Single Event Upset error detection on routing tracks of Xilinx FPGAs

SINGLE EVENT UPSET ERROR DETECTION ON ROUTING TRACKS OF XILINX FPGAS

BY

BILLY TAJ, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2013) McMaster University

(Computing and Software) Hamilton, Ontario, Canada

TITLE:	Single Event Upset error detection on routing tracks of Xilinx FPGAs
AUTHOR:	Billy Taj B.Eng., Mechatronics Engineering McMaster University, Canada
SUPERVISORS:	Dr. Mark Lawford and Dr. Nicola Nicolici
NUMBER OF PAGES:	xii, 89

Contents

List of Tables

List of Figures

Abstract

This thesis proposes a new method to detect routing switch alterations on FPGAs in real-time. By sampling the circuit path at the source and destination, and comparing the samples, it is possible to find out if there has been a routing change in the circuit path. We compare and contrast this probing method with previously established techniques such as Cyclic Redundancy Checks, Built-in-self-tests, Triple Modular Redundancy, Duplication with Comparison, and redesigning the FPGA. The probe method finds the routing error in one clock cycle, using the pre-existing elements on the FPGA, while the FPGA is still operational. This method works on all FPGAs that use the Wilton style switchbox. An automated tool for probing the design circuit is presented in this thesis that applies the probe scheme on a circuit built on the Xilinx Virtex-4 FPGA.

Chapter 1

Introduction

Integrated circuits are used everywhere from the common cellular phone to the various telecommunications and monitor satellites that orbit the Earth. Nuclear power plants around the world are in need of upgrades. New instrumentation and controls systems are being added to the latest plant designs. The International Atomic Energy Agency (IAEA) has begun an initiative to consider placing Field Programmable Gate Arrays (FPGAs) in the next generation of nuclear power plants [IAEA, 2013]. Nuclear power plants emit electromagnetic radiation, which pose problems to integrated circuits [Kayali, 2002]. Traditional fault-mitigation methods such as: Triple modular redundancy [Pratt et al., 2007], Duplication with comparison [Johnson et al., 2008], and Built-in-self-tests [Sun et al., 2001] have been effective at detecting radiation errors, but there is room for improvement.

The particular problem that we will address in this thesis is the tendency for radiation to change the value of computer memory segments that control the FPGA.

1.1 Safety-critical systems

A safety-critical system is a machine, or a set of machines whose operational duties directly impact the safety of the people and the surrounding environment. Should a safety-critical system fail during its operation, the results could be catastrophic damage to the system itself, the surrounding environment, or even loss of life [Knight, 2002]. A classic example of a safety-critical system would be in the automotive industry. New vehicles rely on electronics to perform anti-lock braking, airbag deployment, and electronic speed control [Baleani et al., 2003]. If these systems fail, the result could be the destruction of the vehicle, or the loss of life of the passengers. Another example of safety-critical systems can be found in the Boeing 777 [Yeh, 2001]. Onboard systems such as: the Airplane Information Management System (AIMS), Primary Flight Computers (PFC), and the Air Data Inertial Reference System (ADIRU) are all systems that would be cause for alarm should they fail during the aircraft's operation.

Nuclear power plants contain computer systems that are used to perform calculations for the plant. Some of these computers are safety-critical systems that are used to shut down the plant in case of emergencies. These safety-critical computer systems are at risk of radiation effects because the radiation is capable of adversely affecting these computer systems.

1.2 Field Programmable Gate Arrays

Field programmable gate arrays (FPGAs) are a semiconductor technology that can be configured into different digital circuits on-the-fly by enabling and disabling a series of

transistor switches. At its base level, it is a collection of transistors, D-flip-flop (DFF) registers, computer memory, and look-up tables (LUTs) arranged in a 2-dimensional array. The array can be rearranged in a vast number of ways. These elements are connected by silicon segments with transistors at every junction. Digital circuits are placed onto the FPGA by manipulating the transistors at the junctions.

Traditional nuclear power plant controls systems were built using analog hardware. The initiative [IAEA, 2013] is interested in adding FPGAs to nuclear power plants. The rationale for upgrading the computer systems to FPGAs is that the logic on the FPGA will only contain the specific design logic. By comparison, the equivalent computer implementation requires a full software platform, a task-scheduler to switch between each job, and the computer hardware to run the programs.

FPGAs can also be used to replace obsolete components in safety-critical systems [Guzman-Miranda et al., 2011]. Due to the fact that the circuitry in FPGAs can be changed on-the-fly, the useful lifetime of the FPGA will be greater than an embedded system using a fixed computer core.

Finally, FPGAs can be used in conjunction with existing computer systems to augment the safety controls. The work by [Avizienis and Kelly, 1984] makes a case for a diversified approach to fault-tolerant designs. By having multiple instances of important modules, the point of failure for any one implementation can be mitigated.

1.3 The problem statement

This thesis presents a new method to detect single event upsets. This section will specify which type of problem is being solved, and the constraints applied to the problem for the purpose of this master's thesis.

1.3.1 Single Event Upsets

When an irradiated particle collides with the silicon layers of an FPGA, the path of travel and the impact zone of the particle becomes ionized (see Figure 1.3.1). This excess charge left by the collision causes a change in voltage [Kayali, 2002]. When the voltage change occurs on a piece of memory, the value of the memory will change. Memory values can only be 1 or 0, therefore when this phenomenon occurs, it is called a bitflip. FPGAs use transistors to control the circuit routes that form the desired logic. These transistors receive their selection values from the FPGA configuration memory. If the sectors of memory that control the circuit routing are hit, the circuit pathway changes. If these pathways deviate from the original design, then the circuit is no longer guaranteed to function as originally intended.

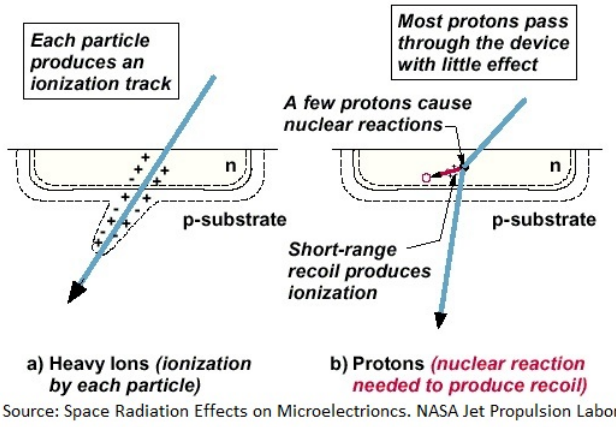


Figure 1.3.1: Single event upsets, viewed on the molecular scale [Kayali, 2002]

.

1.4 The goals of the solution

1.4.1 Fault model

SEUs can cause errors in the FPGA memory, which can affect everything in the FPGA; from LUTs to routing tracks. It is impossible for a system to be fault-free due to the limitations of the physical universe. Therefore, the aim of safety-critical systems is to limit the number of failure occurrences to an acceptable percentage. For the purposes of this research, the fault model will be restricted to SEU effects on the routing mechanisms of the FPGA.

1.4.2 Coverage and cost

Current error detection schemes (Triple Modular Redundancy ([Pratt et al., 2007]), Duplication With Comparison ([Johnson et al., 2008])) can detect errors on the circuit at the cost of FPGA resources. Other error detection schemes (Built-In-Self-Tests ([Sun et al., 2001]), and Cyclic Redundancy Checks (CRC) ([Altera Corporation, 2013])) can detect errors at the cost of time. Another set of error detection schemes (designing new FPGAs (Reddy et al. [2005] and [Rohani et al., 2009]), and buying radiation-hardened FPGAs ([Rockett et al., 2007])) can detect errors by sacrificing non-recurring engineering costs. The ideal error detection scheme should be able to detect all of the errors while using the least amount of resources, as soon as it occurs, and without having to re-invent the FPGA.

1.4.3 Thesis organization

This thesis will present a solution that can detect SEUs that affect the routing network of the FPGA. The solution can detect single SEU strikes. The routing track is controlled by a series of statements that outline the circuit path in a textfile that represents the circuit. The SEU strike will be represented by the enabling or disabling of a connection point within the textfile. The new solution will probe the routing path at the beginning and end of the path, and compare the probe data to determine if the path has changed. Chapter 2 will provide the supporting background information for the solution. Chapter 3 is a review of the previous work that has been done to detect SEU effects on routing tracks. Chapter 4 is a description of the newly proposed detection method. Chapter 5 is an outline of the software tool to implement the detection solution. Chapter 6 contains the analytical details of the solution, as well as the proof-of-concept work. Chapter 7 is a summary of the future work and a conclusion to the thesis.

The contributions of this thesis is summed up in the list below:

- Developing a new method to detect single SEU bitflips that change the routing tracks of an FPGA circuit.
- Reverse engineering the Xilinx switch fabric in order to demonstrate the effectiveness of the detection scheme.
- Creating a software tool to automate the application of the detection method onto an FPGA circuit.
- Applying the software tool to an industrial scale safety critical design example.

Chapter 2

Background

In order to create a new method to detect single event upsets, the internal details of the FPGA must be understood. This section describes the prerequisite background information to understand the work that was done in this thesis. Lookup tables (LUTs), D flip-flops (DFFs), logic elements (LEs), and switchboxes are the essential elements to the FPGAs, and will be explained in this chapter. The chapter will proceed to explain the concept of fanout and fan-in ports, as well as touch upon the Xilinx Design Language (XDL), and the Xilinx development suite. Finally, the chapter will conclude with an outline of the error schemes that can be expected when dealing with single event upsets on the routing tracks of the FPGA.

2.1 Lookup Table

A lookup table is one of the two fundamental pieces of an FPGA. It is responsible for all of the logical behaviour on the FPGA. The lookup table (commonly referred to as a LUT) is a physical representation of a truth table. A truth table is a list of

all possible outputs; given all possible combinations of input signals. An example of a LUT can be seen in Figure 2.1.1.

w	x	y	z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

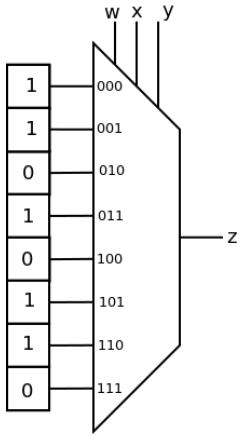
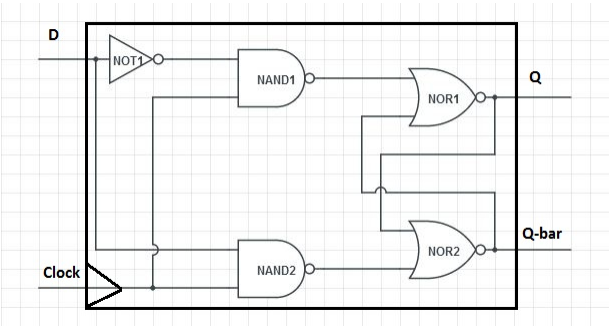


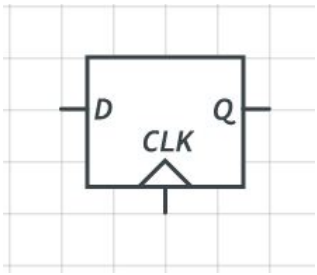
Figure 2.1.1: Truth table and circuit diagram of a 3-input LUT [Bergstra, 2012].

2.2 D flip-flop

A Delay flip-flop (DFF) is a digital circuit that is capable of storing the state of a particular electrical channel. It is commonly referred to as a D flip-flop, a register, or a DFF. In this thesis, it will be referred to as a DFF. There are 4 ports to this electronic circuit. The data, clock, output, and inverted output. The data port is the port labelled D. The clock port is denoted with the triangular shape. Lastly, the output ports are indicated with the letter Q (for plain output) and Q-bar (for inverted output). [Lin, 2003] The clock pulse will arrive at set intervals. When the clock pulse arrives at the clock port, the value at the D port will appear at the Q port. The duration of this output will match the input until the next clock pulse arrives. When the next clock pulse arrives, the new value of D will appear at Q. Figure 2.2.1 is an example of a DFF.



(a) The internals of a DFF



(b) The common symbol for a DFF. The inverted output is often dropped from the symbol

Figure 2.2.1: The DFF. It is the ubiquitous storage element of the FPGA, aside from the configuration memory.

The DFF is aptly named because it can delay the value of a signal from reaching its destination if several DFFs are placed in sequence. DFFs are very important to FPGAs because they can hold the state of the channel. By holding the value of a signal, the circuit has access to present values and past values of signal lines. Without memory, values cannot be held and retained.

2.2.1 Logic Element

The logic element (LE) is a fundamental item of an FPGA. The LE is a collection of sub-elements: Lookup tables (LUT) and DFF storage elements. A digital circuit is created by manipulating the LUTs and DFFs to produce the desired behaviour. The figure below is the LE of the Xilinx Virtex 4 FPGA. It contains two LUTs and two DFFs as well as the necessary support pieces to control them. A digital circuit is created by enabling connections from the outward-facing lines from a LE, to the inward-facing lines of another LE. Figure 2.2.2 is an image of the logic

2.2.2 Switchboxes

The switchbox is a routing structure designed to connect two or more LEts to form a useful circuit. Transistor switches are used to control the routing switch fabric of the FPGA. These switches are also called "Programmable Interconnect Points (PIP). If a PIP was turned on, then a digital signal can pass through the transistor. Conversely, if the PIP was turned off, the signal will not pass through the switch. A switchbox is a grid of programmable interconnect points arranged in a cluster. There are three main varieties of switchbox employed in the FPGA:

- Universal - This switchbox can connect any input port to any output port. This style also uses the most transistors.
- Planar - This style is the most basic switchbox. Each input port has a single designated output port. This style is also the cheapest to create since it uses the least amount of transistors
- Wilton[Wilton, 1997] - This switchbox style connects input ports to a subset of output ports.

In a typical FPGA, there may be more than one type of switchbox used in the design as each one has a specific purpose. For example the Virtex 4 uses Wilton and planar switchboxes. The different types of switchboxes can be seen in Figure 2.2.3. The switchboxes use in the Virtex-4 FPGA can be seen in Figure 2.2.4.

2.3 FPGA routing network

A digital circuit is created on an FPGA by creating circuit paths between the LEs that populate the chip. There are 2 vendors that make up the majority of the FPGA market (Xilinx and Altera) and they have very different methods of routing the circuit.

2.3.1 Altera

The Altera Multi-track interconnect is a proprietary method to connect LEs [Altera Corporation, 2009]. This architecture style uses horizontal and vertical wire segments of set lengths. The LEs are clustered into groups called Logic Array Blocks (LAB). Connections made inside each LAB are handled by a lower-tiered local interconnect. The R-tracks (Horizontal) and C-tracks (Vertical) are to connect adjacent LABs. Figure 2.3.1 shows the Altera routing architecture.

2.3.2 Xilinx

The Xilinx routing architecture is a two-dimensional array of switchboxes, coupled with LEs. The combination of switchbox and LE is called a Configurable Logic Block (CLB). The switchbox of the CLB connects to the wide network of wires on the chip. Unlike the Altera Multi-track, the Xilinx switchbox handles all of the connections. Figure 2.3.2 is an example of the Xilinx routing architecture.

2.4 Fanout and Fan-in

A fan-out is a property of the Wilton switchbox data port in the Xilinx switchbox. A port is said to have a fan-out if there is more than one location for the data to

travel to. Conversely, a port is said to have a fan-in if the port can accept signals from multiple ports. Fanout ports have data coming into a switchbox, while a fan-in has data leaving the switchbox. The purpose of having fan-ins and fan-outs is for routing LEs that are far away. The lines coming out of one switchbox have limited range. If a destination LE was out of range from its source, then the signal path can enter an interim switchbox and use a fan-out to route itself to the final destination. An example of this interim switchbox routing can be seen in Figure 2.4.3. Figure 2.4.1 shows an example of a fan-out port. Figure 2.4.2 shows an example of a fan-in port.

2.5 Xilinx Design Language

The Xilinx Design Language (XDL) is an ASCII representation of the circuit that is to be placed onto the FPGA. To be precise, this file is the HDL design before it is encrypted and downloaded onto the FPGA. The information inside the XDL is a collection of LE occupancy, LUT equations, and routing resources. There are two sections to the XDL:

- The *Logic Element (LE) occupancy section* lists the elements of the chip being allocated by the design.
- The *Routing pathways section* lists the PIPs that are used to connect the elements in the LE occupancy section.

Each element represents a signal source or a signal destination. Every element must be connected in the XDL.

2.6 CAD Flow

The Xilinx ISE Toolsuite is the main working environment for writing HDL designs to be placed into Xilinx FPGAs. The process to turn HDL designs to FPGA bitstream information is illustrated in Figure 2.6.1 and each point in the figure is described in the list below.

1. The CAD flow starts with an HDL design file. The design file is read, and decomposed into a list of modules needed to reproduce the functionality of the design file. The decomposed list is called a netlist. The decomposition phase is called *synthesis*.

2. The Translation phase is where the netlist output from the synthesis phase becomes more refined. At this stage, a device family needs to be specified. The generic netlist is mapped onto the specific FPGA device, creating a netlist used only for this specific device. This netlist is called a *Technology-Mapped* (TM) netlist.

3. After the TM netlist is created, the tool flow heads to the Place and Route (PnR) phase. The TM netlist contains specific device elements that are used to create the original HDL file, but the elements are not yet mapped onto the FPGA chip. The PnR algorithms determine the final placements of the TM netlist elements and route them in an optimal scheme to reduce time delay, power consumption, and FPGA chip area.

4. Once the PnR phase is finished, a new netlist is generated. This file contains the final location and design of the circuit to be placed onto the FPGA. This netlist

is encrypted and downloaded to the FPGA as a serial string of bit information (bitstream).

