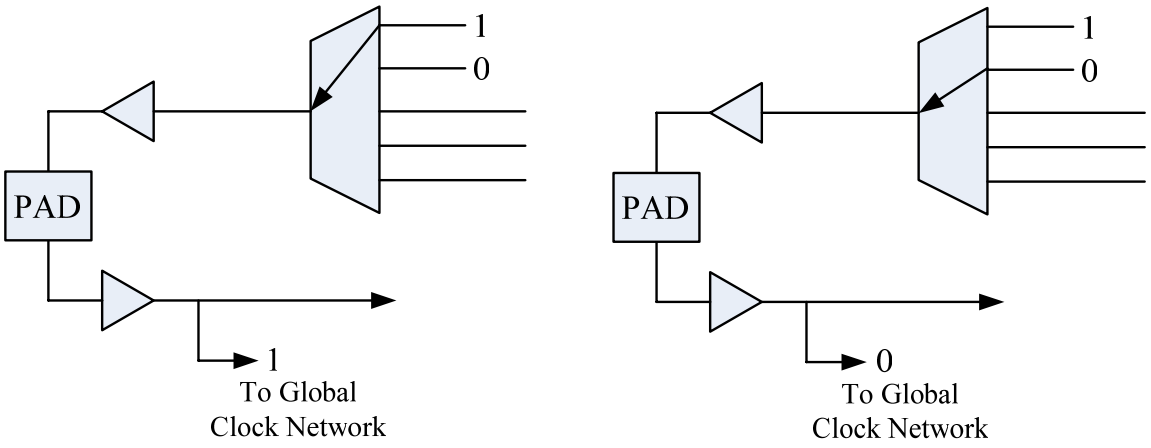


Figure 4.2: Clock Generation using Multiplexer Reconfiguration

The FPGAIORE generates a clock pulse whenever the FISUA register, a register present in the processor, is written with a value. To generate a clock signal from FPGAIORE, some value is repeatedly read into the FISUA register. To generate a clock cycle using FPGAIORE, one instruction is required, whereas to generate a clock cycle using clock I/O cell reconfiguration, two instructions are required, one for selecting hardwired '0' and the other for selecting hardwired '1'. So, the frequency of the clock generated by reconfiguring the clock I/O cell reconfiguration is about half that of the frequency of the clock generated by FPGAIORE.

Using the clock generation by reconfiguring the data multiplexer of the clock I/O cell, any clock I/O cell can be used to generate the BIST clock, whereas the FPGAIORE signal is difficult to route to the clock I/O cells present on the other side of the FPGA or which are located away from that signal. As any of the clock I/O cells can be used to generate a clock by data multiplexer reconfiguration from the AVR, all the clock I/O cells can be easily activated and their proper functioning can be tested.

4.4 Testing Time

The BIST configurations for testing the I/O cells can be categorized into three master BIST configurations, namely primary I/O cell configurations, secondary I/O cell configurations and global reset CIP stuck-off configurations. For all three master BIST configurations, an initial download is required and the successive BIST configurations are generated by AVR processor reconfiguration. The test time for each master BIST configuration includes the download time, the AVR reconfiguration and test run time, and the time required to scan out the ORA results. The reconfiguration time, the test run time and the time required to scan out the ORA results can be collectively referred to as the processor execution time. The processor execution times are calculated using AVR Studio, a tool provided by Atmel for simulating and debugging AVR programs. The processor execution times for BIST configurations of primary I/O cells are shown in Table 4.3. Atmel's AVR Studio gives the number of clock cycles required by the AVR to execute the instructions. The clock cycles required for reconfiguring and testing multiplexer inputs and transmission gates and the clock cycles required for scanning out the ORA results are also shown in Table 4.3 [15]. The AVR processor can operate at a maximum frequency of 25MHz [15] and this frequency is used to calculate the time required for executing the given number of clock cycles.

