2017-04-01

Using On-Chip Error Detection to Estimate FPGA Design Sensitivity to Configuration Upsets

Andrew Mark Keller
Brigham Young University

Using On-Chip Error Detection to Estimate FPGA Design Sensitivity to Configuration Upsets

Andrew Mark Keller

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Michael J. Wirthlin, Chair
James K. Archibald
Brent E. Nelson

Department of Electrical and Computer Engineering

Brigham Young University

ABSTRACT

Using On-Chip Error Detection to Estimate FPGA
Design Sensitivity to Configuration Upsets

Andrew Mark Keller
Department of Electrical and Computer Engineering, BYU
Master of Science

SRAM-based FPGAs provide valuable computation resources and reconfigurability; however, ionizing radiation can cause designs operating on these devices to fail. The sensitivity of an FPGA design to configuration upsets, or its SEU sensitivity, is an indication of a design's failure rate. SEU mitigation techniques can reduce the SEU sensitivity of FPGA designs in harsh radiation environments. The reliability benefits of these techniques must be determined before they can be used in mission-critical applications and can be determined by comparing the SEU sensitivity of an FPGA design with and without these techniques applied to it. Many approaches can be taken to evaluate the SEU sensitivity of an FPGA design. This work describes a low-cost easier-to-implement approach for evaluating the SEU sensitivity of an FPGA design. This approach uses additional logic resources on the same FPGA as the design under test to determine when the design has failed, or deviated from its specified behavior. Three SEU mitigation techniques were evaluated using this approach: triple modular redundancy (TMR), configuration scrubbing, and user-memory scrubbing. Significant reduction in SEU sensitivity is demonstrated through fault injection and radiation testing. Two LEON3 processors operating in lockstep are compared against each other using on-chip error detection logic on the same FPGA. The design SEU sensitivity is reduced by 27× when TMR and configuration scrubbing are applied, and by approximately 50× when TMR, configuration scrubbing, and user-memory scrubbing are applied together. Using this approach, an SEU sensitivity comparison is made of designs implemented on both an Altera Stratix V FPGA and a Xilinx Kintex 7 FPGA. Several instances of a finite state machine are compared against each other and a set of golden output vectors, all on the same FPGA. Instances of an AES cryptography core are chained together and the output of two chains are compared using on-chip error detection. Fault injection and neutron radiation testing reveal several similarities between the two FPGA architectures. SEU mitigation techniques reduce the SEU sensitivity of the two designs between 4× and 728×. Protecting on-chip functional error detection logic with TMR and duplication with compare (DWC) is compared. Fault injection results suggest that it is more favorable to protect on-chip functional error detection logic with DWC than it is to protect it with TMR for error detection.

Keywords: FPGA, SEU mitigation, SEU sensitivity, reliability, error detection, triple modular redundancy, TMR, duplication with compare, DWC, scrubbing, fault-tolerance, neutron radiation testing, fault-injection

ACKNOWLEDGMENTS

I would like to thank all those who have made this thesis possible. I could not have accomplished this work without the support and encouragement of all of the invested professors, organizations, colleagues, family, and friends. Their contributions have been invaluable.

My advisor, Dr. Michael Wirthlin, has been instrumental in guiding me through my research and opening the windows of opportunity to advance this work. I am honored that he would entrust this work to me. His encouragement and enthusiasm inspire me.

I am grateful to the other members of my graduate committee, Dr. Brent Nelson and Dr. James Archibald, for their support and for the knowledge and experience they have helped me gain. I am also thankful for the advice and tutelage of Dr. Brad Hutchings, Dr. Doran Wilde, Dr. D. J. Lee, and the other faculty and staff members who have helped me along the way.

This work could not have been done without collaborating with colleagues outside of the BYU Configurable Computing Lab (CCL). I would like to thank Rick Wong and Shi-Jie Wen from Cisco, David S. Lee from Sandia National Laboratories, Heather Quinn from Los Alamos National Laboratories, Mike Hutton from the Intel Programmable Solutions Group, and Paul J. Vincent and Matthew Noell from Raytheon for all of their support and contributions.

My family has been an unwavering support throughout my journey. I would like to thank my wife, Beth, for her love and patience with "robot soccer close," and my daughter, Hailey Anne, for the joy she is to me. My parents and siblings have also been great supports.

The other students in the BYU CCL have also helped me progress in my education and research. Specifically I would also like to thank Aaron Stoddard, Ammon Gruwell, Parker Ridd, Chase McCloskey, James Swift, Hayden Rowberry, Tim Whiting, and Ken Sawyer.

This work supported by the Los Alamos Neutron Science Center (LANCSE) which provided neutron beam time under proposal NS-2016-7268-F. This work was also supported by the I/UCRC Program of the National Science Foundation under Grant No. 1265957 through the NSF Center for High-Performance Reconfigurable Computing (CHREC) and by Cisco Systems, Inc.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

SRAM-based field programmable gate arrays (FPGA) are being used increasingly in terrestrial [1] and space-based applications [2]. Modern FPGAs provide large amounts of programmable logic, computation resources, I/O, memory and special-purpose functionality. As FPGAs are released on new technology, they are able to provide this functionality at lower power and operating frequencies. FPGAs are also increasingly integrating dedicated, hardened IP to provide a mix of fixed function and programmable function solutions. Their reconfigurability and low overhead for design development make them an attractive alternative to application-specific integrated circuits (ASIC).

Because SRAM-based FPGAs use a large set of static configuration memory cells (CRAM) to support programmability, designs operating on SRAM-based FPGAs are susceptible to failure induced by ionizing radiation. The value stored in an SRAM memory cell can be inverted by the passing of a single energetic atomic particle through the device. This is known as a single event upset (SEU) [3–5]. SEUs are troublesome for SRAM-based FPGA designs because an upset in configuration memory could alter the behavior of logic, routing, or other utilized resources. These alterations could cause the design to fail, or deviate from its defined behavior. Ionizing radiation is a great concern for FPGA applications in harsh radiation environments like space and high-energy physics experiments, and it is a growing concern for FPGA applications in terrestrial environments [6, 7].

The susceptibility of an FPGA design to failure caused by SEUs in configuration memory is discussed throughout this thesis in terms of SEU sensitivity. SEU sensitivity is defined as how prone to failure a given design is when a random SEU occurs. It is not guaranteed that an SEU will cause a design operating on an SRAM-based to fail [8]. An SEU may occur in an unused portion of the FPGA or be masked in such a way that it does not affect the correct behavior of the design. Or it may alter the design and lead to failure [23, 24]. The likelihood of a random SEU causing a

design to fail determines SEU sensitivity. The more likely a random SEU is to cause failure, the more *sensitive* the design.

Much research has been done to develop SEU mitigation techniques for SRAM-based FPGAs [9–12], which reduce SEU sensitivity. These techniques include structural redundancy such as: triple modular redundancy (TMR), duplication with compare (DWC), and state machine encoding. While expensive from a resource perspective, these techniques can greatly improve the reliability of a design [13]. Configuration scrubbing is also employed to rapidly repair upsets and prevent the accumulation of upsets in configuration memory [14].