2.7 SEU error schemes

In FPGA switchboxes, each bitflip represents the activity of a connection. One bitflip may either activate a new connection, or disable a pre-existing connection. The following describes the possible wire effects when an SEU strikes a switchbox. The scenarios may happen at any port. There is no distinction between output or input due to the fact that the PIPs simply toggle the activity of 2 ports.

2.7.1 Single bitflip

The subfigures in Figure 2.7.1 depict what will happen when a single SEU strikes the FPGA. Ports A, D, E, and H in the example are active ports that connect to critical LEs in the test circuit. Ports B, C, F, and G are inactive ports not being used by the example. Input port A connects to output port H, and input port D connects to output port E. The black ports are unused ports that are not connected to any circuitry outside of the scope of the image. Figure 2.7.1a is the switchbox in its unaltered state. The following errors may occur:

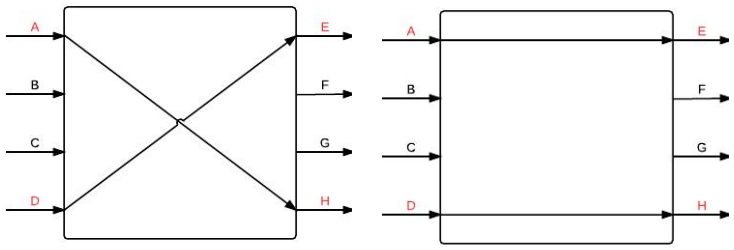
- A connection that branches from an existing connection may appear. This is called a fan-out antenna and can be seen in Figure 2.7.1b.
- A connection that creates an electrical short between 2 unrelated paths in close proximity to each other. Figure 2.7.1c is an example of this route.

- An existing connection can be broken, since a change in memory to an enabled switch would disable the connection. Figure 2.7.1d is an example of this error.
- A connection may occur that does not affect any existing logic, shown in Figure 2.7.1e.

2.7.2 Two bitflips

When two SEUs strike the FPGA, more errors may occur. In addition to having multiple instances of single SEU errors, the cases illustrated in Figure 2.7.2 can occur. For the first case (shown in Figure 2.7.2b) an existing path can be altered. It takes one SEU to disable an existing connection, and a second SEU to enable a new connection. The second case (seen in Figure 2.7.2c) forms two fan-outs that do not connect to active sections of the FPGA. The third case (shown in Figure 2.7.2d) may cause a short to occur by enabling two fan-outs, one of which forms a short circuit with the other active connection. The fourth case is a short circuit caused by a connection being rerouted, which can be seen in FIgure 2.7.2e. Figure 2.7.2a is the switchbox in its unaltered state.

2.7.3 More than two bitflips



(a) Example correct switchbox

(b) A logic change can occur if more than four bitflips happen.

Figure 2.7.3: The possible error configurations that require more than 2 bitflips

In addition to multiple occurrences of single and double bitflip errors, more complex logic changes can occur if more than two bitflips occur. Figure 2.7.3 shows the unique error that may occur if more than three bitflips are present. One SEU turns off the original transistor, another SEU enables a new path. Having four SEUs may result in a logic change, which is shown in Figure 2.7.3b. Figure 2.7.3a is the switchbox in its unaltered state. This thesis is concerned with single bit-flip errors.

2.8 Chapter Summary

LUTs, LEs, DFFs, and switchboxes are the essential elements that form the FPGA, and will be used extensively to detect SEUs on the FPGA routing network. This thesis introduces a new method that relies on the FPGA internals to detect SEUs, which can be seen in Chapter 4. Single event upsets happening on routing tracks will change the datapaths of the circuit on the FPGA. Previous work has been done to detect single event upsets on routing tracks, which can be seen in the next chapter.

This thesis presents a method to detect single SEU errors.

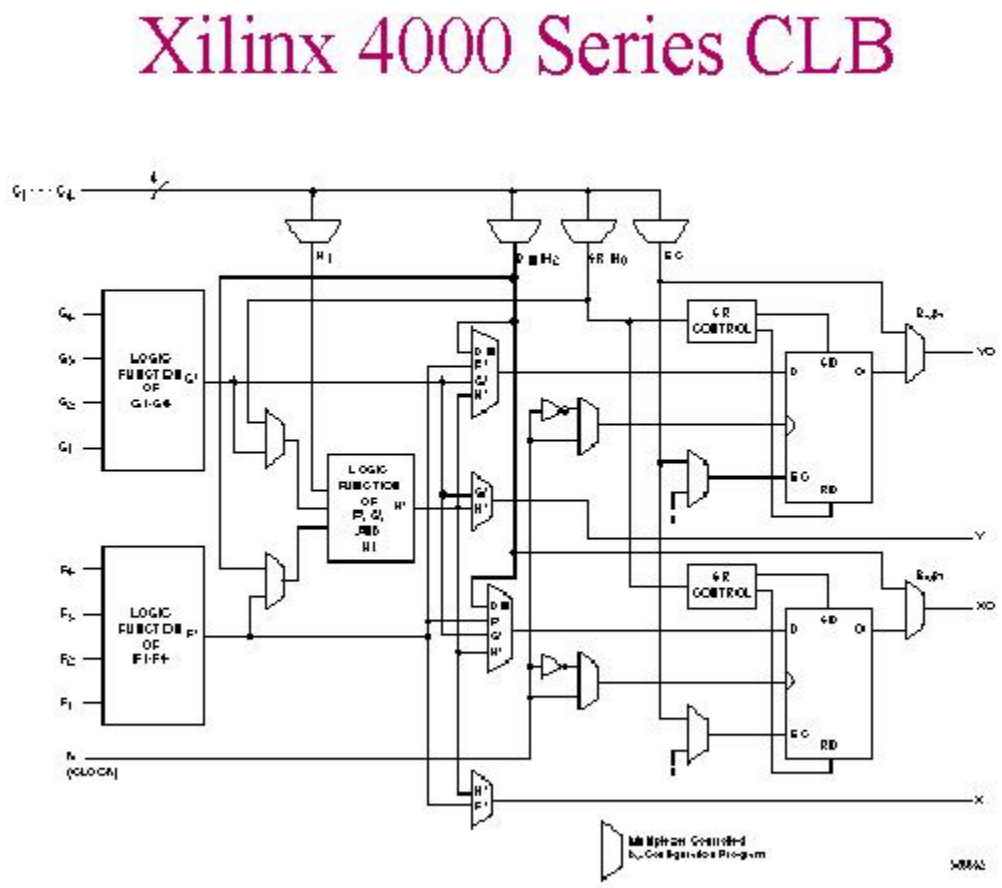
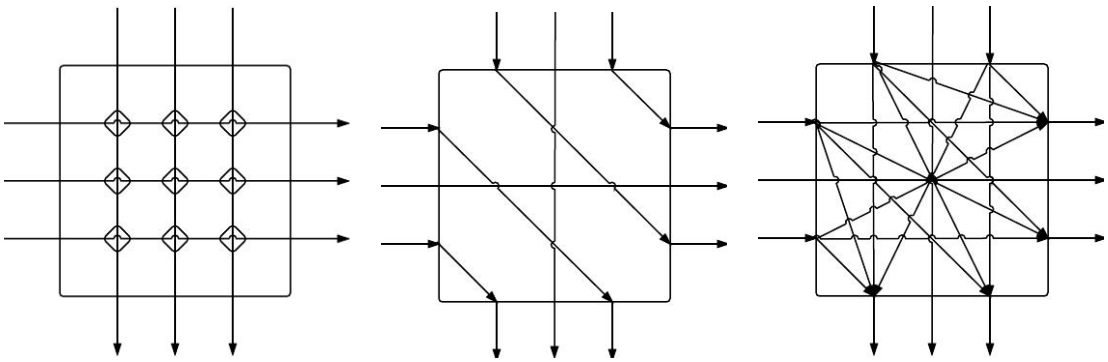


Figure 2.2.2: Logic element of the Xilinx 4000 series FPGA. It contains 2 DFFs and 2 LUTs [Goldstein, 1998]



(a) Universal Switchbox (b) Planar Switchbox (c) Wilton Switchbox

Figure 2.2.3: The different styles of switchboxes of FPGAs. The Xilinx FPGA uses the Wilton and Planar switchboxes

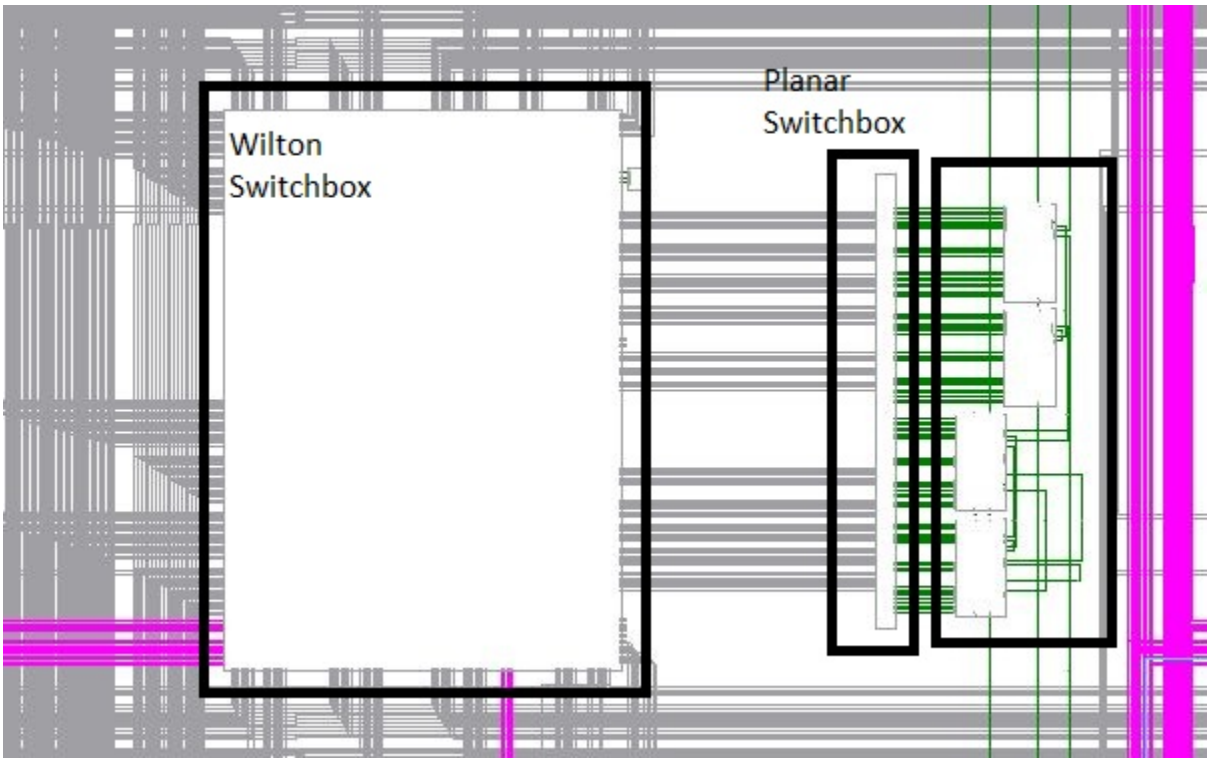


Figure 2.2.4: The layout for the Virtex 4 Configurable logic block; using 2 types of switchboxes

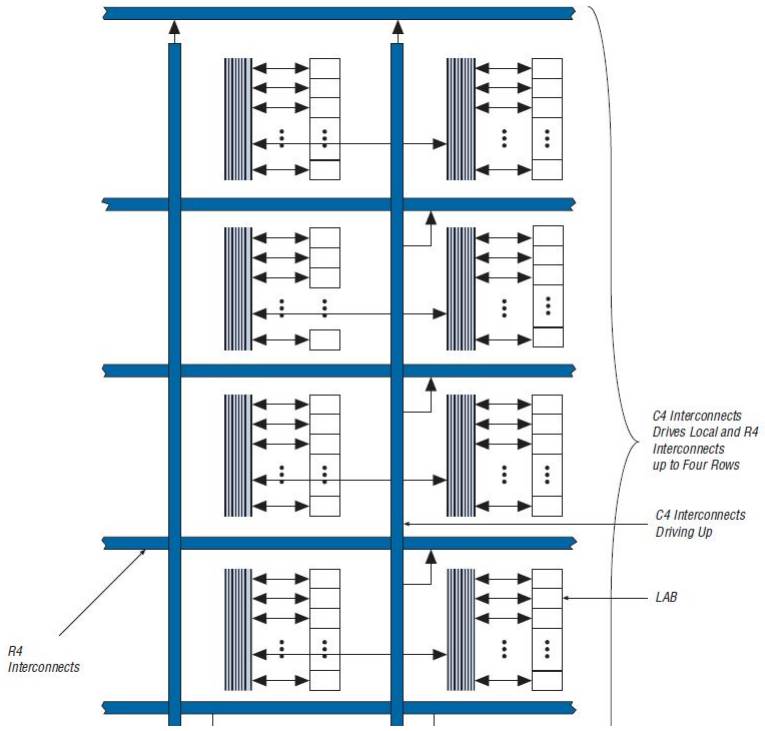


Figure 2.3.1: This is an example of the Altera Multi-track interconnect[Altera Corporation, 2009]. The blue lines are the LAB routing tracks. The smaller lines next to each LAB are the local interconnects

.

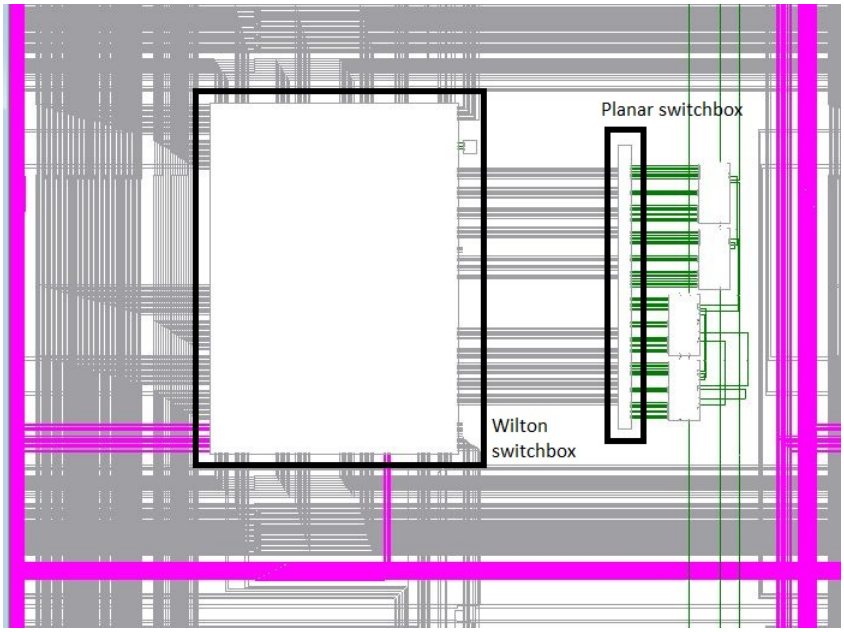


Figure 2.3.2: An example of the Xilinx routing network. Each CLB has their own switchbox. Each switchbox can connect to every other switchbox in the FPGA.

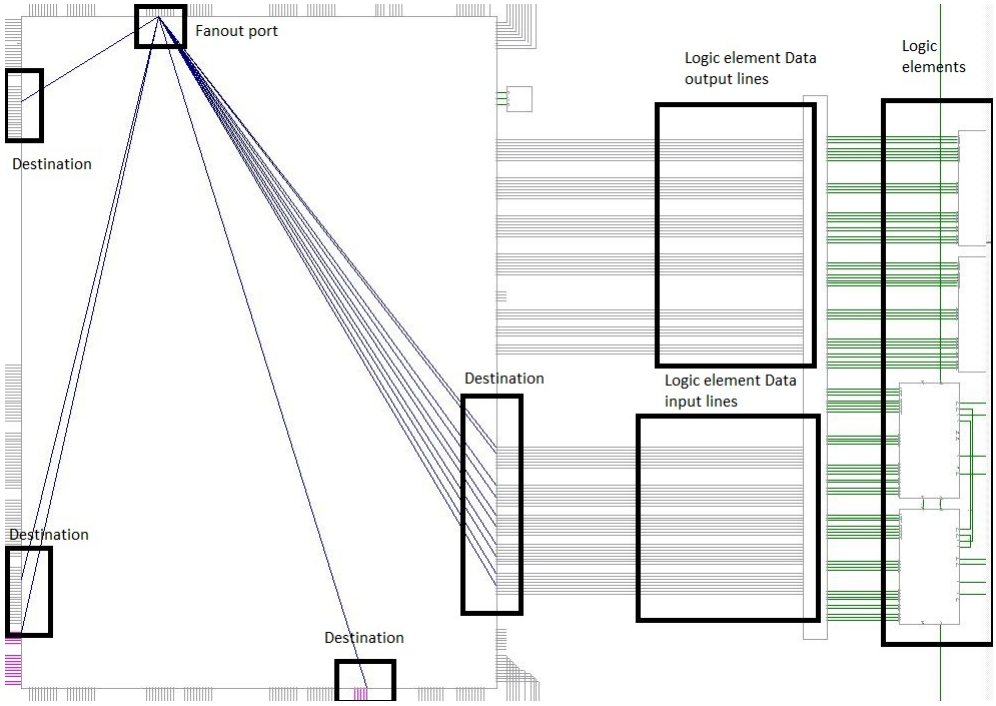


Figure 2.4.1: An example of a dataport with multiple fan-outs. One dataport can have multiple destinations simultaneously.