Table 4.3: Processor Execution Time for Primary I/O Cells

Device Model	Number of Clock Cycles				Processor Execution Time (ms)
	Multiplexer Input Configurations	Transmission Gate Configurations	Scanning the ORA Results	Total	
AT94K10	87,684	26,917	8,724	123,325	4.93
AT94K40	154,193	53,050	33,238	240,481	9.61

The number of I/O cells to be reconfigured in the AT94K40 device (280) is nearly twice the number of I/O cells to be reconfigured in the AT94K10 (138) device. The processor execution times of 9.61ms and 4.93ms are also proportional by the same ratio. Therefore, it can be stated that the total reconfiguration time dominates the test run time, since the test run times are same for both devices. The time required to scan out the ORA results depends on the number of ORAs, as well as the number of 1's being scanned at the end of the scan chain. The total number of bits scanned out of the scan chain for the AT94K40 devices is four times more than the number of bits scanned out for AT94K10 devices. The number of clock cycles required to scan them out is also proportional to that.

The processor execution times required to test the secondary I/O cells of the Atmel devices are shown in Table 4.4, where the format of the table is same as that for the primary I/O cells.

Table 4.4: Processor Execution Time for Secondary I/O Cells

Device Model	Number of Clock Cycles				Processor Execution Time (ms)
	Multiplexer Input Configurations	Transmission Gate Configurations	Scanning the ORA Results	Total	
AT94K10	106,348	27,054	15,175	148,577	5.94
AT94K40	201,619	55,651	56,705	313,975	12.56

The processor execution times of AT94K10 (5.94ms) and AT94K40 (12.56ms) devices are also proportional to the number of I/O cells. Even though the number of configurations required to test the multiplexers of secondary I/O cells (eight) is less than that of primary (nine), the number of secondary I/O cells under test that are to be reconfigured is higher in number (22 additional I/O cells on the interface to the AVR). As the reconfiguration takes more time than executing the test, the number of clock cycles required to test the multiplexer inputs of the secondary I/O cells (106,348 and 201,619) is higher than the number of clock cycles required to test those of primary I/O cells (87,684 and 154,193).

The I/O cells present on the interface to the AVR have no flip-flops, so the transmission gates of those I/O cells are not being tested. As the number of primary and secondary I/O cells under test for transmission gate tests are same, they almost have an equal number of clock cycles (53,050 and 56,705). The number of ORA results being scanned out in BIST configurations of the primary and secondary I/O cells of both the devices is equal. But the number of ORA cycles for scanning out the ORA results of the secondary I/O cells is found to be higher, even though the same ‘C’ program has been

used in both the cases. The difference in times was due to different assembly language implementation of the same 'C' program by the AVR 'C' compiler. A part of the 'C' program with its implementation in assembly language is shown in Table 4.5.

Table 4.5: Implementation of 'C' Program by AVR in Assembly Language

Implementation 1	Implementation 2

149 for (i = 1; i < 8; i++)
LDI R16,LOW(1)
CPI R16,8
BRSH _0xF
150 {
151 ora [ii] = ora[ii] << 1;
MOV R30,R4
LDD R26,Y+3
LDD R27,Y+3+1
LDI R31,0
ADD R30,R26
ADC R31,R27
PUSH R31
PUSH R30
MOV R30,R4
LDI R31,0
ADD R26,R30
ADC R27,R31
LD R30,X
LSL R30
POP R26
POP R27
ST X,R30
152 temp = FISUA;
IN R17,20
153 temp = temp & 0x01;
ANDI R17,LOW(1)
154 ora[ii] = ora[ii] | temp;
	143 for (i = 1; i < 8; i++)
	LDI R16,LOW(1)
	CPI R16,8
	BRSH _0xF
	144 {
	145 ora [ii] = ora[ii] << 1;
	RCALL SUBOPT_0x4
	ADD R30,R26
	ADC R31,R27
	PUSH R31
	PUSH R30
	RCALL SUBOPT_0x4
	ADD R26,R30
	ADC R27,R31
	LD R30,X
	LSL R30
	POP R26
	POP R27
	ST X,R30
	146 temp = FISUA;
	IN R17,20
	147 temp = temp & 0x01;
	ANDI R17,LOW(1)
	148 ora[ii] = ora[ii] | temp;

In the implementations shown in Table 4.5, the lines with the numbers at the beginning are the commands in 'C' language and the assembly language implementation of the same command is shown below it. *Implementation1* of the 'C' program took 8,724

cycles for execution and *Implementation 2* took 15,175 cycles for executing the same loop an equal number of times. The reason for the higher number of clock cycles for *Implementation 2* is the subroutine calls, highlighted in *Implementation 2* column of Table 4.5. The 'C' code in *Implementation 1* is similar to that of a macro. As macros can be executed faster than subroutine calls, *Implementation 1* takes fewer clock cycles than *Implementation 2*. In Table 4.8, it is shown that this affect is negligible.