An improvement in reliability or a reduction in SEU sensitivity of an FPGA design is referred to throughout this thesis as a ratio between the SEU sensitivity of the design without SEU mitigation and with SEU mitigation techniques applied. An improvement in reliability and a reduction in sensitivity are synonymous. For example, a $4\times$ improvement in reliability or a $4\times$ reduction in SEU sensitivity means that there is a 4:1 SEU sensitivity ratio between the unmitigated and mitigated version of an FPGA design respectively.

The SEU sensitivity of an FPGA design plays an important part in advancing the state of the art in FPGA reliability. Knowing the SEU sensitivity of a particular design can help identify what causes FPGA designs to be sensitive to SEUs. The impact of changes in the fabrication process, architecture, and applied SEU mitigation techniques can all be better understood through comparing the SEU sensitivity of various designs. SEU mitigation techniques can be validated and additional vulnerabilities discovered through the SEU sensitivity of an FPGA design.

The SEU sensitivity of an FPGA design is typically evaluated through fault injection and radiation testing. Fault injection is the process of artificially introducing a fault and observing the response. The type of fault injection referred to in this work involves mimicking the behavior of an SEU in an FPGA [15]. Radiation testing is the process of exposing the FPGA design to an accelerated radiation source and observing the response. Several types of radiation used for SEU sensitivity testing include: proton, neutron, and heavy ions [16]. SEU sensitivity is measured in this work by the percentage of injected faults that caused a failure for fault injection and by the design cross section for radiation testing.

Many test setups can be used to evaluate the SEU sensitivity of an FPGA design through fault injection or radiation testing. SEU sensitivity testing is event based, where the event is the

occurrence of a functional failure. Most SEU sensitivity testing approaches depend on the ability to detect when a functional failure has occurred. Typically, input stimulus is provided to the design under test and its outputs are compared against a known *golden* value to detect failures. Most SEU sensitivity testing approaches separate error detection logic from the device being tested. These *off-chip* error detection approaches sometimes involve creating custom test boards that use multiple FPGAs.

An on-chip error detection approach provides a cost effective means for rapidly evaluating the SEU sensitivity of an FPGA design. Implementing *off-chip* error detection can be expensive and time consuming. This thesis presents an *on-chip* error detection evaluation approach that is low-cost and easier to implement. This approach uses additional logic resources on the same FPGA as the design under test to determine when the design has failed, or deviated from its specified behavior. Although the error detection circuitry is also vulnerable to failure, valuable information about the SEU sensitivity of a design can still be obtained using this approach. To improve accuracy, SEU mitigation techniques are always applied to the on-chip error detection circuitry. This thesis describes the benefits and limitations of this approach as compared to those of other SEU sensitivity evaluation approaches.

Using an on-chip error detection approach, the reliability benefits of three SEU mitigation techniques are evaluated: TMR, configuration scrubbing, and user-memory scrubbing. Two LEON3 processors operating in lockstep are compared against each other using on-chip error detection logic on the same FPGA. The design SEU sensitivity is reduced by 27$\times$ when TMR and configuration scrubbing are applied; and by approximately 50$\times$ reduction when TMR, configuration scrubbing, and user-memory scrubbing are applied together. Using this approach, an SEU sensitivity comparison is made of designs implemented on both an Altera Stratix V FPGA and a Xilinx Kintex 7 FPGA. Several instances of a finite state machine are compared against each other and a set of golden output vectors, all on the same FPGA. Instances of an AES cryptography core are chained together and the output of two chains are compared using on-chip error detection. Fault injection and neutron radiation testing reveal several similarities between the two FPGA architectures. SEU mitigation techniques reduce the SEU sensitivity of the two designs between 4$\times$ and 728$\times$. Protecting on-chip functional error detection logic with TMR and DWC is compared.

Fault injection results suggest that it is more favorable to protect on-chip functional error detection logic with DWC than it is to protect it with TMR for error detection.

1.1 Thesis Contributions

This thesis presents an approach for evaluating the SEU sensitivity of a design using on-chip error detection. The advantages and disadvantages of many evaluation approaches are discussed, including those of on-chip error detection and other evaluation approaches used in related works. A generalized composition of an on-chip error detection test fixture is presented and important aspects of on-chip error detection are covered. The importance of evaluating SEU sensitivity and a comparison of various evaluation approaches is discussed in Chapter 3. On-chip error detection concepts and test fixture composition is covered in Chapter 4.

The work presented in this thesis uses on-chip error detection to evaluate the SEU sensitivity of several FPGA designs through fault injection and radiation testing. Unmitigated and mitigated FPGA designs were evaluated on a Xilinx Kintex 7 FPGA and an Altera Stratix V FPGA. Three SEU mitigation techniques were included in this study: TMR, configuration scrubbing, and user memory scrubbing. Various integrations of theses techniques were evaluated, by only applying TMR to mitigate a design, or applying TMR with user-memory scrubbing and no configuration scrubbing, etc. A LEON3 soft processor, a finite state machine, and a 128-bit AES cryptography core were used as benchmark designs. A variety of on-chip error detection schemes and mechanisms were used to evaluate SEU sensitivity.

From the results, three main comparisons are made. First, the reliability benefits of various mitigation schemes are compared, demonstrating the complementary nature of SEU mitigation techniques. Second, the SEU sensitivity of FPGA designs across two different architectures are compared, that of an Altera Stratix V FPGA and that of a Xilinx Kintex 7 FPGA. Finally, SEU sensitivity results obtained from fault injection are used to compare TMR protected and DWC protected on-chip error detection logic.

CHAPTER 2. FPGA CONFIGURATION UPSETS AND MITIGATION TECHNIQUES

SRAM-based FPGAs, like application specific integrated circuits (ASICs), are susceptible to radiation induced failures; fortunately, many mitigation techniques have been developed to improve the reliability of these devices in spite of their vulnerabilities [2]. Most radiation induced failures in these devices result from non-permanent, soft errors that can be prevented, masked, or recovered from. Radiation effects observed in an FPGA may or may not cause the design operating on the FPGA to cease functioning correctly. In developing SEU mitigation techniques and preparing designs for use on commercial FPGAs in harsh radiation environments, it is important to understand where radiation comes from, how it effects FPGA designs, and what can done to improve the robustness of a design.

This chapter motivates the need for evaluating the effects of configuration upsets on FPGA designs. Configuration upsets are caused by ionizing radiation, which is present in space and terrestrial environments. Ionizing radiation can come from galactic cosmic rays or impurities in packing materials. This radiation can cause FPGA designs to fail, or deviate from their specified behavior. These failures are difficult to reproduce and are becoming a greater issue as features sizes shrink and more FPGAs are being deployed in data centers and Internet core routers [1, 17]. SEU mitigation techniques can be applied to FPGA designs to make them less sensitive to configuration upsets. This chapter discusses radiation sources, FPGA architecture, and SEU mitigation techniques. It provides the background information necessary to fully appreciate the purpose and challenges behind SEU sensitivity evaluation, and motivates the need for using the proposed on-chip error detection approach to evaluate the SEU sensitivity of an FPGA design.