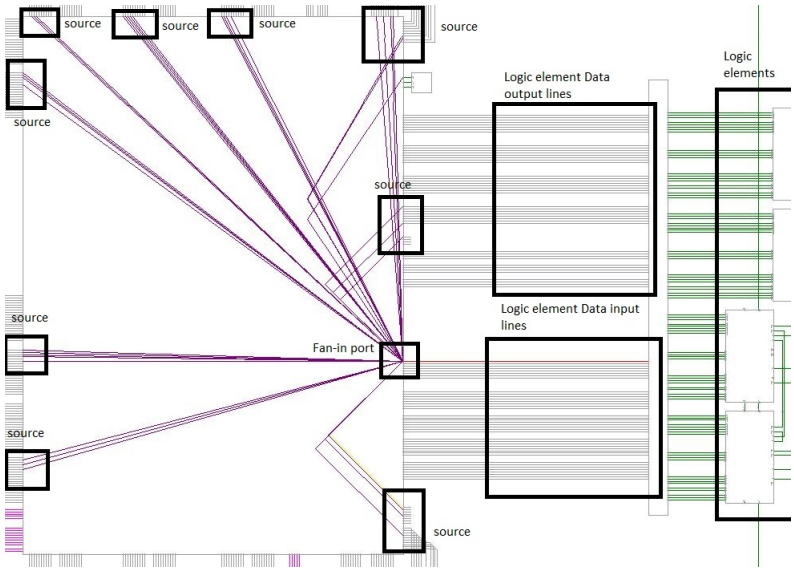


Figure 2.4.2: An example of a port with multiple fan-ins. One data input port can accept data from multiple ports, but only one at a time. If one data input port is connected to multiple sources, a circuit short will be formed.

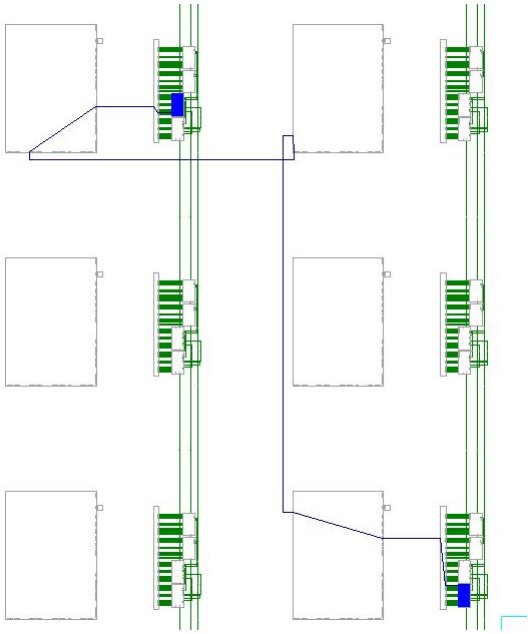


Figure 2.4.3: An example of a fan-out being used to route a signal. The circuit path is marked in black. The signal passes through an interim switchbox that is 3 coordinates north, uses a fan-out for that port, and continues to the destination.

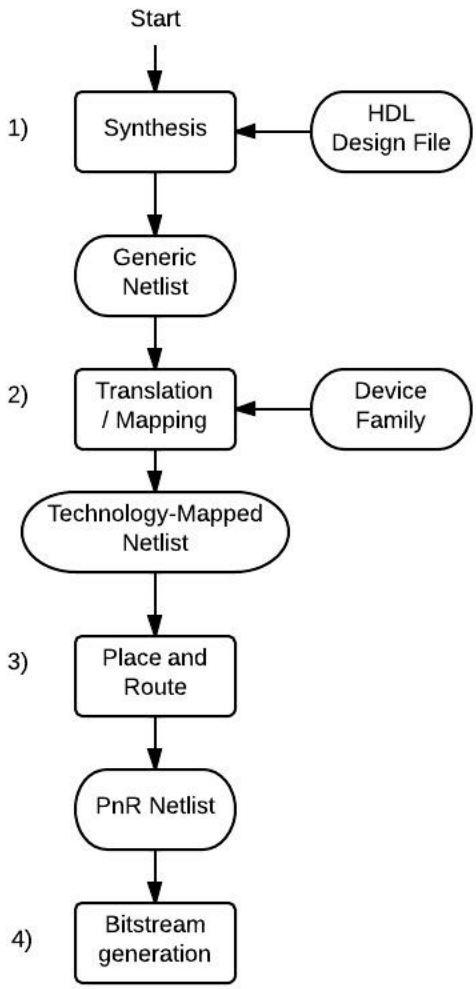
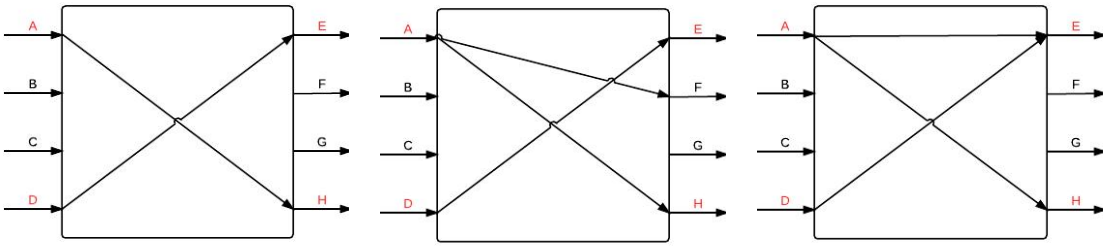


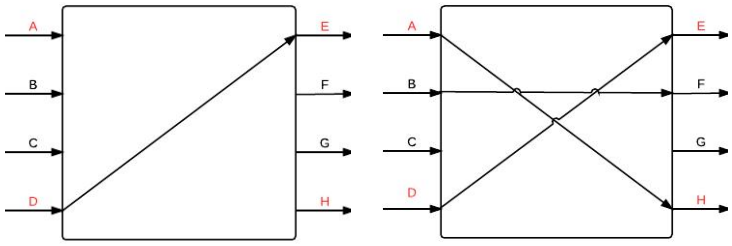
Figure 2.6.1: CAD flow for Xilinx FPGAs



(a) An example of a correct switchbox that has not been affected by SEUs

(b) A fan-out antenna may be formed by enabling a wire that is connected to an active port, and forcing a connection to an inactive port

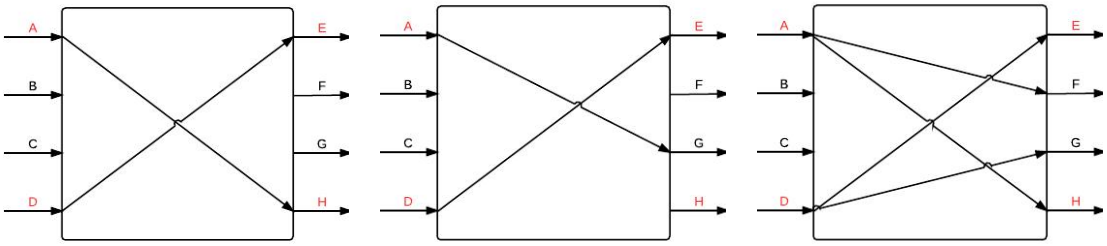
(c) A fan-out short may occur if the connection from one input port to the other output port was physically available to the switchbox



(d) A circuit break may occur if the SEU affected the memory bit

(e) An unused bridge may form if the SEU strikes this location

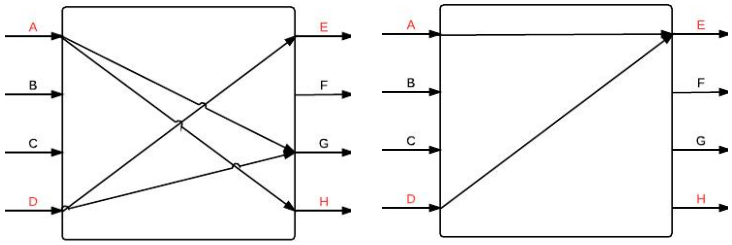
Figure 2.7.1: Possible errors with single SEU bitflips



(a) Example correct switch-box

(b) A reroute to an unused port may occur by disabling an existing connection, and enabling a new connection

(c) Two benign antennae may form



(d) In other cases, the antennae may form a short

(e) A true short (one not from a fan-out) may also be possible

Figure 2.7.2: Possible unique errors for Two bitflips

Chapter 3

Previous Work

We now review previous work that has been done to detect radiation errors on FPGAs. This chapter will discuss triple modular redundancy (TMR) and duplication-with comparison (DWC) and how they have been used to detect single event upsets on routing tracks. The chapter will also address built-in-self-tests, cyclic redundancy checks, and parity bit checks as other means to detect SEUs. The chapter will continue on with solutions that involve redesigning the FPGA by adding new hardware into the chip during the fabrication process. Finally, the chapter will conclude with a discussion about each solution's advantages and disadvantages.

3.1 Error detection and correction methods using standard FPGA hardware

3.1.1 Triple Modular Redundancy

Triple modular redundancy (TMR) is an error correction scheme used to counteract errors in the circuit. The main design logic is replicated into three independent copies. Each copy performs individual calculations and their results are compared to the two other modules. The majority result of the three modules becomes the final output of the circuit. TMR requires three times more resources to implement on an FPGA. The work done by [Pratt et al., 2007] proposes using TMR selectively on the most vulnerable sections of the design logic to eliminate the resource cost of full TMR.

3.1.2 Duplication with Comparison

Duplication with comparison (DWC) is an error detection scheme wherein the module under scrutiny is duplicated. Both modules will process their respective signals, and the result will be compared. If the results from both modules do not match, then an error is present. The work done by [Johnson et al., 2008] demonstrates the effectiveness of DWC on FPGAs. The main method this thesis suggests is comparing the configuration memory of the FPGA with a golden copy. Should the SEU affect the configuration memory, the cell data can be checked with a copy.

3.1.3 Cyclic Redundancy Check

Cyclic Redundancy Checks (CRC) are the current standard for error detection in FPGAs. A short check value is appended to the message being sent to a device. With regards to FPGAs, this message would be a slice of the configuration bits. The CRC check value is created from algorithms based on cyclic codes, which is a polynomial-based code sequence. Once the message is sent to the device, a fresh check value is computed on the destination and is compared with the check value of the message. An error will occur if the two values do not match.

This scheme is currently used by Altera FPGAs [Altera Corporation, 2013]. There is dedicated circuitry on all of the Altera FPGA device families. Though the CRC circuitry is present, it is primarily used to check the correctness of the configuration bits during the download. This method can be used during the chip's operation, but can take up to 50 ms to check the FPGA.

3.1.4 Built-in self-tests

Built-in self tests (BISTs) are another approach that have been used to detect errors in FPGA designs. BISTs are testing circuits that can be programmed onto the FPGA to check the condition of the chip. It is typically a separate design on the FPGA that overwrites the intended logic. BISTs are performed in intervals, or whenever an error is suspected. They can also be used for benchmark tests. The method proposed by [Stroud et al., 1998] is a BIST to test the interconnection points. The suggested test applies a sequence of 0s and 1s at one end of the wire segment, and expects the same result at the other end.

3.1.5 Parity Bit Check

Parity bit checks are error detection schemes that test data field bits. As its name implies, it is concerned with the number of ones and zeros in the data field, but not the representation of the data itself. The error detection comes from checking the parity of the message being sent. If the parity of the data changes overtime, it means that there is an error somewhere in the data. The work completed by Sun et al.[Sun et al., 2001] supports the idea that parity bit checking for interconnects are a valid solution. Their work implements a parity check using BISTs.

3.1.6 New CLB architecture

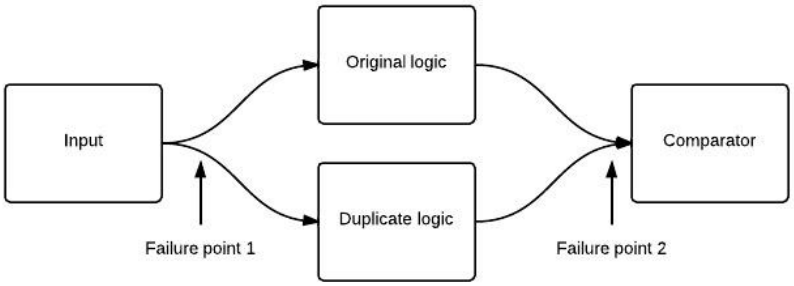
The idea suggested by [Reddy et al., 2005] is a new Configurable Logic Block (CLB) design for FPGAs. This proposal adds two extra 3-input LUTs to every CLB. By hooking these extra CLBs to the source and sink locations of the nets, analysis can be done on-the-fly.

3.1.7 New Switchbox architecture

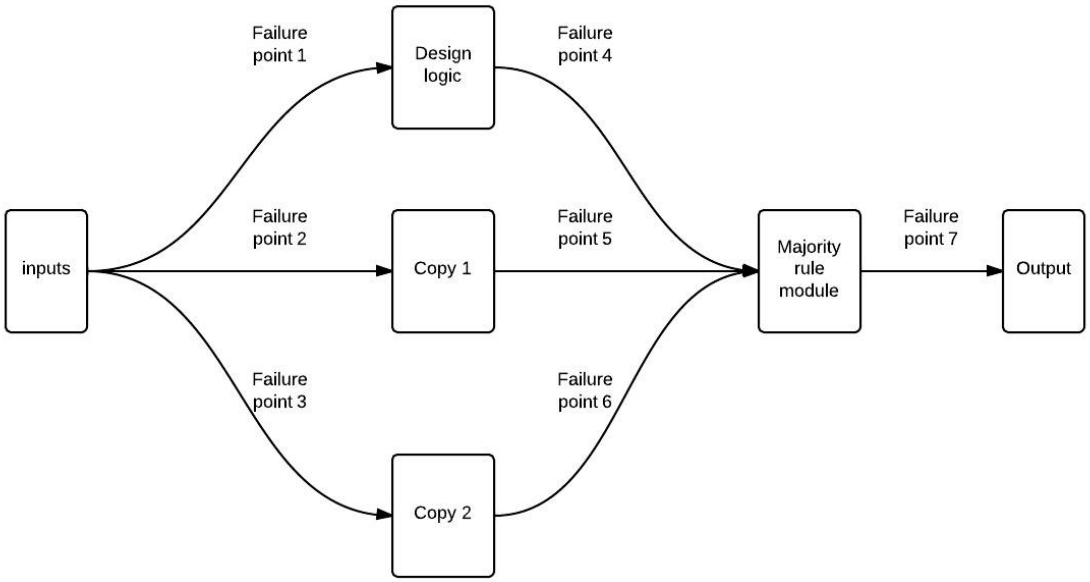
Another approach to detecting routing errors is to implement a new switchbox architecture. The method proposed by [Rohani et al., 2009] seeks to add detection circuitry into each switchbox on the FPGA in order to perform concurrent error detection. The detection method is a parity check of the configuration bits.

3.2 Comparison of error detection and correction methods

3.2.1 TMR and DWC



(a) Bottleneck problems with Duplicate with comparison



(b) Bottleneck problems with Triple modular redundancy

Figure 3.2.1: Bottleneck problems with DWC and TMR

TMR and DWC both have a problem with bottleneck failures. The FPGA will route the inputs to the module copies through the routing network. This routing

network is susceptible to SEU failures. If the input path, comparison path, or output path are affected by SEUs, TMR and DWC will fail to perform correctly. The new method can detect errors at the bottleneck failure points to ensure the modules are still operating correctly. Figure 3.2.1a shows the bottleneck failure for DWC, and Figure 3.2.1b shows the bottleneck failure for TMR.

3.2.2 BISTs

One of the downfalls to BISTs is that it requires the FPGA to be overwritten with the BIST circuitry. When a BIST is requested, the BIST circuit overwrites the existing FPGA circuit. Overwriting the circuit requires time, and means that the FPGA cannot perform its duties while undergoing maintenance.

BISTs are requested in regular intervals to ensure that the FPGA is not damaged. Due to the nature of SEUs, an interval check is not a desirable approach. The SEU may happen at any time, and must be constantly checked. A BIST-style error checking scheme would not be suitable because of the timeframe of the errors and the random occurrences of the error.

Another issue arises when the BIST is loaded onto the FPGA. SEUs are soft errors that will disappear when the FPGA is reprogrammed. If a BIST was loaded onto the FPGA, the error detected by the BIST will be removed. BISTs are not capable of detecting soft errors because the problem disappears as soon as the FPGA is overwritten with the BIST circuit.

3.2.3 Cyclic Redundancy Check

Though CRC is currently the standard for SEU detection, it is too slow to be effective in a nuclear generation environment. The Cyclone 5 family reports a minimum time of 3 milliseconds, and a maximum time of 1.53 seconds to verify errors with CRC [Altera Corporation, 2013]. The big drawback is that CRC does not perform concurrent checking. During continuous checking, the CRC will cycle through the entire Configuration RAM (CRAM) and then repeat the cycle. Due to the nature of SEUs, the errors can happen at anytime, anywhere. Should an error appear at an area that the CRC just verified, the scheme would have to wait one full cycle in order to detect the error. One advantage to CRC is that the current FPGAs in the marketplace have dedicated CRC hardware built into the chip.

3.2.4 Parity Bit check and new architecture designs

The work by [Rohani et al., 2009] uses a parity scheme to check for the actual error. The problem is that the checking of the SRAM cells is difficult without requiring explicit direct access to the cells. Due to intellectual property issues, information about the configuration bits is difficult to obtain. The solutions by Rohani and Reddy cannot be done without collaboration with the FPGA vendors. Another issue with using a new architecture design is that adding new pieces to the FPGAs would require non-recurring engineering costs (NRE) and would render the existing FPGAs obsolete.

3.3 Chapter summary

In short, the previous work detects SEUs effectively, but they require a significant cost in resources, time, or money. A solution will be presented in the next chapter that does not require a new FPGA design, and can detect the error in real time as the error occurs. This new detection method mitigates the bottleneck error in TMR and DWC.