The number of clock cycles and processor execution times required for the global reset CIP stuck-off tests are shown in Table 4.6.

Table 4.6: Processor Execution Time for Global Reset CIP Stuck-off Tests

Device Model	Number of Clock Cycles			Processor Execution Time (ms)
	Testing All Global Reset CIPs	Scanning the ORA Results	Total	
AT94K10	28,874	14,711	43,585	1.74
AT94K40	59,495	56,193	115,688	4.63

The number of clock cycles for testing all the global reset CIPs is proportional to the number of I/O cells of the devices. As the number of ORAs scanned out for AT94K40 devices is four times higher than the AT94K10 devices, the number of clock cycles for scanning the ORA results are also proportional. *Implementation 2* of Table 4.5 was used by the AVR 'C' program for scanning out the ORA results.

The processor execution times, for the AT94K10 and AT94K40 devices using AVR reconfiguration, are shown in Table 4.7. Even though the AT94K40 devices are four times larger than the AT94K10 devices, the number of I/O cells in AT94K10

devices is only twice compared to AT94K10 devices and the total processor execution times (12.61ms and 26.8ms) are almost proportional.

Table 4.7: Total Processor Execution Time

Device Model	Processor Execution Time			Total Processor Execution Time (ms)
	Primary I/O Cells (ms)	Secondary I/O Cells (ms)	Global Reset CIP Stuck-off (ms)	
AT94K10	4.93	5.94	1.74	12.61
AT94K40	9.61	12.56	4.63	26.8

The total test time, including the download time, and a comparison of the download time with the processor execution time is shown in Table 4.8. It can be seen that the percentages, 93.75 and 98.32, of download time are very high and dominate the total test time. Therefore, the un-optimized subroutines by AVR 'C' compiler, as shown in Table 4.5, do not have much affect on the test time even though additional clock cycles are required for processor execution.

Table 4.8: Total Test Time Using AVR Reconfiguration and Percentage of Download Time

Device Model	Download Time (ms)	Processor Execution Time (ms)	Total Test Time (ms)	% of Download Time in Total Test Time
AT94K10	63 x 3 = 189	12.61	201.61	93.75
AT94K40	523 x3 = 1569	26.8	1595.8	98.32

The total reduction in test time using AVR reconfiguration, compared to the test time using MGL downloads is shown in Table 4.9. The test download times using MGL are taken from Table 3.6 and test run time and ORA scan out times have been added.

Table 4.9: Comparison of Total Test Times

Device Model	Test Time Using MGL (sec)	Test Time Using AVR Reconfiguration (ms)	Speed Up Achieved
AT94K10	10.196	201.61	50.57
AT94K40	158.606	1595.8	99.39

4.5 Configuration Memory Storage Requirements

The number of downloads required to test the I/O cells in the Atmel devices using MGL is 161 for the AT94K10 and 303 for the AT94K40. This has been reduced to three, and also the number is independent of the array size of the FPGA core. The configuration memory storage requirements for configurations using MGL and AVR-based reconfiguration are shown in Table 4.10. There is more than two orders of magnitude reduction in configuration memory storage requirements for the AT94K40 devices.

Table 4.10: Comparison of Configuration Memory Storage Requirements

FPGA Model	Configuration Memory (KB)	Total Downloads		Total Configuration Memory (MB)		Configuration Memory Storage Requirements are Reduced by
		MGL	AVR	MGL	AVR	
AT94K10	16.947	161	3	2.728	0.051	53.67
AT94K40	65.115	303	3	19.73	0.195	101

A speed up of 99.39 times and the reduction in configuration memory storage requirements by 101 times are achieved using the reconfiguration by the embedded AVR processor. It can also be seen that as the size of the device is increased, the speed up time is increased, along with the reduction in configuration memory storage requirements. So, as the size of the FPGA increases, the use of embedded processor reconfiguration has more advantages.