In this thesis, on-chip error detection is used to evaluate the SEU sensitivity of various FPGA designs. Fault injection and radiation beam testing are used to simulate and cause SEUs in FPGA configuration memory. Specific bits that cause the design to fail are observed and the overall sensitivity of the design to configuration upsets is determined. Unmitigated and SEU mitigated

FPGA designs are evaluated on an Altera Stratix V and a Xilinx Kintex 7 FPGA. The SEU mitigation techniques evaluated are: TMR, configuration scrubbing, and user-memory scrubbing. The collected SEU sensitivity data assists in understanding the effects of radiation on FPGA designs and the benefits of SEU mitigation techniques.

2.1 Radiation Sources

FPGAs are exposed to radiation present in their operating environments. FPGAs are deployed in a number of applications including terrestrial and space based applications. In terrestrial environments, there are three main sources of fault-inducing radiation: alpha particles, high-energy cosmic rays and thermal neutrons [18]. These radiation sources pose a threat to FPGA reliability in terrestrial [6, 7] and space environments. Other sources of radiation in space environments include: trapped protons, galactic cosmic rays (GCRs), and heavy ions [19]. The primary concern is making sure that FPGA designs are able to withstand the type and amount of radiation present in their operating environments.

A significant source of ionizing radiation are alpha particles emitted from impurities in device materials. Alpha particles directly ionize the device as they pass through it. The dominant sources of alpha particles are uranium and thorium found in the packing materials, which makes shielding difficult. Special considerations can be taken to lessen the influence of alpha particles emitted from packaging materials on FPGA devices [18].

Another significant source of fault-inducing radiation comes from cosmic rays. Cosmic rays are high energy atomic particles that mostly originate from beyond the solar system. Less than 1% of particles directly from cosmic rays ever make it to the earth's surface. Due to their density and stability, neutrons are the most likely candidate from cosmic rays to cause an upset in an FPGA here on earth. Since neutrons are not charged, they indirectly ionize a device by colliding with other atoms in the device, releasing secondaries [18]. The high energy neutrons flux (above 10MeV) in New York City is approximately 13 $n\ cm^{-2}h^{-1}$ and increases as a function of altitude [4].

The third significant source of fault-inducing radiation comes from the interaction of thermal neutrons with boron-10 found in borophosphosilicate glass (BPSG), which has been used to fabricate some FPGAs. Thermal neutrons are relatively slow and have energies of less than .4 eV.

After boron-10 absorbs a thermal neutron, it breaks apart into an alpha particle, a gamma ray, and a lithium nucleus. The alpha particle ionizes the device and may cause a configuration upset. To eliminate the threat posed by low-energy thermal neutrons, FPGAs can be manufactured without the use of BPSG [7].

Radiation poses a greater concern in harsh radiation environments like space and high-energy physics experiments [6]. Mission length, environmental radiation, and the number of deployed units are all factors to take into consideration when determining the reliability needs of a design [16]. The presence of radiation is a concern for FPGA designs because ionizing radiation can affect the functionality and state of a design.

2.2 Radiation Effects

Radiation effects on SRAM-based FPGAs can be caused by a single energetic particle or result from the accumulation of charge injected by radiation [19]. When a single energetic particle creates an observable effect within the FPGA, it is known as a single event effect (SEE) [3]. Long term effects of radiation exposure include total ionizing dose (TID). There are other radiation effects such as displacement damage and prompt dose. This overview of radiation effects gives attention to SEEs and focuses on SEUs, which alter FPGA configuration memory. This work does not address TID or other radiation effects beyond SEEs. An explanation of various SEEs is given and their influence on FPGAs is described.

Radiation effects result primarily from the funneling phenomenon. As shown in Figure 2.1, when an ionized particle passes through a sensitive node in a semiconductor, a path of electron-hole pairs is produced from the semiconductor material atoms, resulting in funnel-shaped equipotential surfaces that create a current in the device [19, 20]. This can also occur when secondary particles from a neutron collision pass through the sensitive region of the device. This phenomenon contributes to radiation effects and is the primary mechanism behind SEEs. Because of the complexities of this process, simplified fault models have been created to represent the behavior of these effects at a higher level of abstraction.

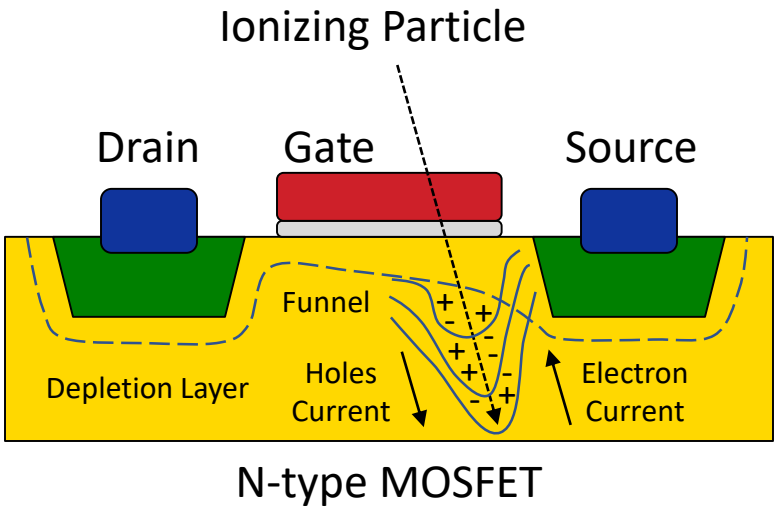


Figure 2.1: Funneling phenomenon in a N-type MOSFET. Adapted from a figure in [19].

2.2.1 Single Event Effects

SEEs are any observable effects caused by single energetic particle [3, 4]. There are two main types of SEEs: destructive and non-destructive. Destructive SEEs can permanently damage the FPGA, whereas non-destructive SEEs are transient and can be removed by reconfiguring or power cycling the device. Destructive SEEs include: single event latchup (SEL), single event burnout (SEB), and single event gate rupture (SEGR). Non-destructive SEEs include: single event transient (SET), single event upsets (SEUs), and single event functional interrupts (SEFI). Of these SEEs, SETs and SEUs are the primary concern for FPGA vulnerability because they occur more frequently [16, 19, 20].

SELs occur when a particle strike causes an internal short between power and ground within the FPGA. Only power cycling the FPGA can stop the SELs. If the SELs is not removed in time, the FPGA can be permanently damaged because of the high-currents induced by the latchup [19]. A thoughtful FPGA fabrication process and layout can mitigate this effect [21]. FPGAs can be tested for SEL immunity before being used in harsh radiation environments. Devices that are not immune to SELs are often avoided for use in harsh radiation environments like space [16].