Chapter 4

Switchbox Probing

This chapter discusses the new probing method that can be used to detect routing errors caused by SEUs in real-time. This method is designed to detect single SEU errors. The Wilton switchboxes of the Xilinx architecture can be manipulated to create probes in the datapath. By placing probes at key junctions in the datapath, the datapath can be monitored for irregularities caused by single event upsets. Once the probes are in place, the contents of the probe DFFs are fed into a LUT.

4.1 Taking advantage of the switchboxes

A routing net is a wire segment that connects the output port of a LE to an input port of another LE. On Xilinx FPGAs, every net passes through Wilton switchboxes. Figure 4.1.1 shows an example of a routing net. Figure 4.1.2 shows an example of two routing nets chained together. The Wilton switchbox allows for fanout connections to be made inside the routing. By selectively forcing the circuit to branch at the beginning of the net, and at the end of the net, the circuit can be observed. The

assumption is that there is no LE between the beginning and end of each net. It is also assumed that the value traveling through the net is not going to change during the clock cycle. Therefore, if a change is seen somewhere between the beginning and the end of the net, it represents an error with the routing network.

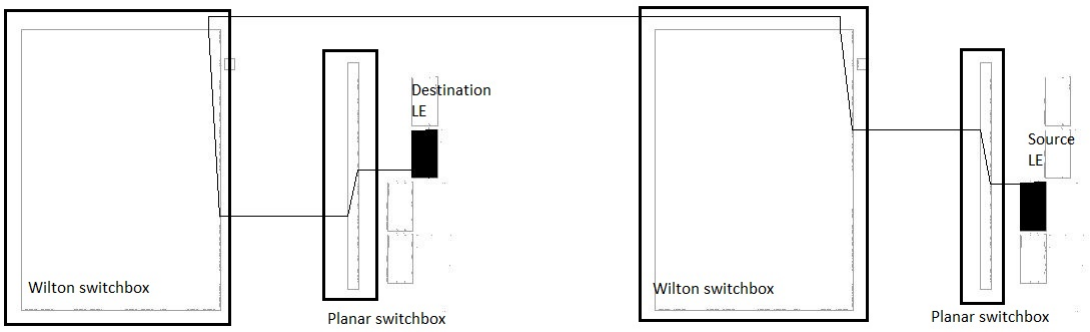


Figure 4.1.1: An example of a routed net on the Xilinx FPGA. Every net passes through the Wilton and planar switchboxes.

4.2 The testing circuit

Once the branching routes have been established, the routes are connected to LUTs. The logic inside the LUT is an XNOR gate that checks if the values at the beginning (source probe) and end (destination probe) have changed. Figure 4.2.1 shows an abstract example of how the probes are applied to a routing path.

If a circuit is only partially routed, it means that there is a break in the path. This path becomes an antenna, and the LE at the end of the antenna will be affected. Due to electronic metastability, the value sensed by the antenna may drift to logic 0, or logic 1. If the drift happens, the XNOR gate can still perform a calculation. So long as the XNOR gate calculates a logic 1, there is no problem with the path. However, this assertion is only true for the path covered in the probing zone. Table

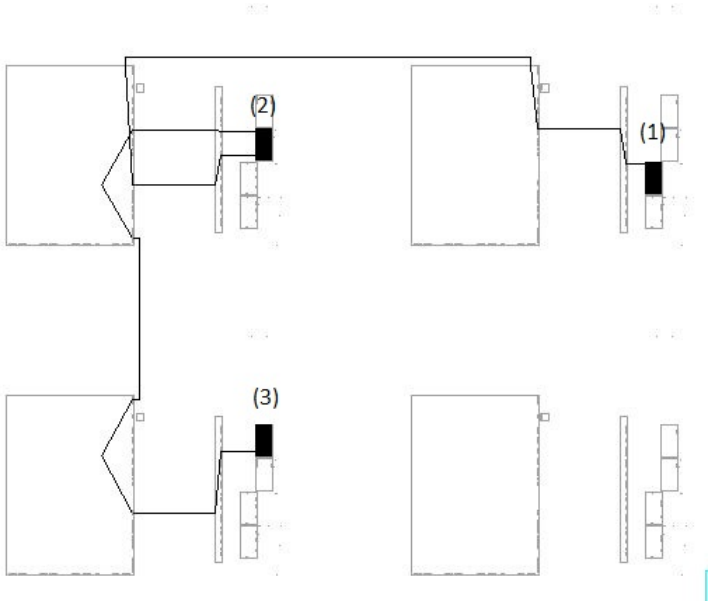


Figure 4.1.2: This is an example of two routing nets. One net connects LEs (1) and (2). Another net connects LEs (2) and (3).

Truth table for XNOR		
probe 1	probe 2	probe 1 XNOR probe 2
0	0	1
0	1	0
1	0	0
1	1	1

Table 4.1: XNOR truth table for detecting routing errors

4.1 shows the truth table that would be implemented inside the LUT. The probing zone will be discussed later in the chapter.

Figure 4.2.1 shows how the probes are hooked up to the circuit path. The fanout starts at the switchbox closest to the source. Once this fanout is placed, the divergent path is completed by leading it into a DFF. The fanout path for the destination is created in the same manner. The fanout for the destination probe leads from the entrance path of the switchbox closest to the destination, and into another DFF.

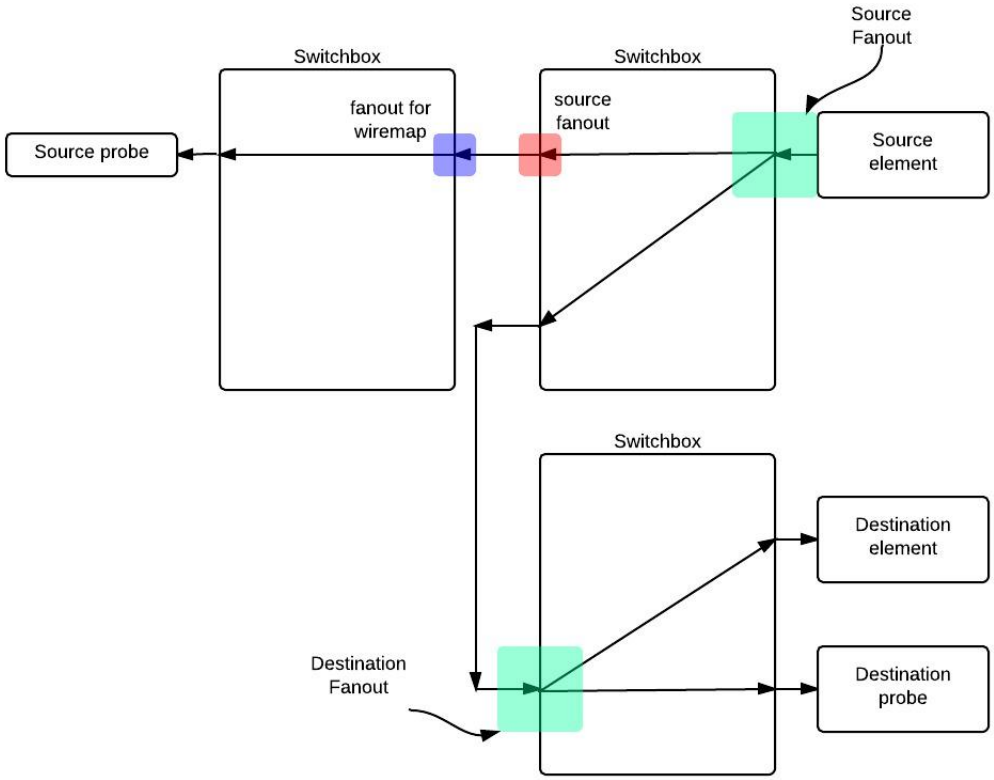


Figure 4.2.1: Circuit routing

4.3 Using the LUTs

Once the probes are attached to DFFs, the data analysis is performed using the LUT. The analysis must compare a net's source to its corresponding destination. The Virtex 4 FPGA employs 4-input LUTs in its LEs. The probes can be arranged in one of two fashions: One source and three destinations (see Figure 4.3.1), or two sources and two destinations (see Figure 4.3.2).

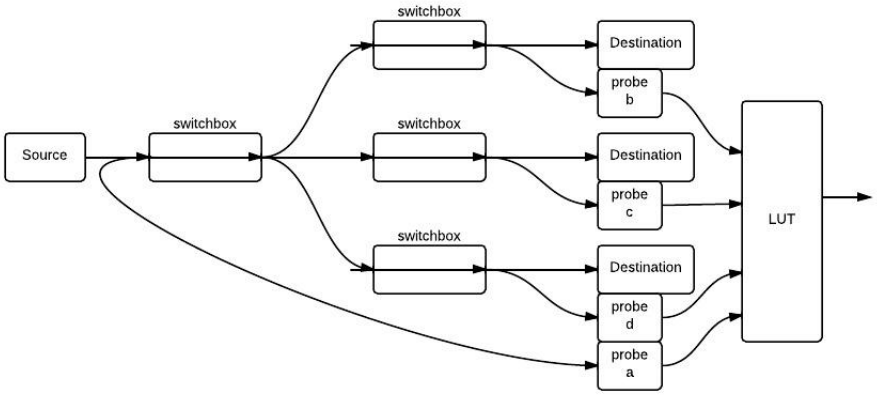


Figure 4.3.1: One source and three destinations

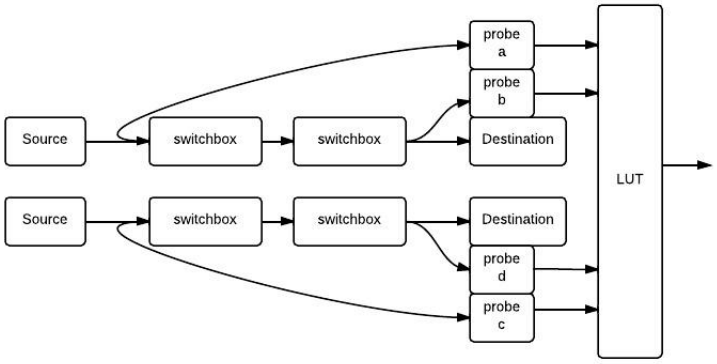


Figure 4.3.2: Two sources and two destinations

4.4 Inside the LUT

The logic function for the 4-input LUT will change depending on which circuit hookup is used.

The function required for circuit 1 will be:

$$(A\&B\&C\&D)\|(\neg A\&\neg B\&\neg C\&\neg D)$$

Truth table for logic with no faults										
Truth Table for circuit 1						Truth Table for circuit 2				
a	b	c	d	output		a	b	c	d	output
0	0	0	0	1		0	0	0	0	1
0	0	0	1	0		0	0	0	1	0
0	0	1	0	0		0	0	1	0	0
0	0	1	1	0		0	0	1	1	1
0	1	0	0	0		0	1	0	0	0
0	1	0	1	0		0	1	0	1	0
0	1	1	0	0		0	1	1	0	0
0	1	1	1	0		0	1	1	1	0
1	0	0	0	0		1	0	0	0	0
1	0	0	1	0		1	0	0	1	0
1	0	1	0	0		1	0	1	0	0
1	0	1	1	0		1	0	1	1	0
1	1	0	0	0		1	1	0	0	1
1	1	0	1	0		1	1	0	1	0
1	1	1	0	0		1	1	1	0	0
1	1	1	1	1		1	1	1	1	1

Table 4.2: Truth table for LUT functions

The function required for the circuit 2 configuration will be:

$$((A\&B)\|(\neg A\&\neg B))\&((C\&D)\|(\neg C\&\neg D))$$

The truth tables for the two functions are shown in Table 4.2. They show that the function will be true only when the source matches the destination. This truth table and the method is designed to detect single SEU errors.

Truth table for logic with 1 fault										
Truth Table for circuit 1						Truth Table for circuit 2				
a	b	c	d	output		a	b	c	d	output
u	0	0	0	u*		u	0	0	0	u*
u	0	0	1	u*		u	0	0	1	u*
u	0	1	0	u*		u	0	1	0	u*
u	0	1	1	u*		u	0	1	1	u*
u	1	0	0	u*		u	1	0	0	u*
u	1	0	1	u*		u	1	0	1	u*
u	1	1	0	u*		u	1	1	0	u*
u	1	1	1	u*		u	1	1	1	u*

Table 4.3: Truth table for logic with 1 fault. u is for a metastable input. u* is for an undetermined output.

4.5 Single SEU striking a net

In a single SEU strike, the error may be a circuit break, short, or antenna. In circuit breaks, one of the probes will have a metastable reading. Table 4.3 illustrates the truth table for a metastable input. All of the entries in the table are marked “u” or “ u* ” for unknown. This is because the final result is determined by the metastable value. The error signal is tripped only when the output of the XNOR is not logic 1. If the output of the XNOR gate is not 1, then there is an error with the routing net. The results of the truth table for circuit 1 will be the same regardless of where the SEU strikes within the probing zone. So long as there is a disconnect, the output of the function will show the break.

4.6 Probing zone

The implementation of the probing method has a gap in detection coverage. Figure 4.6.1 shows the detection zone (henceforth known as the probing zone). Due to the

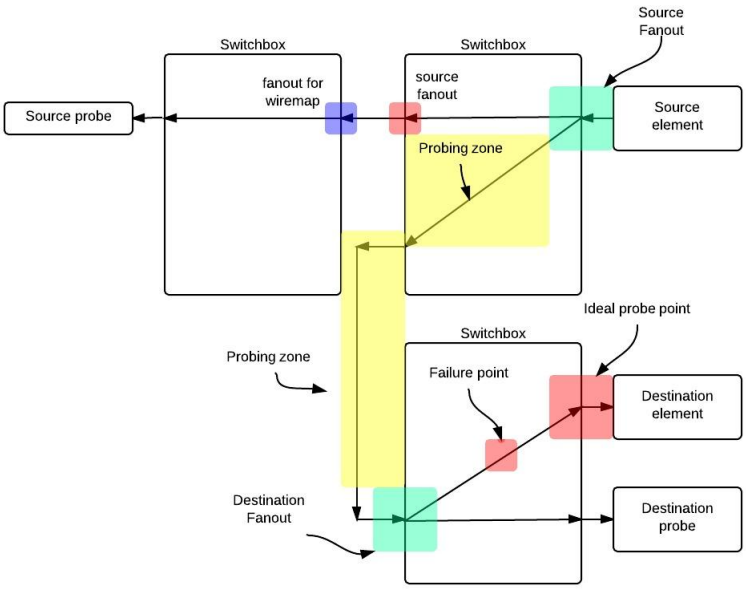


Figure 4.6.1: Probe zone

nature of the Wilton switchbox ports, the ideal probe point (also seen in Figure 4.6.1) cannot branch out. The closest port that can branch out to be probed is the destination fanout point. This point is the first instance of the datapath entering the switchbox. The gap in coverage is the PIP that connects the destination fanout port to the port that reaches the destination element. The probe method will not be able to detect errors that happen outside of the probing zone. This gap in coverage is mitigated by the use of triple modular redundancy. If the failure point is not connected, the result will be caught by the other two modules in the system.

4.7 Chapter summary

In summary, every routing net in the Xilinx architecture travels through Wilton switchboxes. The Wilton switchboxes have ports that can direct the data to multiple

paths. By branching the datapath at the first switchbox, and the last switchbox, the value of the data in the path can be monitored for changes. If the value at the end of the datapath does not match the value at the beginning, then there is an error with the routed net. This method is implemented on the netlist after the circuit has been placed and routed. In the worst case scenario, every datapath of the circuit is probed. Identifying all of the source and destination ports of the nets through inspection is tedious. There is a gap in the coverage when using the probe method. This gap is between the destination probe point and the destination element. Errors occurring on this gap will not be detected if the probe method is used by itself. This gap in coverage is mitigated by the use of triple modular redundancy. The next chapter introduces a software tool that can automate the probe creation process using the Xilinx Design Language.

Chapter 5

Circuit Insertion Tool

This chapter outlines the operation of the Circuit Insertion Tool (CIT); the software tool used to add the probes after the place-and-route stage of FPGA development. The chapter explains each module in the CIT and provides a module guide for each module. This tool scans the netlist and inserts branches at the switchbox closest to the signal's source, and at the switchbox nearest to the signal's destinations.

5.1 Expected Changes

5.1.1 Architecture

Every FPGA family is different from the others. The differences range from the number of LEs, to the arrangement of the LEs, the switchboxes that are used. This also means that the naming conventions used in the XDL file will also change. Port names such as "W2BEG9" or "BYP_INT_B0" may appear in the Virtex-4 family, but are not guaranteed to appear in the Virtex-5 family of FPGAs. New fan-out and

wiremap libraries will have to be constructed in order for the CIT to work on another family. The fan-out and wiremap libraries were designed to be separate entities in anticipation of this likely change.

5.1.2 Dead zone

The dead zone of an FPGA is the coordinate section of the chip that does not contain any LEs. This dead zone outlines the large memory block on the chip and the outer boundary of the chip. This data is needed because the routing algorithm relies on the coordinate system to determine a suitable path. The FPGA dead zone changes from chip to chip.

5.2 Unlikely Changes

5.2.1 XDL format

The format of the XDL file does not appear to change when the same circuit is compiled using different FPGA families. There are four sections to the file:

1. A brief description of the syntax for section 2
2. The instantiation section. This lays out the FPGA elements being used by the design
3. A brief description of the syntax for section 4
4. The routing section. Each net path is organized into a list of PIP statements

```
net "c_2" ,
  outpin "c_2" YQ ,
  inpin "c<2>" O ,
  pip CLB_X14Y41 YQ_PINWIRE0 -> SECONDARY_LOGIC_OUTS4_INT ,
  pip INT_X14Y41 SECONDARY_LOGIC_OUTS4 -> OMUX12 ,
  pip INT_X15Y42 OMUX_NE12 -> IMUX_B25 ,
  pip IOIS_LC_X15Y42 IMUX_B25_INT -> IOIS_O10 ,
  pip IOIS_LC_X15Y42 IOIS_O10 -> IOIS_O_PINWIRE0 ,  #  _ROUTETHROUGH:D1:OQ
  "XDL_DUMMY_IOIS_LC_X15Y42_OLOGIC_X1Y85" D1 -> OQ
  pip IOIS_LC_X15Y42 IOIS_O_PINWIRE0 -> IOIS_O0 ,
  ;
```

Figure 5.2.1: An example of a net, described using a series of PIP statements

5.2.2 PIP statements

The routing path of the net is laid out in PIP statements. Figure 5.2.1 is an example of a net. The statements are arranged in alphabetical order. The FPGA circuit can be changed by adding and removing statements.