Chapter 5

Summary and Conclusions

A general BIST approach to test the logic and routing resources of the programmable I/O cells in a FPGA or associated with the FPGA core of a SoC was presented. None of the prior BIST approaches to test the programmable logic and routing resources in FPGAs have addressed testing the I/O cells. The approach proposed in this thesis can be applied to test the I/O cells of any FPGA. The technique was applied to the I/O cells of Atmel FPGAs and FPGA cores in SoCs. It is observed that the proposed BIST technique can test all the logic resources but it cannot test all the routing resources associated with the I/O cells. So, two additional test approaches have been developed to completely test the associated routing resources. The additional BIST approaches are used to test the transmission gates, a part of the routing resources, and the global reset connection associated with the I/O cells.

The proposed I/O cell BIST approach can be used to test the I/O cells at manufacturing level and also at device level. The I/O cell BIST approach can test all the bonded and unbonded I/O cells, so it is package independent. The BIST for I/O cells can detect all the faults associated with the logic and routing resources, along with the major defects in the analog programmable features, like pull-up and pull-down capabilities. But it cannot detect all of the parametric faults that affect the analog programmable features of the I/O cell, like V_{OL}, V_{OH}, V_{IL}, V_{IH}, current sink and source capabilities,

programmable delays, etc. The BIST approach can detect the faults in the configuration bits controlling the analog parametric features. For example, if the configuration bits for drive capability do not provide any drive then the fault can be detected. Similarly, if the delay offered by the delay element is too large and does not meet the set up time of the ORA flip-flops then those faults can also be detected.

Even though the BIST configurations can be used for manufacturing as well as device level testing, they cannot be used at the system-level testing as the connections from other devices on the same PCB may back drive the I/O cells which are normally configured as input cells. If all the inputs from the other devices present on the PCB can be tri-stated, then the developed BIST configurations can be applied to test at the system-level. Sometimes the loads connected to the I/O cell will increase the worst case path delays of the I/O cells and this might require reduction in the BIST clock frequency for the test to be performed with the I/O cells being loaded [38].

5.1 Main Contributions

BIST configurations to test the programmable I/O cells in the FPGA core of Atmel AT94K10 and AT94K40 devices were developed. 100% stuck-at gate level fault coverage of the I/O cells was obtained with the developed BIST configurations. The 100% stuck-at gate level fault coverage was verified by performing fault simulation. Three master BIST configurations, for primary I/O cells, secondary I/O cells and global reset CIP stuck-off, were developed using MGL for each of the AT94K10 and AT94K40 devices. Separate AVR reconfiguration programs were also developed for each of the master BIST configurations. All of these BIST configurations have been downloaded

and verified on the Atmel AT94K10-25BQC and AT94K40-25AJC packages of the Atmel FPSLICs. Some of the material associated with this development was published in [39] and [40].

A new BIST clock generation scheme by AVR-based reconfiguration of the data multiplexer of the I/O clock cell has been designed and implemented. The new clock generation scheme is easier to implement when compared with the previous BIST clock generation scheme. Also, this new clock generation scheme enables testing of all the CIP connections to the global clock network from all the I/O clock cells without any routing modification requirements.

5.2 Potential Application to Other FPGAs/SoCs

The proposed general BIST architecture for testing the I/O cells can be used to test the logic resources of programmable I/O cells in any FPGA or SoC. But some new BIST configurations might be required to test the routing resources, depending on the routing architecture. The special features included in the programmable I/O cells might vary from one FPGA or SoC manufacturer to another and might require different BIST configurations to test them. The BIST architecture was also investigated to test the logic resources in the I/O cells of the Xilinx Virtex-4 devices. From the fault simulation results, a maximum gate level stuck-at fault coverage of 98.56% can be obtained with seven BIST configurations using the same BIST approach proposed in this thesis [38]. This demonstrates the general application of the I/O cell BIST approach.