SETs occur when a particle strikes the channel region of an inactive n-type MOSFET or the drain region of an inactive p-type MOSFET [19, 20]. The induced current pulse appears as a *glitch*

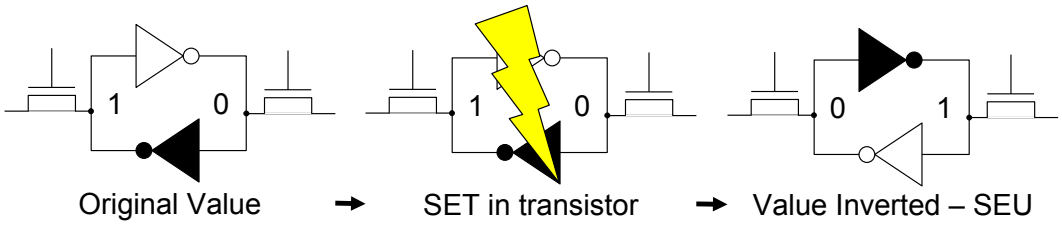


Figure 2.2: An SEU within an SRAM-cell

and is short lived. SETs are assigned a polarity based on the transition of the output of the CMOS device. Positive SET pulses transition from 0 to 1 and back to 0; negative SET pulses transition from 1 to 0 and back to 1. SETs can be latched by memory elements as they propagate through the device, which becomes a greater issue at faster clock speeds.

SEUs occur when ionizing radiation inverts the value stored in a memory element, such as a configuration bit of an FPGA. An illustrative example, depicted in Figure 2.2, shows how a single particle is able to upset the value stored in an SRAM cell. In this example an SET occurs within a transistor of the feedback loop used to maintain the value of the SRAM-cell. With enough amplitude, this SET disrupts the feedback loop of the cell and causes it to take on the inverted value. When this occurs in an SRAM-based FPGA it can have many adverse effects (see Section 2.3).

SEFIs occur when elements of the FPGA that control its functionality as a whole are affected by a single particle strike. SEFIs can cause the entire FPGA to malfunction [19]. SEFIs are temporary but could prevent reprogramming the FPGA until the next power cycle. There is some state within an FPGA that, if upset, could disrupt the global functionality of the FPGA. While SEFIs are a concern to FPGA reliability, more SRAM-based FPGA design failures are due to SEUs and SETs.

2.3 FPGA Architecture and SEU Failure Modes

Understanding FPGA architecture and design flow helps explain how SEUs, induced by ionizing radiation, can adversely affect an FPGA design. This section describes basic FPGA architecture and how SRAM-based FPGAs support programmability. An overview is given for an FPGA design flow from hardware description to configuration, and then common SEU failure

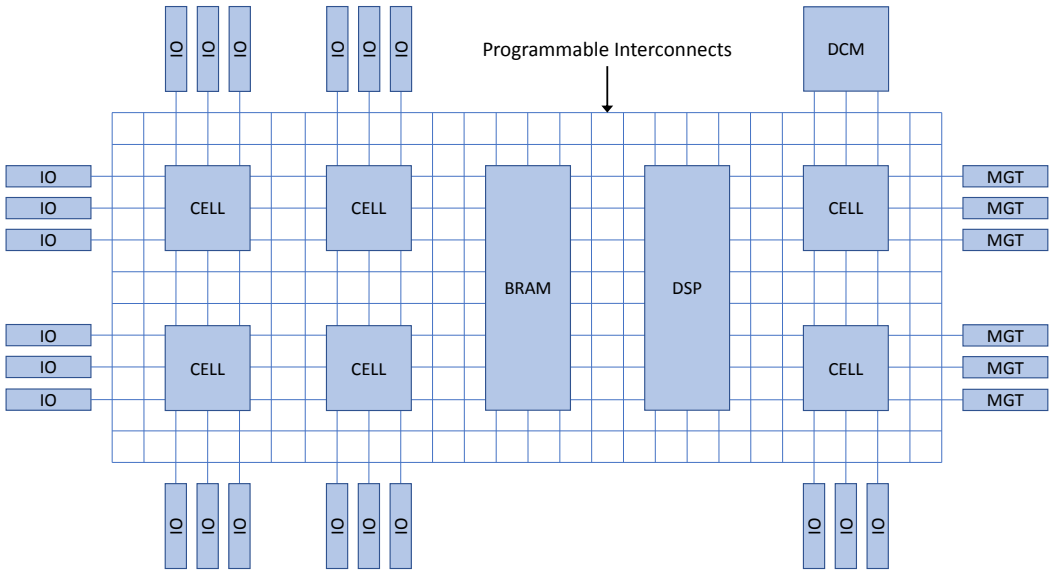


Figure 2.3: Island-style FPGA Architecture Layout. Adapted from a Figure in [16].

modes are discussed. This background information motivates the need for SEU sensitivity evaluation of FPGA designs by explaining how an SEU can cause a design to fail.

An FPGA consists of basic logic building blocks and specialized hard IP blocks organized into a specific architecture. Lookup-tables and flip-flops are grouped into slices or cells with other supporting hardware (e.g., carry-chains, multiplexers, control signals). Some lookup-tables are read only, while others can have their values changed by the design (e.g. LUTRAMs, shift-registers). Additional hard IP blocks can include: large user-memories (BRAMs), arithmetic units (DSPs), analog-to-digital converters (ADCs), high-speed serial I/O (MGTs), and clock-managers (DCM). The available resources and architecture are vendor specific, but many popular FPGA architectures are *island-style* meaning that groups of logic resources are surrounded by programmable routing interconnects in a 2D matrix like formation [22]. This layout is depicted in Figure 2.3.

There are two major types of bits within an FPGA's programming data: configuration (CRAM) bits and internal block memory (BRAM) bits. Data associated with programmable interconnects, lookup tables, control signals, and the contents of smaller user-memories (e.g., flip-flops, LUTRAMs, shift-registers) are stored in CRAM bits. Data associated with the contents of larger blocks of user-memory are stored in BRAM bits. Although the percentage is small, there are other

types of bits within the FPGA's programming data (e.g., global status registers, test control logic). The contents of user-memories may be altered by the design during operation. The configuration of specific designs are loaded into the FPGA's programming data.

Figure 2.4 shows the generalized FPGA design flow. Digital designs are described in a hardware description language (HDL) and synthesized into a netlist, which describe the cell instances and connections needed to implement a given user-design. Computer aided design tools (CAD) map the cell instances to technology specific primitives (e.g. a six input look-up-table). These primitives are then *placed*, or assigned to a specific location in the FPGA. All of the connections are then *routed* using programmable interconnects. This information is used to generate a bitstream, which is the contents of the CRAM bits and the initial value of the BRAM bits used to program the FPGA. When the bitstream is configured onto the FPGA, all of its information, (the lookup-table contents, interconnects, initial state, and other configuration), is loaded into the respective memory cells. SRAM-based FPGAs use SRAM-cells to store this information. Once the FPGA is loaded, the design begins to operate on the FPGA fabric. Some memory cells maintain their value throughout the operation of the design, others are user-memory resources and change their values according to the design.