5.2.3 XDL circuit checks

The XDL tool performs a cursory check of the XDL file during the conversion of the XDL back into an NCD file. This check looks for routing faults such as short-circuits, incomplete paths, and fan-out antennae. If a circuit is not properly routed, the NCD will not be generated.

5.3 CIT Design

The Circuit Insertion Tool (CIT) is the tool that will add extra circuitry to the pre-assembled circuit from the HDL level. Figure 5.3.1 shows the revised CAD flow that incorporates the CIT. The tool is inserted after the Place and Route (PnR) stage, but before the configuration data is downloaded onto the FPGA.

1. Synthesis, TM, and PnR remain the same

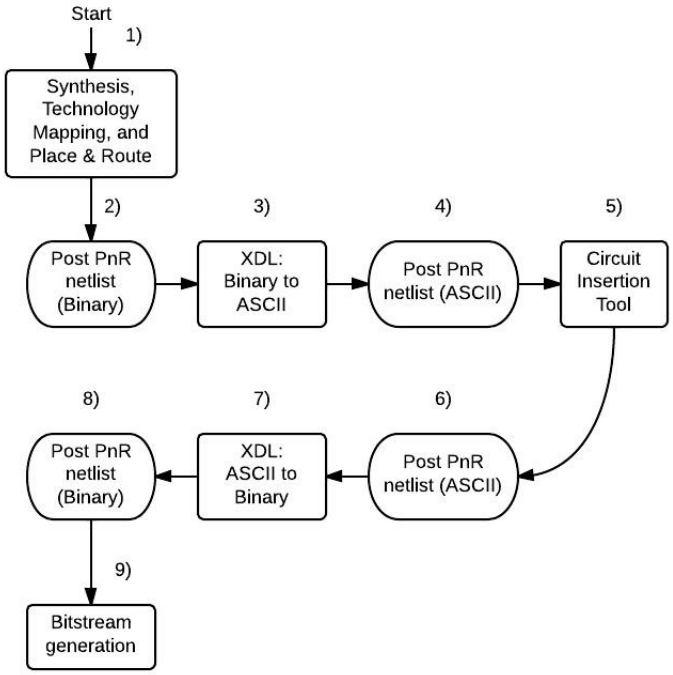


Figure 5.3.1: The new CAD tool flow to incorporate the circuit insertion tool

2. After PnR, the binary netlist (NCD) is generated.

3. In the toolflow without the CIT, the NCD is encrypted and downloaded to the device. Instead, the XDL tool will translate the NCD to ASCII form (XDL file).

4. The output of the built-in XDL program generates an ASCII netlist.

5. The CIT takes the XDL file and manipulates it, producing a modified XDL (point 6).

6. The new XDL file is ready to be fed back into the Xilinx CAD flow.

7. The modified XDL is then translated back into binary (NCD) using the built-in XDL program.

8. The new binary is encrypted using the Xilinx process.

9. The new configuration file is ready to be downloaded to the device as a bit-stream.

The CIT manipulates the ASCII netlist produced by the Xilinx XDL tool. In order to add extra testing logic, the new statements are inserted into the XDL file. This modified XDL file is fed back into the Xilinx ISE toolflow by converting the XDL back into the NCD file. During this conversion, the XDL is checked for errors in the routing. If there are antennae, or incomplete connections left in the XDL, conversion will fail and the NCD will not be created.

5.4 How the CIT works

Figure 5.4.1 shows the organization of the CIT's design. The structure of the tool was dissected into manageable modules for incremental design and test. The list below describes the software:

1. The original XDL file is read by three modules and will produce three separate library files based on their function:

   - "Locate all occupied LEs"
   - "Locate all occupied routes"
   - "Determine probe points"

2. The newly made library files, as well as the FPGA constraints libraries are used by the "Route the probes" module.

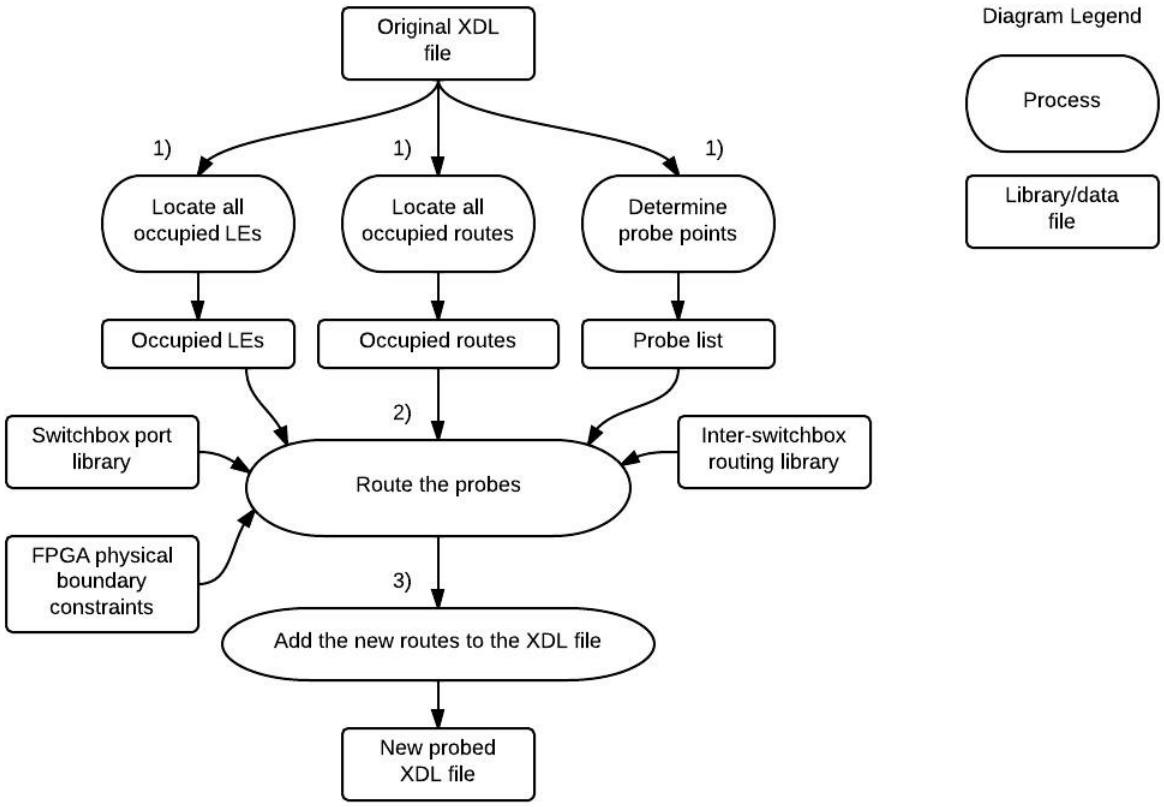


Figure 5.4.1: The CIT internals

3. After the "Route the probes" module has completed a successful run, the new routes are added to a new XDL file produced by the CIT.

5.4.1 Module secrets

Table 5.4 describes the modules of the CIT and their secrets.

5.4.2 Locating all occupied LEs and occupied routes

The "Locate all occupied LEs" and "Locate all occupied Routes" modules parse the XDL file and looks for the defined logic elements and PIPs being used in the circuit.

Module name	**Module secret**
Locate all occupied LEs	The behaviour to extract the LE locations.
Locate all occupied routes	The behaviour to extract the route locations.
Determine the probe points	The behaviour to extract the probe points.
First-stage routing	The behaviour to route a probe using one switchbox.
Secondary routing	The behaviour to route a probe using multiple switchboxes.
Mix-port probing	The behaviour to route a probe using a mix port as a probe point.
Add the new routes to the XDL file	The behaviour to add new PIPs to the XDL file.

Table 5.4: Module secrets table

They form the Occupied LEs and the Occupied Routes library respectively, which will be used in another module.

5.4.3 Determining probe points

The individual wire paths (nets) are grouped by the alphabetical order. Inside each net are the list of PIPs that make up the full path. These PIPs are listed in ascending numerical order based on their X and Y coordinates, and then by alphabetical order using the names of the ports within the same coordinate. Certain ports in the switch-box will always be used as an input or output. The probe point detection module searches for the specific ports and places them in a list for the routing algorithm to

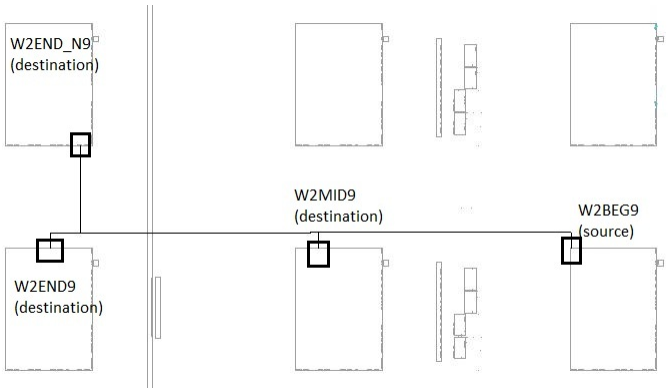


Figure 5.4.2: This image shows how the port W2BEG9 can connect to other switchboxes using the wire associated with that port.

systematically work through each port. For source ports, this is a simple task since all source ports heading into the switchbox from the source element will always have fan-out. The destination ports are found by backtracking through the list of occupied PIPs to determine which connection along the line can serve as a branching point. The backtracking starts at the destination element's port. The algorithm reconstructs the list and checks each port for its eligibility.

5.4.4 Inter-switchbox routing library

Every port that heads out of the switchbox has wire segments that connect to other switchboxes. The Inter-Switchbox Routing Library (ISRL) contains the connection coordinates for every port that leaves the switchbox in the Xilinx Virtex 4 architecture. Every port uses a different wire segment. Figure 5.4.2 shows the ISRL entry for the port "W2BEG9" located at the top-left corner of the switchbox.

The internal routing details are not documented by any external source, and must be extracted manually using the XDL file in conjunction with the FPGA editor's graphical interface.

Figure 5.4.3: An example of a wiremap. One port's exit can connect to other ports at different locations. The path is marked in black.

5.4.5 Switchbox port library

Every port in the Wilton switchbox has a specific name. The XDL file uses these names to create the routed net. Each port has a data direction associated with it. Some ports can only send data out of the switchbox (fan-outs), while other ports can only receive data into the switchbox (fan-ins). There are also ports that can send and receive data. Ports that can send and receive data are called mix ports, because their purpose is to redirect the data inside the switchbox. The switchbox port library contains all of the switchbox port names, and their associated fan-ins and fan-outs. This information was extracted using the graphical user interface of the Xilinx ISE tool. Official documentation regarding the switchbox port names is not available to the public. An example of a switchbox port library entry can be seen below. Figure 5.4.4 shows an example of a port's fan-out selection. Figure 5.4.5 shows an example

of a port's fan-in selection

W2BEG9 is a fan-in port that can receive data from:

- E6END9
- W2END9
- W2END7
- W6END9
- S6END_S0
- S2END_S0
- OMUX15
- OMUX_SW5
- N2END8
- N6END9
- S2MID9
- N2MID9
- S6MID9
- N6MID9

E2END9 is a fan-out port that can send data to:

- E2BEG9
- E2BEG7
- N2BEG8
- S2END8
- CE_B3
- CE_B1
- BYP_INT_B7
- BYP_INT_B5
- IMUX_B31
- IMUX_B27
- IMUX_B23
- IMUX_B19
- IMUX_B15
- IMUX_B11
- IMUX_B7
- IMUX_B3

5.4.6 FPGA physical boundary constraints

This module lists the boundaries of the FPGA in terms of switchbox coordinates. The FPGA has a finite number of switchboxes, and the CIT is only aware of the FPGA's boundaries by comparing the X and Y coordinate values with the entries

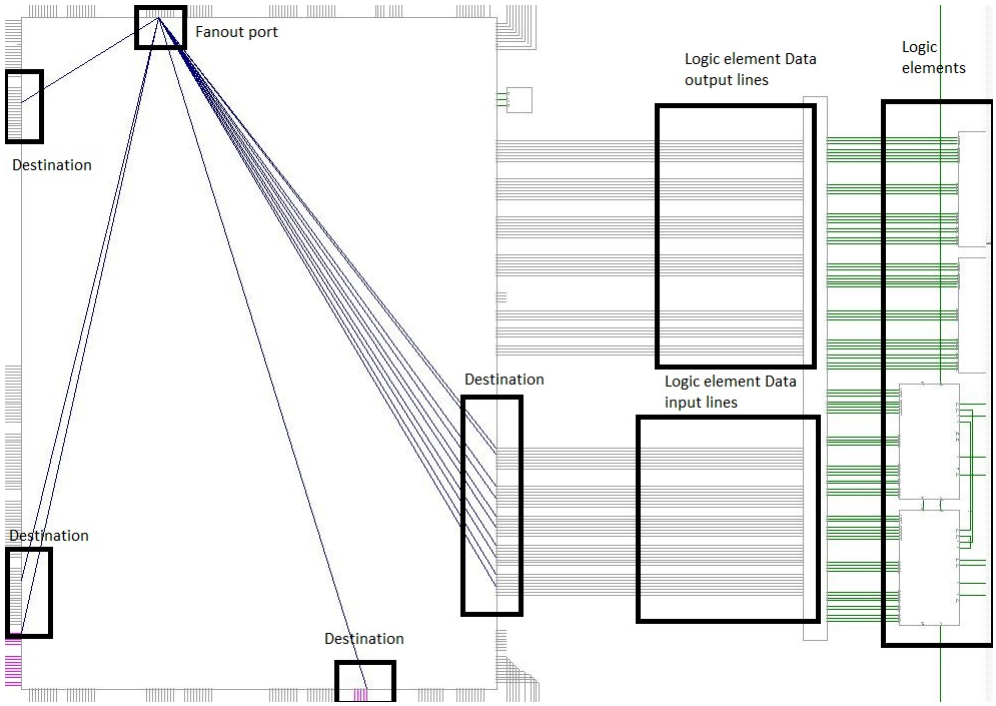


Figure 5.4.4: An example of the possible fan-out connections that one port can have.

inside this library file. Additionally, the FPGA contains a region of dead space that does not contain LEs. When a route is suggested, the coordinates of the new location are checked with the coordinate checker module.

5.4.7 Port probing

The bulk of the CIT is in the port probing modules. The port probing modules are responsible for routing the probe points to a free LE. There are multiple stages to this routing in case the algorithm cannot route a probe on the first pass. There are three main algorithmic modules that make up the routing logic: direct routing, first stage routing, secondary routing, and mix-port routing.

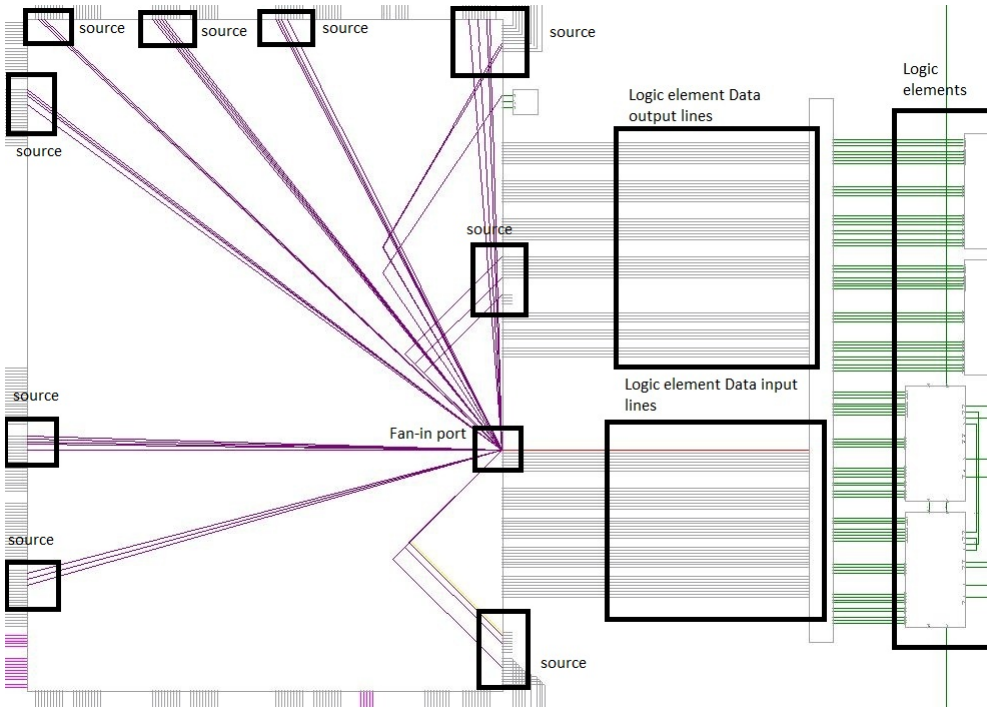


Figure 5.4.5: An example of the possible fan-in connections that one port can have.