5.3 Areas of Future Research and Development

The proposed BIST approach is not parameterized to test a given subset of I/O cells. If the BIST approach can be parameterized to test a given set of I/O cells, then the BIST configurations can be used to test the FPGAs or SoCs at the system-level as well. At system-level testing, parameterized BIST configurations can be used to test the I/O cells which are configured as output and bi-directional cells, and the unbonded I/O cells can also be tested. The I/O cells configured as inputs might back drive the I/O cells under test and may not be included in the test. Even the I/O cells configured as output cells might be required to be tested at different BIST clock frequencies, depending on their loading, because the larger loads will increase the worst case path delay.

In most of the recent Xilinx FPGAs, a pair of programmable I/O cells can be configured in a differential pair mode and have dedicated routing resources to be used in the differential pair mode. A new BIST architecture should be developed to the test the programmable I/O cells configured in the differential pair mode.

BIBLIOGRAPHY

[1] M. Bushnell, V. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-signal VLSI Circuits. Boston, MA: Kluwer Academic Publishers, 2000.

[2] L. Harriott, "A new role for E - Beam: Electron projection," *IEEE Spectrum*, Vol. 36, No. 7, pp. 41-45, 1999.

[3] G. Moore, "MOS Transistors as Individual Devices and in Integrated Arrays," in *Proc. of the National Electronics Conf.*, pp. 25-30, 1965.

[4] S. Hamdioui, *Testing Static Random Access Memories Defects, Fault Models and Test Patterns.* Boston, MA: Kluwer Academic Publishers, 2004.

[5] C. Stroud, *A Designer's guide to Built-in Self-Test.* Boston, MA: Kluwer Academic Publishers, 2002.

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

[7] I. Dear, "Economic Effects in Design and Test," *Proc. IEEE Design and Test of Computers*, Vol. 8, No. 4, pp. 64-77, 1991.

[8] S. Brown, R. Francis, J. Rose, Z. Vranesic, *Field-Programmable Gate Arrays.* Boston, MA: Kluwer Academic Publishers, 1992.

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

[10] J. Rose, A. Gamal, A. Sangiovanni-Vincentelli, "A Classification and Survey of Field-Programmable Gate Array Architectures," *Proc. IEEE*, Vol. 81, No. 7, pp. 1030-41, 1993.

[11] C. Stroud, S. Wijesuriya, C. Hamilton, M. Abramovici, "Built-in self-test of FPGA Interconnect," *Proc. IEEE. International Test Conference*, pp. 404-411, 1998.

[12] W. Wolf, *Modern VLSI Design: System-on-Chip Design.* Upper Saddle River, NJ: Prentice Hall PTR, 2002.

[13] __, “Virtex-II Pro/ Virtex-II Pro X Complete Data Sheet (All Four Modules),” Data Sheet DS083 (v 4.5), Xilinx, Inc., 2005 (available at www.xilinx.com).

[14] R. Baker, H. Li, D. Boyce, *CMOS Circuit Design, Layout, and Simulation.* New Delhi: Prentice-Hall of India, 2003.

[15] __, “AT94K Series Field Programmable System Level Integrated Circuit,” Data Sheet, Atmel Corp., 2001 (available at www.atmel.com).

[16] T. Gabara, W. Fischer, W. Werner, S. Siegel, M. Kothandaraman, P. Metz, D. Gradl, “LVDS I/O Buffers with a Controlled Reference Circuit,” *Proc.* 10^{th} *IEEE, International ASIC Conference and Exhibit*, pp. 311-315, 1997.

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

[18] L. Zhao, D. Walker, F. Lombardi, “IDDQ Testing of Input/Output Resources of SRAM-Based FPGAs,” *Proc. Asian Test Symposium*, pp. 375-380, 1999.

[19] M. Abramovici, C. Stroud, “BIST-based Test and Diagnosis of FPGA Logic Blocks,” *IEEE Trans. on VLSI Systems*, Vol. 9, Issue: 1, pp. 159–172, 2001.