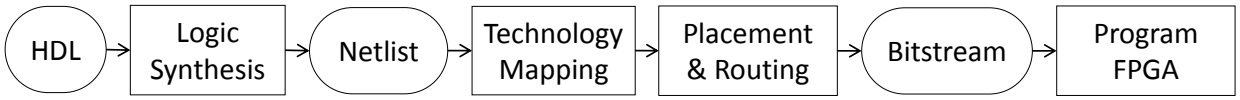


Figure 2.4: Basic FPGA Design Flow

Since FPGA programming data stores circuit configuration and state, SEUs in an FPGA's programming data can alter the design operating on the FPGA and lead to failure. The main cause of SEU failure modes in FPGAs are SEUs in routing resources, lookup tables, control signals, and user-memories. SEUs in lookup tables and control signals can corrupt logic and alter the functionality of primitive blocks used by the design. SEUs in user-memories can corrupt the state of a design (see Figure 2.5). SEUs in routing resources may cause nets to disconnect, short to power or ground, or bridge with other nets in the design (see Figure 2.6) [23, 24].

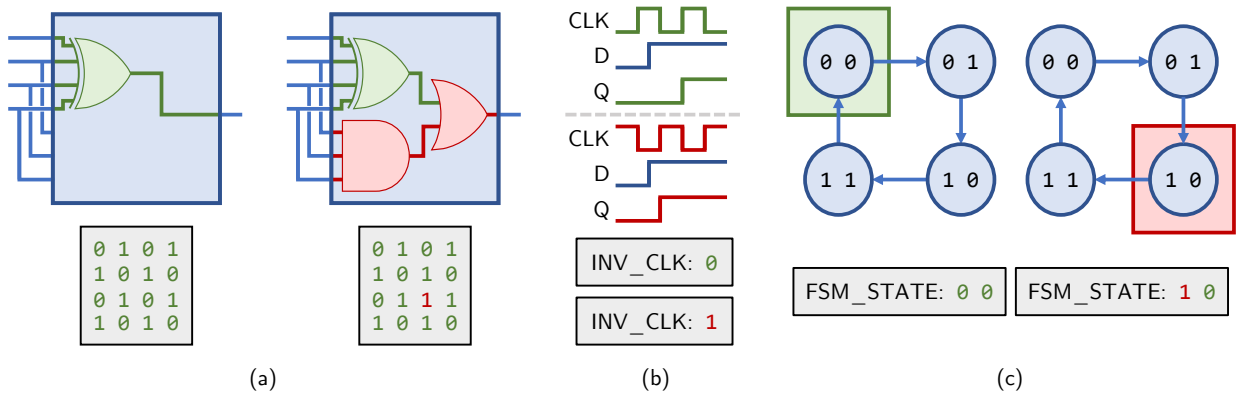


Figure 2.5: SEU failure modes. SEUs can alter correct logic (a), control (b), or state (c).

To support all of the resources and possible interconnect configurations many CRAM cells are needed to store the programming data. The specific role of each bit in the data is proprietary, but a large portion of configuration memory is dedicated to functionality beyond user flip-flops, BRAM content and LUT contents [25]. One study found through fault injection that approximately 80% of FPGA design failures originate from SEUs in routing structures [23].

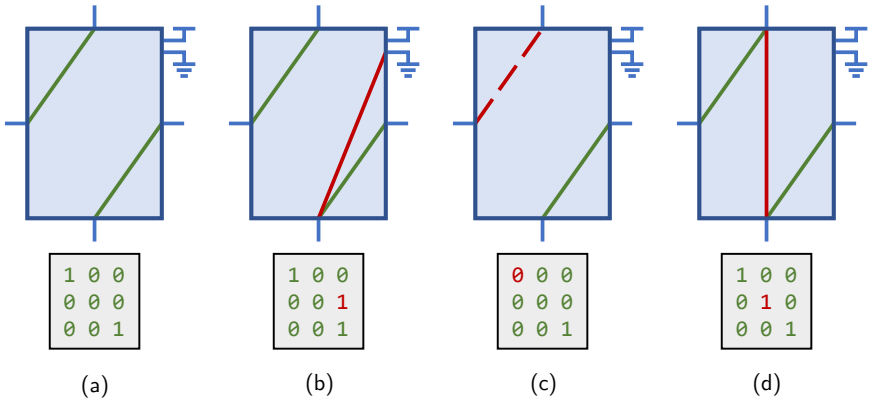


Figure 2.6: SEU routing failure modes. SEUs can alter correct routing (a) by shorting to ground (b), disconnecting a route (c), or bridging two routes together (d).

2.4 Reliability: Faults, Errors and Failures

A reliable FPGA design is one that can withstand environmental factors, such as radiation, without ceasing to provide its defined service. Unreliability stems from faults, errors and failures.

A fault is the lowest level problem that occurs within the FPGA, such as an SEU. An error is the manifestation of a fault, and a failure occurs when the provided service deviates from the intended service [26]. It is important to understand that in any FPGA design, a fault *may* lead to failure but the occurrence of a fault does not guarantee that an FPGA design will fail. A fault-tolerant FPGA design can withstand more faults without failing.

Techniques can be applied to an FPGA design that will reduce its sensitivity to SEUs. These *SEU mitigation* techniques typically mask or detect faults within the system through redundancy, checkpoints, or some type of error correction code. By correctly masking or handling fault within the design, the design becomes more fault-tolerant, meaning that it can tolerate more faults without failing.

Radiation can cause faults within SRAM-based FPGAs. Any SEU in configuration memory is a fault. Not all faults will affect an active design on an FPGA; a fault may or may not manifest as an error. If an SEU occurs in an unused portion of the FPGA or in such a way that its effect on the design is masked, it will not alter the operating design. For example, if an SEU causes the contents of a LUT to change at a lookup address that is unused or otherwise masked, it will have no effect on the operating design. This is significant because FPGA designs only utilize a portion of the FPGA. As covered in Section 2.3, a fault can change the behavior and state of an FPGA design in many different ways such as: changing the logic in a LUT, altering routing within the circuit, and modifying the behavior of components. If an SEU does alter a configuration bit associated with the proper operation of a design, it may cause an error in the design such as: providing incorrect output, halting its operation, or otherwise deviating from its specified behavior.

An error is a deviation from expected behavior. Some errors can be detected and recovered from or simply ignored. It is possible for an error to occur without being considered a failure. An example of this is when the checksum of a communication interface does not check out but after re-transmission, the correct data is received. Here the defined service is still provided even though an error has occurred. Unacceptable errors are considered failures.

A failure occurs when an error causes a service to deviate from its defined specification. A major concern when it comes to failure is that of silent data corruption (SDC). SDC occurs when output or state is incorrect, but the error goes undetected. This can be particularly troublesome for missile guidance systems, file-compression, and encryption algorithms. The on-chip error

detection evaluation approach used in this thesis focuses on detecting SDC of output signals for selected benchmark circuits.

2.5 SEU Mitigation Techniques

Adverse affects of SEUs on FPGA designs can be reduced by applying various SEU mitigation techniques to the design. Many post-manufacturing SEU mitigation techniques have been developed for SRAM-based FPGAs. Generally, concepts of redundancy, repair, error detection and correction (EDAC), or the elimination of single point failures (SPF) are used to design SEU mitigation techniques. The three SEU mitigation covered in this thesis are: TMR, configuration scrubbing and user-memory scrubbing. While other SEU mitigation techniques exist, these SEU mitigation techniques are common examples of ways to improve the reliability of an FPGA design, and they are used in the experiments of this thesis.