First Stage Routing

First stage routing is a minimal effort method of determining if a port can be probed. If a port can make contact with a free LE that is either inside the same X and Y coordinate, or has to jump once to another coordinate, the probing is successful. If the connection is made within the same X and Y coordinate, this is called a direct routing. If the route leaves the original coordinate, this is called a first stage route. If there are no free paths for the port to travel, or the routing requires more than one coordinate jump, first stage routing will fail. Figure 5.4.6 is a flowchart of how first stage routing is performed. The list below describes the points in Figure 5.4.6.

1. A port is fetched from a pre-processed list of probes. The fan-out for the port is fetched and is checked to see if the port can reach a free LE in the same

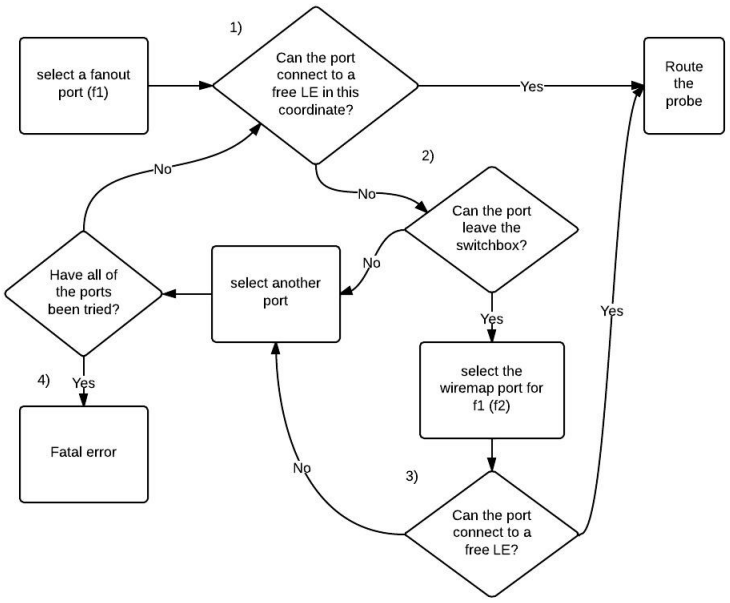


Figure 5.4.6: First stage routing

X and Y coordinate. This check is called the shortcut check, as it can bypass the majority of the checking logic and easily find a probe. The fan-out port candidate that is selected from the list is called F1. If it is successful, then the probing is complete.

2. If the shortcut check fails, then a check is performed to see if F1 can make connections to other switchboxes. If it cannot, another fan-out port is selected. If all of the ports have been tried, the First stage routing module returns a fatal error.

3. If F1 can leave the switchbox, the corresponding port at the other end (F2) will be checked to see if it can connect to a free LE at the new coordinate location. If the connection can be made, the probe is routed. If the connection cannot be made, another F1 port is chosen.

4. In the event that the backtracking has been totally exhausted, the algorithm declares a fatal error. In order to reach this point, all of the source fan-outs have to be tried. If all of the fan-outs are occupied, then the algorithm returns a fatal error.

Secondary Routing

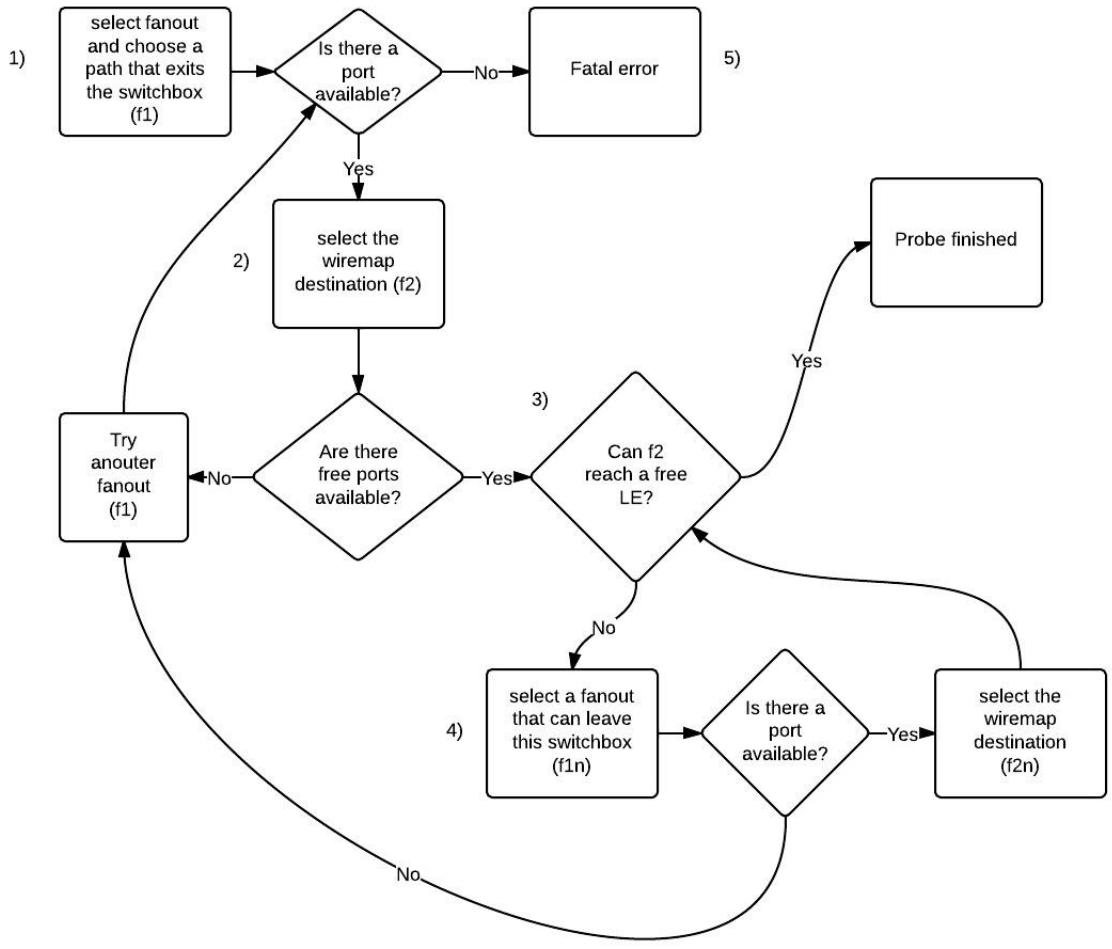


Figure 5.4.7: Secondary Routing

Secondary routing is performed when first stage routing fails. Where first stage

routing will stop after attempting to route to an empty LE at one adjacent switchbox coordinate, secondary routing will continue on. Secondary routing will search through multiple switchbox coordinates until a free LE can be found. The list below describes the secondary routing process shown in Figure 5.4.7:

1. The fan-out list of a probe point list is fetched, and a port is picked from the list. This port will be called F1. F1 is checked to see if it can leave the switchbox.

2. If there is a suitable F1 port, the corresponding port at the other end of the wire is fetched. This port is called F2. If there is no F1 port, the process returns a fatal error. If all of the ports on the wire are occupied, another F1 port is fetched.

3. F2 is checked to see if it can reach a free LE. If the check is successful, the probe is routed.

4. If the check is unsuccessful, the fan-out data for F2 is fetched and a port that can leave this new coordinate will be chosen. This new port is called F1n. If no F1n can be chosen, the process chooses another F1 port from the beginning. Else if an F1n port can be chosen, the port at the other end of the wire connected to F1n is chosen. This port is named F2n, and will be checked if it can reach a free LE at the new coordinate. This step repeats until either a free LE can be reached, or there are no more fan-outs available to leave the switchbox coordinates.

5. In the event that all fan-outs at the initial probe point have been tried, the algorithm returns a fatal error and no probing is completed.

Mix port probing

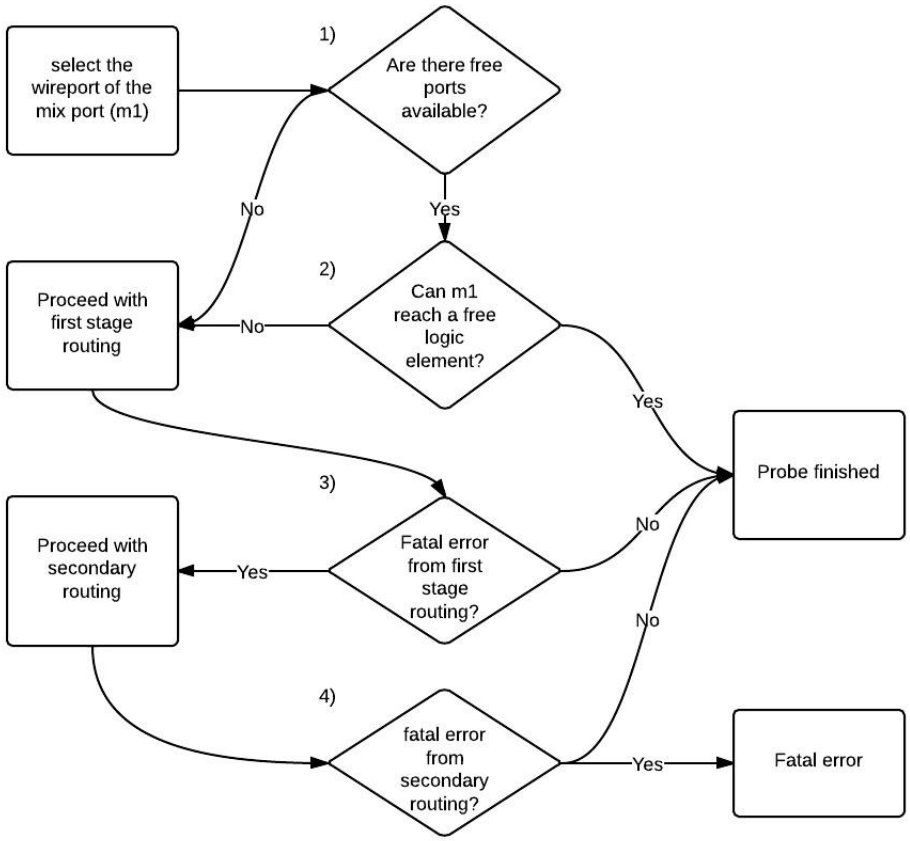


Figure 5.4.8: Mix Port routing

The Xilinx Placement and Routing algorithm and switchbox topology will force some of the wires to take an indirect route in a switchbox. A single net can bounce off of several ports in the switchbox before arriving at the intended destination. Figure 5.4.9 shows an example of this ricochet route. If a port can send data out of the switchbox and receive incoming data, this port is a mix port. Some probe points are mix ports, which require additional instructions to route correctly. The list below outlines the process shown in Figure 5.4.8.

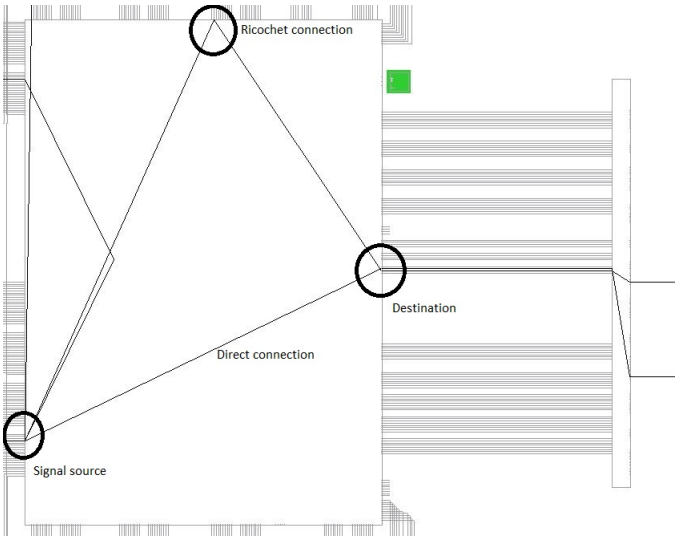


Figure 5.4.9: Ricochet routing

1. A mixport can leave and enter a switchbox. Therefore, the process will utilize this fact and fetch the corresponding port from the other end of the wire. This port will be labelled M1.

2. M1 will be checked to see if it can connect to a free LE. If the check is successful, the probe is routed.

3. If the check is unsuccessful, the process turns to the steps for first stage routing.

4. In the event that first stage routing is unsuccessful, the process turns to secondary routing. If secondary routing fails, the process returns a fatal error.

5.4.8 Add the new routes to the XDL file

The probe routing from the routing process is stored in a dictionary. The organization of the PIPs inside the netlist were organized by groups of datapaths. Each PIP in the datapath is arranged by ascending X and Y coordinates. Each PIP grouped by

X and Y coordinates is then arranged in alphabetical order based on the name of the port that is being used. The ordering of the PIP statements does not affect the overall routing of the path. The XDL creation module simply adds the new route to the existing nets and writes the information to a new file.

5.5 Comparator fitting

The current form of the CIT only establishes the probes and routes them to a free LE of the FPGA. The other step to the probe method is to connect the source and destination probes to a LUT. This feature lies outside of the scope of this thesis because it requires a fully-developed placement and routing algorithm. The issues concerning the placement would be:

- The location of the LUT for the source and destination probe DFFs to be compared.
- The additional FPGA occupancy carried by the routing of the probes to the comparison LUT.

5.6 Chapter summary

In summary, the CIT inserts branching statements into the XDL ASCII netlist. The tool scans the XDL ASCII netlist and determines each datapath's source and destination coordinates. The CIT determines the suitable branching paths based on the switchbox ports that are currently occupied by the circuit. If a port can be branched, the tool will insert a PIP statement into the netlist. After all of the statements have

been placed into the netlist, a new XDL file is created. This new netlist is fed into the Xilinx ISE toolflow. The advantages and disadvantages of the probes will be discussed in the next chapter.

Chapter 6

CIT Analysis

This chapter discusses the advantages and disadvantages of using the switchbox probing method to detect single event upsets on routing tracks. The resource costs, and the limitations of this method will be addressed in this chapter. This chapter also explains the procedure to test the concept of the probing method. A net in the routing netlist represents a connection between two or more LEs. The end connections of the nets can either terminate at a LE, regardless if the routing heads to a DFF, or a LUT. At every location where data is stored, there must be a probe attached to the routing pathway. This detection works within one clock cycle because the source and destination probes are compared directly from the net. This proposed solution brings into question the amount of additional resources that are required.

6.1 Resource allocation

Every net in the circuit requires two DFFs to store the signal data for the initial probing stage. The source DFF stores the value at the beginning of the net, and the

destination DFF stores the value of the signal. In order to process the probe data, LUTs will be used.

The equation below summarizes how many DFFs are needed to implement the probe stage.

$$(R = 2 * (NET))$$

- NET = the number of nets in the circuit
- R = the total number of DFFs used in the probing stage

If the results of the probes were to be compared, LUTs and DFFs will have to be used. The Virtex-4 FPGA uses 4-input LUTs. The LUT requires one of the inputs to be from the source probe, whereas the three other inputs to the LUT are the destination probes from the same datapath.

$$(X = Y = (\lceil (destination/3) \rceil)$$

- X = the number of DFFs required for the comparison logic
- Y = the number of LUTs
- destination = the number of destination probes for a single datapath

If the datapath has only one source and one destination probe, then two nets can be analyzed using the same LUT.

The CIT probes can monitor every signal that is routed through the switch matrix. In the event that every signal needs to be checked, the maximum occupancy of the circuit is roughly 33 % of the FPGA.

6.2 Port fan-out

The number of fan-out connections for the source and destination ports limits the configurations of the probes. If the design logic is large enough such that the fan-out ports of a source port are occupied, then the CIT cannot create a probe, even if there is unallocated space on the FPGA.

For example, port SECONDARY_LOGIC_OUTS7 is a source port to the LE. This port can fan-out to the following ports:

- OMUX0
- OMUX1
- OMUX2
- OMUX3
- OMUX4
- OMUX5
- OMUX6
- OMUX7
- OMUX8
- OMUX9
- OMUX10
- OMUX11
- OMUX12
- OMUX13
- OMUX14
- OMUX15

Each of these ports will have their respective wiremap destinations at other switch-boxes. If these fan-out ports are located at the same X and Y coordinates as the source port, or the wiremap destinations to these ports are occupied, then there is no space for the signal to branch.

6.3 Applicability

6.3.1 TMR and DWC

The CIT probes can be applied to the entire circuit, but the resource cost would be at least two DFFs for every routing net. Instead, the CIT probes should be applied to selective parts of the design logic. This method would solve the problem with TMR and DWC. If the CIT probes were applied to the bottleneck failure points of TMR (shown in Figure 6.3.1band DWC (shown in Figure 6.3.1a), the failure points would no longer be a problem.

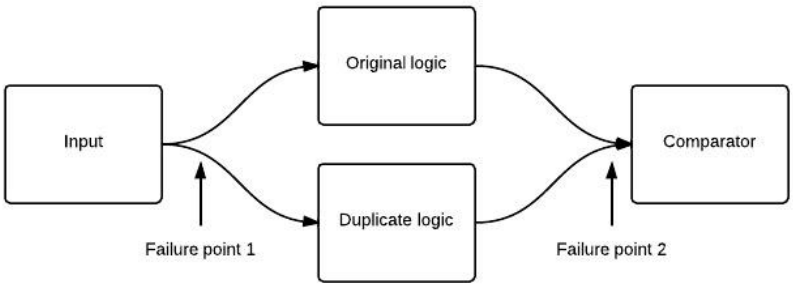
6.3.2 Nets terminating at LUTs

If there is a circuit path that contains a long string of LUTs hooked up in sequence, this will not cause a problem for the CIT probes. This is because each net of the Xilinx FPGA terminates at a LE, regardless if the DFF in the LE is being used. Therefore, it is possible to probe each wire segment connecting the LUT chain. This fact is illustrated in Figure 6.3.2.