[20] S. D. Brown, R. Francis, J. Rose, Z. Vranesic, *Field Programmable Gate Arrays.* Boston, MA: Kluwer Academic Publishers, 1992.

[21] S. Garimella, “Built-In Self Test for Regular Structured Embedded Cores in Sytem-on-Chip,” Masters Thesis, Auburn University, 2005.

[22] M. Smith, *Application-Specific Integrated Circuits.* Addison-Wesley, 1997.

[23] J. Rabaey, A. Chandrakasan, B. Nikolic, *Digital Integrated Circuits A Design Perspective.* Prentice Hall, 2002.

[24] __, Virtex-4 User Guide, User Guide UG070 (v 1.4), Xilinx, Inc., 2005 (available at www.xil-inx.com).

[25] __, “AT94K Series Field Programmable System Level Integrated Circuit,” User Guide, Atmel Corp., 2001 (available at www.atmel.com).

[26] H. Michinishi, T. Yokohira, T. Okamoto, T. Inoue, H. Fujiwara, “A Test Methodology for Interconnect Structures of LUT-Based FPGAs,” *Proc. IEEE Asian Test Symposium*, pp. 68-74, 1996.

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

[28] C. Stroud, J. Sunwoo, S. Garimella, J. Harris, "Built-In Self-Test for System-on-Chip: A Case Study," *Proc. IEEE International Test Conference*, pp. 837-846, 2004.

[29] J. Sunwoo, "Built-In Self Test of Programmable Resources in Microcontroller Based System-On-Chips," Masters Thesis, Auburn University, 2005.

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

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

[32] D. Fernandes, I. Harris, "Application of Built-In Self-Test for Interconnect Testing of FPGAs," *Proc. IEEE International Test Conference*, pp. 1248-1257, 2003.

[33] C. Stroud, S. Garimella, "Built-In Self-Test and Diagnosis of Multiple Embedded Cores in SoCs," *Proc. International Conference on Embedded Systems and Applications*, pp. 130-136, 2005.

[34] C. Stroud, "AUSIM: Auburn University Simulator – Version L2.3", Dept. of Electrical and Computer Engineering, Auburn University, 2004.

[35] J. Sunwoo, C. Stroud, "Built-In Self-Test of Configurable Cores in SoCs Using Embedded Processor Dynamic Reconfiguration", *Proc. International SoC Design Conference*, pp. 174-177, 2005.

[36] __, "AT94K Series Configuration", Application Note, Atmel Corp., 2001 (available at www.atmel.com).

[37] __, " AT94K/AT94S Series System Designer 3.0", User Guide, Atmel Corp., 2004 (available at www.atmel.com).

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

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

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

APPENDIX A

LIST OF ACRONYMS

ASIC	-	Application Specific Integrated Circuit
AVR	-	Advanced Virtual RISC
BIST	-	Built-In Self-Test
BUT	-	Block Under Test
CAD	-	Computer Aided Design
CIP	-	Configurable Interconnect Point
CUT	-	Circuit Under Test
CSoC	-	Configurable System on Chip
DC	-	Direct Current
DFT	-	Design For Testability
DSP	-	Digital Signal Processor
DLL	-	Delay Locked Loop
EPROM	-	Erasable Programmable Read-Only Memory
EEPROM	-	Electrically Erasable Programmable Read-Only Memory
FPGA	-	Field Programmable Gate Array
FPSLIC	-	Filed Programmable System Level Integrated Circuit
HDL	-	Hardware Description Language
IC	-	Integrated Circuit

I/O	-	Input/Output
LUT	-	Look-Up Table
LSB	-	Least Significant Bit
LFSR	-	Linear Feedback Shift Register
MGL	-	Macro Generation Language
ORA	-	Output Response Analyzer
PCB	-	Printed Circuit Board
PIP	-	Programmable Interconnect Point
PLB	-	Programmable Logic Block
RISC	-	Reduced Instruction Set Computer
SoC	-	System on Chip
SRAM	-	Static Random Access Memory
TPG	-	Test Pattern Generator
VLSI	-	Very Large Scale Integration
WUT	-	Wire Under Test