2.5.1 Triple Modular Redundancy

TMR is a well-known SEU mitigation technique that involves triplicating the original circuit and inserting voters (see Figure 2.7). There are many ways TMR can be applied [27]. TMR can be applied spatially or temporally [10]. Majority voters mask SEUs by propagating the dominant output of the triplicated circuit. If at least two of three redundant circuits are operating correctly, the output of the system will be correct as well. To avoid single point failures, voters themselves are often triplicated as well [12]. TMR protects the designs from errors in the routing resources, lookup tables, control signals, and internal state. While TMR overhead is at least $3\times$ in area, it has been shown to provide significant improvements in reliability on a number of FPGA circuits [28].

TMR cannot protect against SEU accumulation across more than one redundant copy. The granularity of TMR affects the amount of SEUs a design can tolerate before failing; fine grain TMR (i.e., TMR of the smallest components) is able to withstand more faults at the cost of additional area and delay due to the insertion of more voters [29]. By placing voters throughout the design, the granularity of TMR is made smaller. Figure 2.8 shows a fine grain TMR scheme that places majority voters in feedback paths [12]. By triplicating a design into smaller partitions, SEU are allowed to accumulate in ways that would otherwise cause the design to fail. That being said,

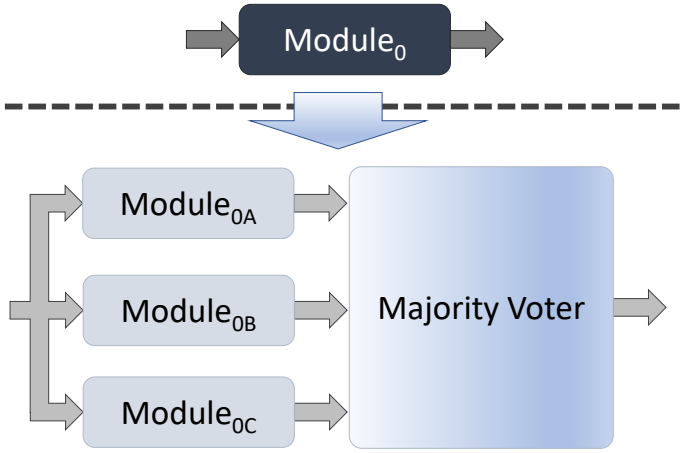


Figure 2.7: Triple Modular Redundancy (TMR).

SEUs can still accumulate within the same partition causing TMR to fail. In order to prevent this kind of failure, an SEU repair mechanism, such as scrubbing, must be combined with TMR [30].

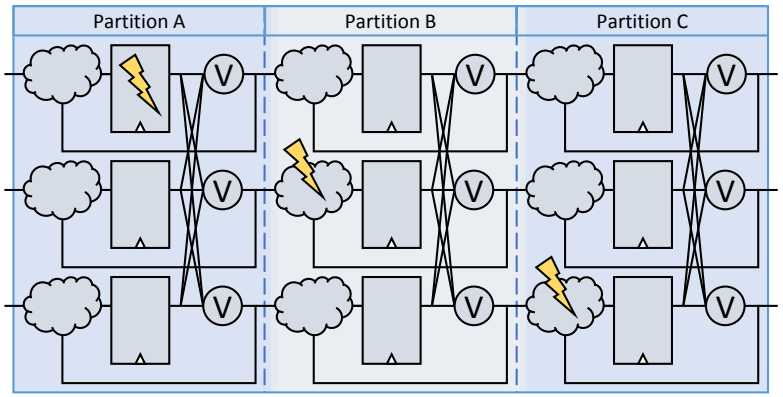


Figure 2.8: Fine-grain TMR with Voter Synchronization in the Feedback Path. So long as no more than one TMR domain per partition is damaged, the output will be correct.

It is possible for a single SEU to cause the entire system to stop functioning correctly. Any aspect of the design that is not protected with redundancy or fault tolerance presents an SPF vulnerability. Within an FPGA's architecture, it is also possible that an SEU breaks redundancy by causing failures in multiple circuit copies [31, 32].

2.5.2 Duplication with Compare

Another SEU mitigation technique based on redundancy is duplication with compare (DWC). DWC involves duplicating a module and comparing the outputs of the two copies against each other to detect errors. Similar to voters in TMR, detectors are inserted into the circuit to detect any discrepancies. A major distinction between TMR and DWC is that TMR is able to mask errors by propagating the majority vote of the three redundant circuits whereas DWC is only able to detect that an error has occurred; DWC cannot mask errors. DWC has lower overhead than TMR because only two copies are made rather than three. When DWC is employed, detectors themselves are often duplicated as well to prevent errors in detection logic from being reported as errors in the design under-test (i.e., a false-positive). When duplicate detectors are used, their output is encoded so that an error is only reported with both detectors detect an error. Using DWC for on-line error detection has been shown to be an effective means of detecting errors caused by configuration upsets in FPGA-based designs [33].

2.5.3 Configuration Scrubbing

Configuration scrubbing involves the rapid and continuous repair of upsets in configuration memory as they occur [34]. Configuration scrubbing is performed without interfering with the operation of an active design. In other words the application loaded onto the FPGA does not have to go off line for configuration scrubbing to take place. As such, configuration scrubbing can only repair upsets that affect bits of the programming data that are not changed by the design. User-memories are not protected by configuration scrubbing. Bits that are protected include the contents of look-up tables used to implement logic, interconnect and multiplexer configurations for routing nets between components, and other constant configuration values.

Configuration scrubbing prevents the accumulation of more than one upset in configuration memory that may break TMR. Configuration scrubbing can be performed in many different ways. External readback configuration scrubbing compares the current configuration of the FPGA against a golden copy to detect and repair SEU. External scrubbing has been shown to greatly reduce the sensitive cross section of a design [35]. Internal configuration scrubbing uses frame-based error correction codes (ECC) and global cyclical redundancy check (CRC) calculations to detect SEUs

and correct them when possible [14]. SEUs that affect user-memory bits cannot be repaired by configuration memory scrubbing. To repair SEUs that affect these bits, additional SEU mitigation techniques need to be applied.

2.5.4 User-Memory Protection

Internal memories that contain large amounts of state pose a unique challenge when faced with SEUs. Although a number of memory coding techniques can be used, these techniques do not adequately protect the memory from the accumulation of multiple upsets within a single memory word. This problem is especially prevalent in memories whose contents do not change very often (such as read only memories) as there is greater potential risk of SEU accumulation. In order to prevent the accumulation of upsets that may invalidate memory coding techniques, such as ECC, some form of memory scrubbing may be used in environments where soft error rate (SER) is high (e.g. space environments) [11].

Scrubbing BRAM bits repairs SEUs as they occur and prevents the failure of additional protection mechanisms, such as TMR or ECC, that can result from the accumulation of SEUs. One of the most common memories that may need to be protected with scrubbing are user block memories called BRAM modules. These modules are user configurable and provide a limited number of access ports to the associated BRAM bits. Protecting BRAM modules with scrubbing can be challenging if all available ports are already in use. Figure 2.9 shows the contents of a dual-port BRAM module being scrubbed. One port is dedicated to scrubbing and another port is a dedicated access port for the design. In this case, the memory module is triplicated and the majority vote between the three copies is written back to the memory to scrub away SEUs.