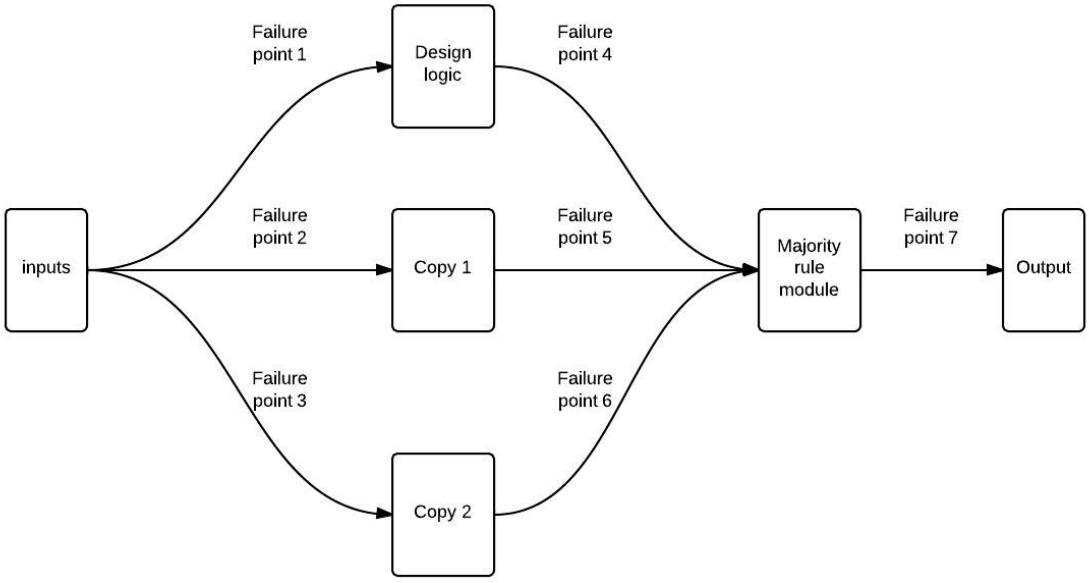
6.4 Details about the CIT probes

6.4.1 Errors occurring outside of the testing zone

The limitations to the probing circuit occur when a problem arises outside of the testing environment. Every circuit in a Xilinx FPGA passes through a planar switchbox before it connects to the wider switchbox network. The circuit also has to pass through another planar switchbox before leading into a LE. The testing environment



(a) Bottleneck problems with Duplicate with comparison



(b) Bottleneck problems with Triple modular redundancy

Figure 6.3.1: Bottleneck problems with DWC and TMR

is limited to the first instance of the route leaving a switchbox to the last instance of the circuit entering the destination planar switchbox. Figure 6.4.1 shows a typical routed path.

The problem with the planar switchbox is that they too are susceptible to SEUs. The only error that a planar box can experience is a circuit break. Since the CIT probes rely on the fan-out property of the Wilton switchboxes, the planar switchboxes cannot be probed.

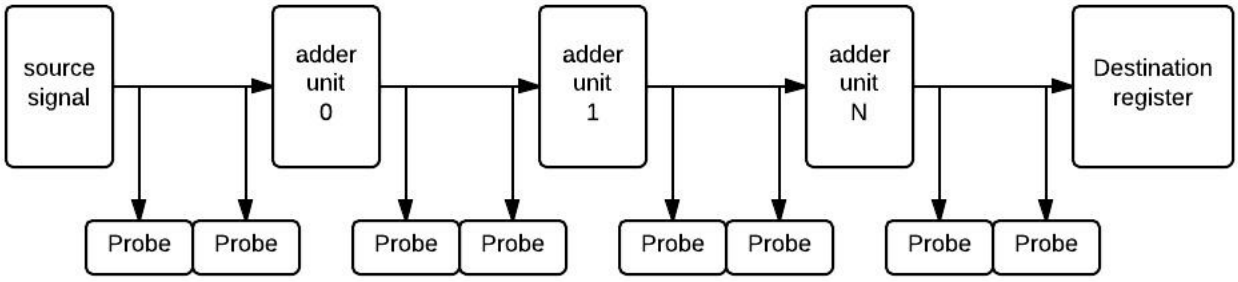


Figure 6.3.2: A possible probing example with an adder circuit

A second problem is with the adders of the LEs. Each configurable logic block (CLB) coordinate contains four LEs that can connect to each other without passing through a switchbox. The carry-in and carry-out connections in the logic cluster are examples of these connections. The carry-in to carry-out connection is still controlled by a PIP. An example of this connection can be seen in Figure 6.4.2.

The problems are solved by the fact that the circuit will eventually have to connect through the switchbox network. The error will be caught if it enters the switchbox grid. In the example of the adder circuit, if a disconnected value appears in one of the carry lines, the adder may add the unknown value, which will propagate through the other adders. The output of the adder will eventually head to a DFF. Once this unknown value heads into a DFF, the error will be caught because the unknown value will cause the comparator to reach a metastable state (please refer back to Table 4.3 in Chapter 4 for the behaviour). If the planar switchbox fails, the error will also be seen due to metastability (Table 4.3). The extreme case would be at an I/O port. If the I/O port is an input, the error will be caught by the network. If the I/O port is an output, the error will be seen by the external logic.

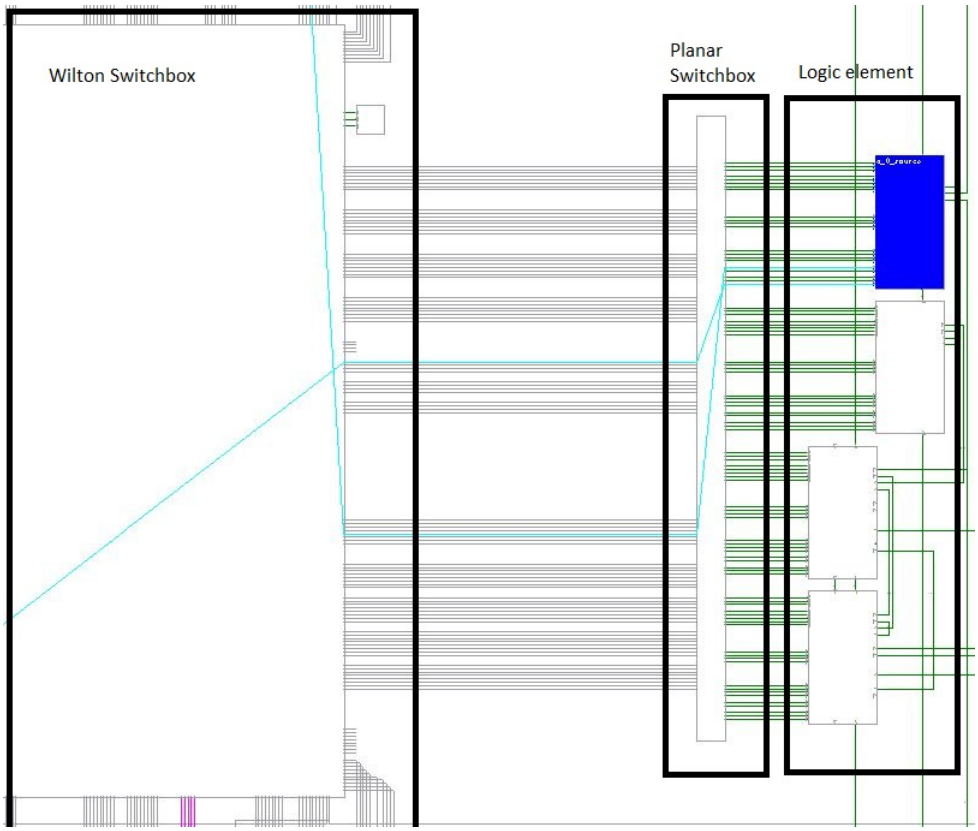


Figure 6.4.1: An example of a routed net. All paths start and end at planar switchboxes.

6.4.2 Timing

The initial error detection is in real-time because there is no logic between the raw signal and the probe DFF. The results of the probes take one clock cycle to propagate into the storage DFF. The only delay in the error detection would be from the electrical resistivity of the transistors and wire material used in the FPGA.

6.4.3 False positive readings

SEUs are caused by irradiated particles bombarding the FPGA. Adding probes to the design logic occupies more space on the FPGA. By occupying more space on the

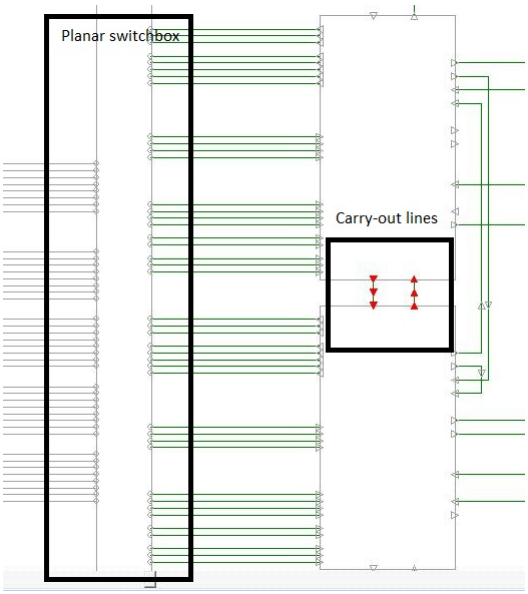


Figure 6.4.2: Carry-out port from 1 LE to the Carry-in port of an adjacent LE

FPGA, the probability of SEU occurrences increases. If a particle strikes the probes, an error reading will be produced. However, if a SEU happens on an unused region of the FPGA, an error reading will not be triggered. The probe method only detects SEUs that strike the design logic, or the probe logic.

6.4.4 SEUs striking the testing probes

If a SEU strikes the circuit, an error signal will be produced. However, the error message does not indicate where the error is happening. The SEU may have hit the testing probes instead of the design logic. The probe method makes no distinction as to whether the SEU hit a location before or after the testing probes.

6.4.5 Clock, reset, and enable signals

The clock, reset, and enable signals for flip-flops are routed in a different manner than the data signals. The FPGA has dedicated, high-speed lines to carry the clock signals. These signals eventually enter the switchbox through gates GCLK0 to GCLK7. Every DFF needs a clock line to properly synchronize their operation. The CIT currently ignores these lines because if anything were to happen to these signals, their effects would be visible to the entire FPGA. By disconnecting a clock, enable, or reset line, the FPGA ceases to function properly.

6.5 Proof of concept test results

6.5.1 Proving the probe method

This method was tested by creating a Verilog circuit and compiling the design for the Virtex-4 FPGA. The netlists were created from the ISE tool and the XDL file generated from the post Place-and-Route netlist. The post Place-and-Route netlist was edited in the FPGA editor tool to simulate some of the errors that may occur during SEU collisions.

Some of the errors were created successfully, such as: fan-out antennae, circuit breaks (seen in Figure 6.5.3), and circuit swaps (seen in Figure 6.5.4). Figure 6.5.2 shows the probe readings of an unaltered circuit. Figure 6.5.1 shows an example of the probes applied to a sample circuit. Other errors such as: circuit shorts, and unused circuit bridges were not successful. The Xilinx tool has a built-in checks to stop the user from intentionally damaging the FPGA.

Once the errors were created, a post-PnR simulation netlist was created for the

altered circuit. The effects of the errors can be viewed by applying a Verilog testbench script to the simulation netlist.

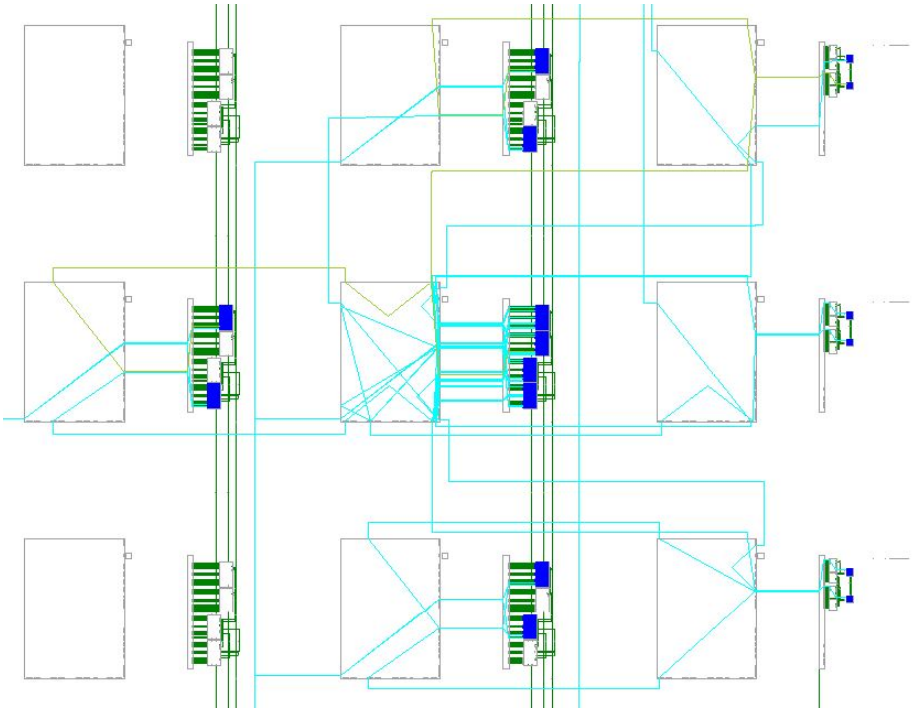


Figure 6.5.1: A view of the CIT probes at work.

6.5.2 Analysis on the CIT software

Probing failure

The CIT can fail if there is no free port available to route the probe point. The probe method relies on the ability to reroute and branch an existing routing net. The switchbox is made up of three types of ports: output, input, and mix. Output ports are ports that enter the switchbox and can fan-out the data to different paths. Input ports are ports that leave the switchbox and can accept connections from different sources. Mix ports are input and output ports combined into one. The switchbox used

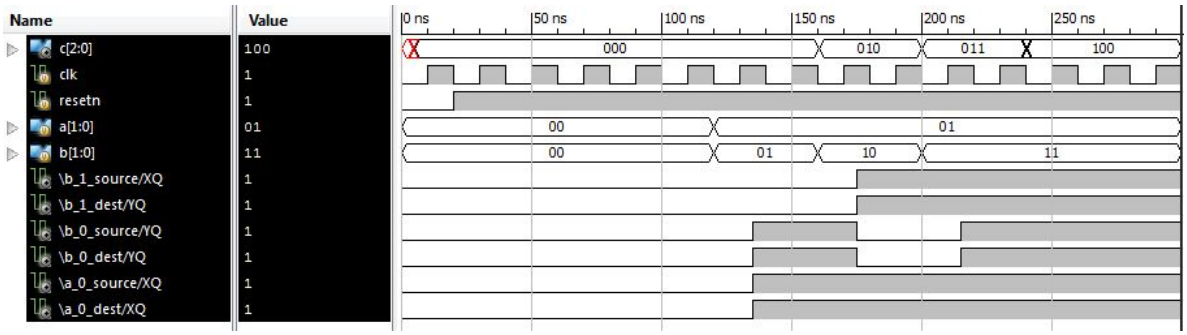


Figure 6.5.2: This is an example of the CIT probes monitoring a signal. In this image, the least significant bit of the C data line is being tapped. If the source probe matches the destination probe, there are no errors with the routing path.

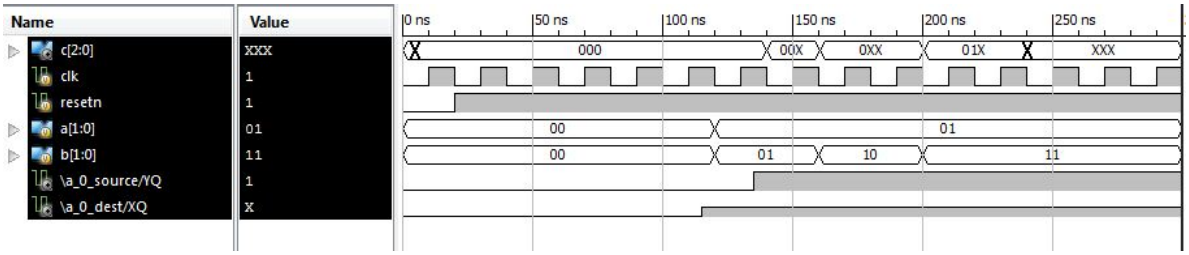


Figure 6.5.3: This is an example of a circuit break error. The least significant bit of "A" is broken between the source and the destination. The source probe picks up the intended value, but the destination probe shows an unknown value.

by the Virtex-4 family contains 408 ports. Of the 408 ports on the switchbox, 107 are inputs, 55 are mix, and 254 are outputs. The output ports are meant to connect to the input ports of the LEs. The destination probe points use these limited-fan-out ports to connect to the destination LE. The problem is partially mitigated if the coordinate system contained free LEs; the solution would be to branch the destination probe point to another LE in the same coordinate. The issue occurs when the switchbox is too densely packed. If all of the suitable fan-out ports are occupied, the probe point has no way to branch off to another LE. The current version of the CIT will return an error message if this occurs. A true solution would be to do custom placement and routing with emphasis on the critical datapaths taking precedence. This new

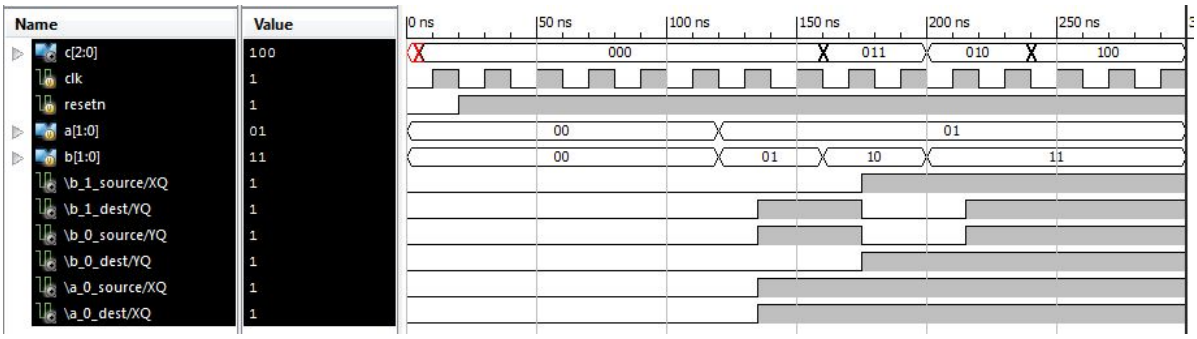


Figure 6.5.4: This is an example of a route swap. bit 0 and bit 1 of the "B" line have been swapped, but their respective probes are still intact. In this image, we see a disparity between the source and destination probes. The probe method is designed to detect single SEU errors. This swap requires four SEUs to occur.

placement and routing engine will ensure that there is enough space for the probes before the rest of the elements and routes are set into the FPGA.