Scrubbing is performed in a three step process. First, a memory word is read out of memory. Second, the word is decoded and corrected through ECC or another form of memory protection. Finally, if an error is detected in the word, the corrected word is written back to memory to overwrite the upsets. Scrubbing can be performed deterministically by iterating through all of the words in memory, or probabilistically by scrubbing a memory word only when it is used by the design [36].

Other memory protection techniques, such as ECC, can also benefit from scrubbing. Encoding is used to mask SEUs in ECC protected memory. Without correcting upsets in memory, by

either overwriting them with new values or scrubbing them, eventually enough SEUs will accumulate to cause the ECC method to fail [11]. By providing memory scrubbing, memories affected by SEUs will not accumulate upsets.

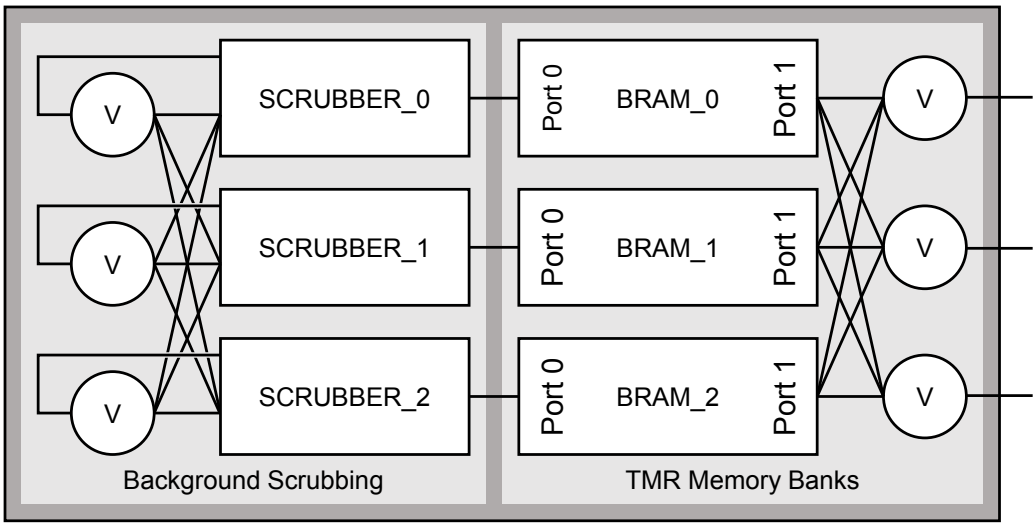


Figure 2.9: Internal BRAM Scrubbing

There are some limitations to the benefits that user-memory scrubbing provides. Additional resources are consumed by scrubbing logic. This logic may also be vulnerable to failure if not protected with SEU mitigation techniques. Also, memory that changes value often benefits less from scrubbing because the risk of accumulating SEUs that would invalidate ECC is lower. The benefits of scrubbing memory caches are limited because when an error is detected in a cache, the cache can be easily be flushed and refreshed instead of being scrubbed [37]. Another limitation of user memory scrubbing is that it can only repair errors that occur within a memory; it cannot prevent a error from being written to memory.

2.5.5 Complementary SEU Mitigation

Complementary SEU mitigation and repair techniques work together to improve an FPGA design's fault-tolerance and prevent SEU failure modes from occurring. TMR masks SEUs to prevent failure. Scrubbing repairs SEUs to prevent accumulation that would break TMR. The

reliability of a system can be significantly improved when both TMR (fault masking) and configuration scrubbing (repair) are used together [28]. Without configuration scrubbing, the benefits of TMR are limited (especially for long missions).

2.6 SEU Sensitivity Estimation

The likelihood of a design failing when a random upset occurs is the SEU sensitivity of an FPGA design. There are many factors that contribute to the SEU sensitivity of an FPGA design such as the size of the design and whether or not any SEU mitigation techniques were applied to it. The main idea is that there are bits within an FPGA design that will cause the design to fail if upset. These *sensitive* bits exist within the FPGA configuration memory for a particular design and are most likely different for each FPGA design. Upsets in these bits cause a design to fail by introducing the SEU failure modes discussed earlier into the design. The ratio of sensitive CRAM-bits to non-sensitive CRAM-bits reflects the SEU sensitivity of a design. The likelihood of an SEU occurring from radiation exposure also affects a design's SEU sensitivity. This term attempts to provide common ground between fault injection and radiation testing.

Knowing the SEU sensitivity of a design provides great insights to the potential risks of radiation exposure for FPGA designs and the benefits of SEU mitigation techniques. Every design has a different level of SEU sensitivity. SEU mitigation techniques need to be proven and the SEU sensitivity of a mitigated design can provide supporting data or reveal vulnerabilities. As newer FPGAs are released, changes in FPGA design SEU sensitivity can reflect the impact of changes in the fabrication process, and FPGA architecture on reliability. The SEU sensitivity of a design provides insight into every change and each new design.

SEU sensitivity of an FPGA design is often estimated by sampling a portion of CRAM-bits on the FPGA for sensitivity. Because newer FPGAs now have tens of millions or more CRAM bits and each design has numerous possible states, it is nearly impossible to exhaustively test each bit to estimate the SEU sensitivity of a design. Instead, a portion of CRAM cells are sampled for sensitivity through fault injection and radiation testing. Fault injection involves artificially emulating an SEU by altering the contents of CRAM-bits [15]. Radiation testing involves exposing the FPGA design to accelerated radiation that causes SEUs [16]. A key component of most SEU sensitivity testing approaches is being able to detect that a functional failure has occurred in the design. The

next chapter discusses aspects of SEU sensitivity testing and the benefits and limitations of various testing approaches.

CHAPTER 3. SEU SENSITIVITY TESTING APPROACHES

SEU sensitivity estimation plays an important role in radiation hardness assurance (RHA), SEU mitigation technique validation, and in improving the fault-tolerance of FPGA designs. Mission-critical space applications have rigid requirements for system reliability. Radiation hardness assurance seeks to qualify devices for space. So much goes into qualifying an integrated circuit for space [38]. Part of qualification includes SEU testing. Test guidelines and standards have been written for radiation testing [4, 39]. The benefits of SEU mitigation techniques can be proven through SEU sensitivity estimation. Additional unanticipated vulnerabilities can be exposed through SEU sensitivity estimation.

In order to estimate the SEU sensitivity of an FPGA design, there must be a means of observing failure events caused by SEUs, and the factors that lead up to failure must be recorded for analysis [16]. Many approaches have been developed to estimate the SEU sensitivity of an FPGA design through fault injection and radiation testing. Fault injection mimics the behavior of an SEU on the FPGA under test, and radiation testing exposes the FPGA to accelerated radiation sources that cause SEUs within the FPGA. Various test setups have been created to configure fault injection and radiation testing towards SEU sensitivity estimation of FPGA designs. This chapter discusses common elements found in FPGA SEU sensitivity testing approaches. SEU sensitivity testing approaches used in related works are presented and an on-chip error detection approach is introduced. The limitations and benefits of each approach are discussed.