6.6 Tool timing data

6.6.1 About the tables

Table 6.5 is the runtime data collected by running the CIT on siz test modules of a shutdown system mockup. These modules were created by McMaster University based off of the specifications for the Darlington Nuclear Generation Station [A.Wong et al., 2007]. Information regarding the size of the design is reported in this table. The following is a breakdown of the columns in the table:

1. The name of the module being tested

2. The LE occupancy percentage, represented as a percentage, and a fraction of all LEs on the FPGA.

Timing results for CIT on Triple Modular Redundant designs						
Circuit (TMR)	LE occupancy	IO occupancy	Occupied PIPs	Reason	Time elapsed	Verified?
High Log Rate trip	1% 31/5472	6% 20/320	586	FPGA islands	0.073 s	Cond.
Pushbutton Backlighting	1% 24/5472	4 % 15/320	509	Fatal error in CIT	0.019 s	No
Modified Heat Transport Low Flow	31% 1697/5472	59% 190/320	56531	Fatal error in CIT	8.142 s	No
LogN trip	10% 554/5472	3 % 26/320	14962	Fatal error in CIT	0.358 s	No
Heat transport High pressure and Heat Transport RRDP trips	6% 341/5472	23% 76/320	8028	Fatal error in CIT	0.125 s	No
Moderator level trips	1% 95/5472	11% 37/320	1642	Fatal error in CIT	0.015 s	No

Table 6.5: Timing results for the CIT with FPGA occupancy percentages

Timing results for CIT on Triple Modular Redundant designs				
Circuit (TMR)	Occupancy checks	Number of probes routed before error	Occupied PIPs	Time elapsed
High Log Rate trip	615	85/85 No fatal error	586	0.073 s
Pushbutton Backlighting	132	20/67	509	0.019 s
Modified Heat Transport Low Flow	353	67/1230	56531	8.142 s
LogN trip	31	1/226	14962	0.358 s
Heat transport High pressure and Heat Transport RRDP trips	28	1/621	8028	0.125 s
Moderator level trips	21	3/290	1642	0.015 s

Table 6.6: Timing and results table for the CIT with attempted probes

3. The input/output port occupancy percentage of the FPGA, represented as a percentage, and a fraction of all L/O ports on the FPGA.

4. The total number of PIPs used in the circuit

5. The verification condition of the test. This column indicates whether or not the module is probed successfully.

6. If the verification was not successful, the reason was provided in this column

7. The time elapsed for the tool to return a status message. If the tool was not successful and returned a fatal error, this time represents how much time the tool took to figure out the error. If the tool was successful in routing, the time elapsed represents the time used to probe the circuit.

Table 6.6 is the runtime data collected for the CIT using the same test cases from table 6.5. Information about the tool's actions are reported in this table. The following is a breakdown of the columns in the table:

1. The name of the module being tested.

2. The total number of occupancy checks performed during the run.

3. The total number of successfully probed points represented as a fraction of the total number of probes in the design.

4. The total number of PIPs used by the circuit.

5. The time elapsed for the tool to return a status message. If the tool was not successful and returned a fatal error, this time represents how much time the

tool took to figure out the error. If the tool was successful in routing, the time elapsed represents the time used to probe the circuit.

6.7 Results analysis

6.7.1 Fatal errors

The CIT was used on six submodules of a shutdown system trip computer mockup. Table 6.5 refers to the recorded tool runtime of the various test circuits. Each module was created with TMR, which included a triplication of the mission logic, and a majority rule module for each bit of each output signal of the mission logic. In the cases where the CIT returned a fatal error, the probe being routed by the CIT at the time had no free path to fork out. This is an expected result when the occupancy percentage of the FPGA rises above 30%, but every test circuit with the exception of the Modified Heat Transport Low Flow module are all under 30% LE occupancy. The problem is with the initial placement algorithm built into the Xilinx ISE tool. The circuit is packed too tightly for the probe points to branch off into other directions. In the remaining case where the verification was conditional, the CIT finished routing, but did not pass the verification stage.

6.7.2 Tool verification

The Xilinx XDL tool can turn a binary netlist (NCD) into an ASCII netlist (XDL). The XDL tool can also change an XDL back into an NCD file. The XDL file can be changed, and so when the XDL file is converted to binary, the XDL is checked for consistency. If the file contains connections that are erroneous, the conversion will

not occur. Errors such as: short-circuit connections, connections that are physically impossible, disjointed paths, and incomplete LE instantiations will result in an NCD file generation failure. If such a failure occurs, the XDL tool will not produce the NCD file. Disjointed paths are a problem for the CIT. The Virtex-4 FPGA LEs are organized into clock domain zones. Every zone can have their own unique clock signal. If a proposed probing path were to use multiple FPGA zones, there are special programmable interconnect points that need to be enabled in addition to the ones found in the switchbox. These PIPs have not been identified in the current version of the CIT. Therefore, if the XDL tool reports back errors due to FPGA islands, this is considered as a conditional success. In the single conditional pass case, the Xilinx ISE reported FPGA islands.

6.7.3 Results timing

The time elapsed columns in Table 6.5 and Table 6.6 show the amount of time the tool used to process the XDL. In the case of fatal errors, the time elapsed represents the time to return the first instance of a failed probe point. In the successful case, the time elapsed is the time needed to probe the test circuit.

The CIT algorithm relies on traversing through python lists. To make a successful connection, the algorithm suggests a PIP and compares it to the list of occupied PIPs. The suggested PIP at the desired X and Y location is checked against the list of all occupied PIPs. If the suggested PIP matches with an entry inside the list, this means that the suggested PIP is already in use. This process is called an occupancy check. If a suggested PIP fails the occupancy check, a new suggestion is made and the check is run again. If all combinations have been tried, then the tool cannot route the probe.

This occupancy check is run every time a new PIP is needed.

The runtimes of the CIT is based on the size of the occupied PIPs list. The results can be seen in Table ‘6.6. Each occupancy check looks through the list of occupied PIPs. If the list is large, then more time will be needed to traverse the list to check each value. The Modified Heat Transport Low Flow module takes the most amount of time to run because the occupied PIPs list has 56531 entries. The next largest occupied PIPs list belongs to the LogN trip module with 14958 entries. The Heat transport High pressure and Heat transport RRDP trip module is the third slowest module with 8028 occupied PIPs.

The frequency of the checks determines the runtime if the difference in the size of the occupied PIPs lists is small by comparison to the other test modules. The LogN trip module's occupied PIPs list size is approximately 25 times larger when compared to the occupied PIPs list of the High LogN Rate trip module. The difference in occupied PIPs list size is very large. Even though the High LogN Rate trip module performs 615 occupancy checks, 615 checks through High LogN Rate's 586 entries is faster than 21 checks through LogN Trip's 14958 entries. LogN trip's time of 0.358 seconds is much slower compared to High LogN Rate trip's time of 0.072 seconds.

When the High LogN Rate trip's results are compared to the Pushbutton Backlighting module, the frequency of occupancy checks becomes a factor in performance. The Pushbutton Backlighting module contains 132 occupancy checks, and an occupied PIPs list of 509 entries. The High LogN Rate module performs approximately 4 times more occupancy checks than the Pushbutton Backlighting module, and difference in the PIPs list size is 77 entries. The High LogN Rate module is slower than the Pushbutton Backlighting module because it performs more occupancy checks.

The frequency of the occupied checks also explains why the Pushbutton Backlighting module has the second fastest runtime, even though this module has the smallest occupied PIPs list. This module runs 132 occupancy checks, and is slower to process when compared to the Moderator Level Trips module with 21 occupancy checks and an occupied PIPs list size of 1642 entries. The Moderator Level Trips module contains approximately 3 times more occupied PIPs than the Pushbutton Backlighting module, which is a small difference in list size.

6.8 Chapter Summary

In summary, for every net in the design logic, two DFFs are needed: one to hold the value of the net at the source, and another DFF to hold the value at the destination. The probe method would be most effective when used in conjunction with the Triple Modular Redundancy and Duplication with Comparison error detection methods. The probes can detect single SEUs on every datapath segment that passes through the Wilton switchbox network within one clock cycle. The probes will only detect single SEUs that occur on the design logic, or the probe logic. If the LE cluster is too densely packed, a circuit with a 1 % occupancy percentage cannot be probed. This metric is also based on which data line is important. Assuming that the initial placement engine will always place all circuit elements in close proximity, the circuit where the data lines are placed at the outskirts of the mass will have a greater chance of a successful probing against one where the data lines are placed in the middle of the wiring mass. The runtimes of the tool is based on the number of occupied PIPs that the tool has to check in order to form a successful probe. The testing circuits had a range from 509 occupied PIPs to 56531 occupied probes. If the size of the

occupied PIPs list was similar amongst the modules, then the limiting factor to the runtime is the number of occupancy checks that the tool must run in order to form a successful probe. The testing circuits has a range of 21 checks to 615 checks.

Chapter 7

Future Work and Conclusion

In this thesis, a new detection method was developed for SEUs that affect the routing pathways of the FPGA. By attaching probe lines to the source and destination locations of each path, the changes along the path can be monitored. The goals of the solution were to detect all of the errors while using the least amount of resources, as soon as it occurs, and without having to re-invent the FPGA. The probe solution senses the SEU in real-time, and uses the existing hardware built into the FPGA. The previous work required a redesign of the FPGA, made copies of the design logic, or did not detect the error in real-time.

7.1 Planned work

7.1.1 CIT

Robustness

As it stands, the tool will fail if a probe point has no free fanout port paths that leave the switchbox. This is a problem when the switchbox is too densely packed to accommodate connections. The Xilinx placement and routing takes this into account and uses mix ports to ricochet the signal in order to connect to the target. Manual routing can be done to fix a single problem in routing, but this fix is not scalable. In order for the tool to be completely robust, it will replace the placement and routing engine of the Xilinx software. However, this solution is outside of the scope of this thesis. Later versions of the CIT will remove the initial placement and routing performed by the Xilinx software and proceed to set and connect the logic whilst placing effort on connecting the probe ports to free LEs. Another method to solve this problem may be to apply a manual constraint to the design in order to allow room for the probes. This method has not yet been tested.

FPGA islands

The Virtex-4 FPGA is sectioned into zones. Each zone can have its own clock rate. When an FPGA circuit occupies more than one zone, there are specific PIPs in the pathway that must be enabled for the connection to be complete. This is the reason why the conditional pass case of the shutdown system examples could not properly be probed on the Virtex-4 FPGA. The current version of the CIT is not aware of these specific PIPs and does not enable them in the process. The future versions of

the CIT will address this issue.

7.1.2 CIT comparator-fit

The final form of the comparator-fit module will be a place-and-route algorithm. The ideal form of the comparator-fit module would exhaustively choose a suitable location for the LUT comparator, using the remaining free space on the FPGA. This also falls outside of the scope of the thesis and will be completed at a later date.

7.1.3 Different FPGA families

The fanout and wiremap libraries only apply to the Virtex-4 family of Xilinx FPGAs. In order for this tool to work with other device groups, new fanout and wiremap libraries will have to be made. There is currently no automated process to make these libraries.

7.2 Conclusion

The tool for placing probes into the design logic still requires a substantial amount of work before it is ready for commercial use. In its ideal form, the CIT is an addition to the built-in placement and routing engine of the Xilinx development suite. The majority of the information stored in the libraries of the tool are already known by the placement and routing engine, and would not need to be manually entered.

Bibliography

Altera Corporation. Multitrack interconnect in Stratix III devices, May 2009. URL http://www.altera.com/literature/hb/stx3/stx3_siii51003.pdf.

Altera Corporation. SEU Mitigation for Cyclone V devices, 2013. URL http://www.altera.com/literature/hb/cyclone-v/cv_52008.pdf.

A. Avizienis and J.P.J. Kelly. Fault tolerance by design diversity: Concepts and experiments. *Computer*, 17(8):67–80, 1984. ISSN 0018-9162. doi: 10.1109/MC.1984.1659219.

A.Wong, D.K. Lau, M. Viola, and R. Black. SDS1 Trip Computer Design Description. Technical Report NK38-DID-68200-002 Rev. 008, Ontario Power Generation, May 2007.

M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, Maurizio Peri, and Saverio Pezzini. Fault-tolerant platforms for automotive safety-critical applications. In *Proceedings of the 2003 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, CASES '03, pages 170–177, New York, NY, USA, 2003. ACM. ISBN 1-58113-676-5. doi: 10.1145/951710.951734. URL http://doi.acm.org.libaccess.lib.mcmaster.ca/10.1145/951710.951734.

J. P. Bergstra. A formal approach to concurrent error detection in FPGA LUTs. Master's thesis, McMaster University, Department of Computing and Software, Hamilton, ON, Canada, July 2012. URL http://digitalcommons.mcmaster.ca/cgi/viewcontent.cgi?article=8485\&context=opendissertations.

S. Goldstein. Xilinx 4000 series CLB, 1998. URL http://www.cs.cmu.edu/afs/cs/academic/class/15828-s98/lectures/0119/img004.JPG.

H. Guzman-Miranda, L. Sterpone, M. Violante, M.A. Aguirre, and M. Gutierrez-Rizo. Coping with the obsolescence of safety- or mission-critical embedded systems using FPGAs. *Industrial Electronics, IEEE Transactions on*, 58(3):814 –821, March 2011. ISSN 0278-0046. doi: 10.1109/TIE.2010.2050291.

International Atomic Energy Agency IAEA. Application of Field Programmable Gate Arrays (FPGAs) in Instrumentation and Control Systems of NPPs. February 2013. URL http://www.iaea.org/NuclearPower/News/2013/2013-02-15-npe.html.

J. Johnson, W. Howes, M. Wirthlin, D.L. McMurtrey, M. Caffrey, P. Graham, and K. Morgan. Using duplication with compare for on-line error detection in FPGA-based designs. In *Aerospace Conference, 2008 IEEE*, pages 1–11, 2008. doi: 10.1109/AERO.2008.4526470.

S. Kayali. Space radiation effects on microelectronics, 2002. URL http://parts.jpl.nasa.gov/docs/Radcrs_Final.pdf. This is an electronic document. Date of publication: March 19, 2002. Date retrieved: September 23, 2012. Date last modified: [Date unavailable].

J. C. Knight. Safety critical systems: Challenges and directions. In *Proceedings of the*

24th International Conference on Software Engineering, ICSE '02, pages 547–550, New York, NY, USA, 2002. ACM. ISBN 1-58113-472-X. doi: 10.1145/581339.581406. URL `http://doi.acm.org/10.1145/581339.581406`.

C. Lin. Introduction to flip flops: D and T. Lecture, 2003. URL `http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Seq/flip.html`.

B. Pratt, M. Caffrey, J.F. Carroll, P. Graham, K. Morgan, and M. Wirthlin. Fine-grain SEU mitigation for FPGAs using Partial TMR. In *Radiation and Its Effects on Components and Systems, 2007. RADECS 2007. 9th European Conference on*, pages 1–8, 2007. doi: 10.1109/RADECS.2007.5205468.

E.S.S. Reddy, V. Chandrasekhar, M. Sashikanth, V. Kamakoti, and N. Vijaykrishnan. Detecting SEU-caused routing errors in SRAM-based FPGAs. In *VLSI Design, 2005. 18th International Conference on*, pages 736 – 741, Jan. 2005. doi: 10.1109/ICVD.2005.79.

L. Rockett, D. Patel, S. Danziger, B. Cronquist, and J.J. Wang. Radiation Hardened FPGA Technology for Space Applications. In *Aerospace Conference, 2007 IEEE*, pages 1–7, 2007. doi: 10.1109/AERO.2007.353098.

A. Rohani, H. R. Zarandi, and M. Zandrahimi. New Switch Box Architecture for SEU Detection in SRAM-Based FPGAs. In *Computer Science and its Applications, 2009. CSA '09. 2nd International Conference on*, December 2009. doi: 10.1109/CSA.2009.5404269.

C. Stroud, S. Wijesuriya, C. Hamilton, and M. Abramovici. Built-in self-test of

FPGA interconnect. In *Test Conference, 1998. Proceedings., International*, pages 404 –411, October 1998. doi: 10.1109/TEST.1998.743180.

X. Sun, S. Xu, Jian X., and P. Trouborst. Design and implementation of a parity-based BIST scheme for FPGA global interconnects. In *Electrical and Computer Engineering, 2001. Canadian Conference on*, volume 2, pages 1251–1257 vol.2, 2001. doi: 10.1109/CCECE.2001.933621.

S. J. E. Wilton. *Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memory*. PhD thesis, University of Toronto, 1997.

Y.C. Yeh. Safety critical avionics for the 777 primary flight controls system. In *Digital Avionics Systems, 2001. DASC. 20th Conference*, volume 1, pages 1C2/1–1C2/11 vol.1, 2001. doi: 10.1109/DASC.2001.963311.