3.1 General Approach for SEU Sensitivity Testing

SEU sensitivity testing is event based, where the event is usually the occurrence of a functional failure within the design. Most SEU sensitivity testing approaches depend on the ability to detect when the design under test is not operating correctly [16]. Testing approaches for SEU sensitivity estimation create an environment where a design can fail, and they measure and record

the factors that lead up to failure for analysis. Common elements are found in most SEU sensitivity testing approaches to support SEU sensitivity estimation.

There are five common elements found in most SEU sensitivity testing approaches. First, the design under test must be active. Second, the output of a design under test is usually monitored for errors. Third, the environment is set up to create possible failure conditions through fault injection or radiation testing. Fourth, factors leading up to failure, such as the number of of injected faults or amount of radiation exposure, are recorded for analysis. Finally, a method is provisioned to place the design back into a known good state so that additional failure events can be observed.

In SEU sensitivity testing, the design is kept active so that functional failures can occur and be detected. If a design sits idle, the effects of SEUs on the design may go unnoticed. Keeping as much of the design active during SEU sensitivity testing as possible allows failures caused by SEUs to propagate to the outputs of the design where they may be observed. Designs are typically kept active using a set of input vectors [16]. A clock signal is also usually provided to the design as well.

Two main methods are used to monitor the outputs of a design for errors. First, the outputs of a design may be compared against a set of golden output vectors that match the expected behavior of the design. Second, the outputs of a design may be compared against those of an identical design instance that operates in lockstep with the design under test. If any disagreement is found, a functional failure is detected and reported. Other methods for monitoring outputs could include monitoring checksums produced by the design.

Failure conditions are introduced to the active design through fault injection or radiation testing. Typically, only the design under test is exposed to factors that cause SEUs. Test control unit, functional error detection logic, and the golden output vectors or lockstep design instance are commonly separated from the design under test to make them fault-immune. For example, many SEU sensitivity testing approaches place the design under test on a separate FPGA. This FPGA is then injected with faults or placed in an accelerated radiation beam so that SEUs only occur within the design under test and not in other parts of the test setup.

The factors that lead up to a functional failure are recorded for analysis. For fault injection, the total number of fault injected and the total number of failures observed may be collected. This can be used to estimate the sensitivity of the design. Also the particular bit injected prior to a

functional failure may be recorded for reproducibility and failure characterization. For radiation testing, the total radiation exposure of the design and the total number of observed failures may be collected. Radiation exposure is usually measured in terms of fluence, and the SEU sensitivity of the design is approximated by the cross section of the design, or number of failures divided by total fluence.

Many failure events must be observed for the collected data to be statistically significant and representative of the SEU sensitivity for a particular design [15,16]. To support the observation of multiple failure events, most SEU sensitivity testing approaches provision a method of resetting the design, or resynchronizing it into a known good state so that additional failure events can be observed. Any necessary control signals are provided to the design. The process of resynchronizing the design for the next test run is often automated to assist in rapid data collection.

Most testing approaches organizes these five common elements into a test setup with an accompanying test fixture to support SEU sensitivity testing. The general setup for SEU sensitivity testing is described in the next section.

3.2 General Setup for SEU Sensitivity Testing

While each approach is different, the test setup for many SEU sensitivity estimation approaches is very similar. Each test setup typically has a host computer that interfaces with a test fixture. The test fixture holds the FPGA design under test and contains additional components to support the experiment. Figure 3.1 depicts the general setup of an SEU sensitivity test and the common components of a test fixture. Clock and control signals drive the design under test and provide a means to reset or resynchronize the design. A test control unit orchestrates the operation of the design under test with the rest of the test fixture. It keeps the design active by managing input vectors or can make sure the design operates in lockstep with a golden design instance if used. Functional error detection mechanism provides the means of detecting functional errors in real time. This mechanism usually compares the output of the design under test against a set of golden output vectors, or a fault-immune instance of the design operating in lockstep outside the scope of fault injection or radiation testing. If a disagreement is found, a failure is reported. This mechanism is especially important because most approaches depend on the ability to detect errors in order to collect good data. The external interface provides a way to communicate with the host

computer. Other common aspects incorporated into a test setup may include automated logging on the host computer and using heartbeats or watchdogs, (e.g., provided by the test control unit), to ensure that the design is still operational [16].

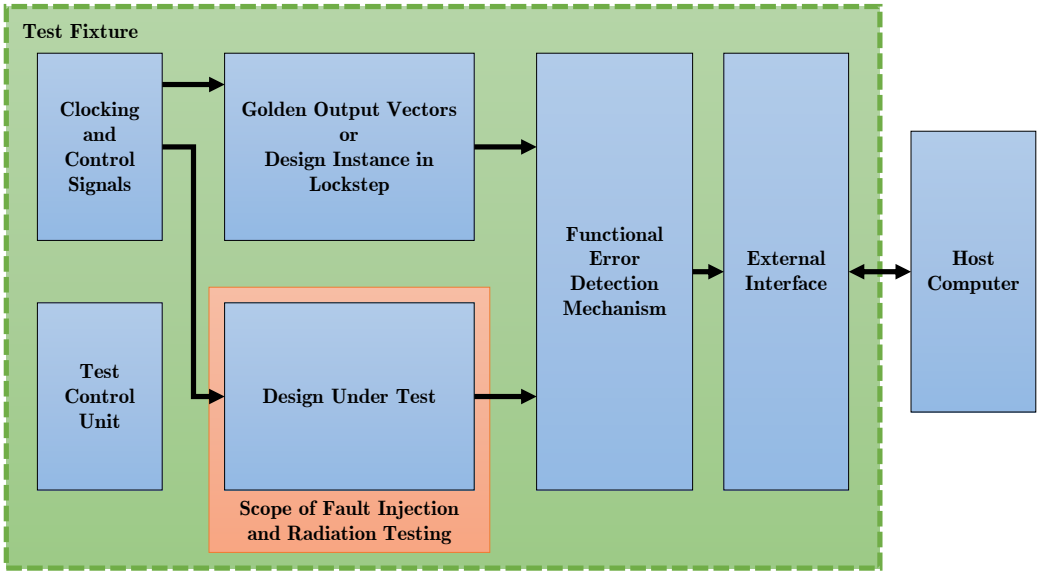


Figure 3.1: General Setup for SEU Sensitivity Test

It is important that design instances operate in lockstep so that they can be compared against each other to detect functional failures. When designs operate in lockstep, no instance advances to next computation until all instances have completed the previous computation, and all instances produce outputs in the same order. For many FPGA designs this is not hard to accomplish since computations usually have a predictable data flow path. So long as predictable design instances begin execution on the same clock cycle, they will operate in parallel. When design instances use roll back check pointing or perform out-of-order computation, a more complicated mechanism must be used to keep the design instances in sync [40]. The state machine can be used to ensure the synchronous operation of design instances and tag the appropriate signal samples for comparison in the case of out-of-order computation.

One of the most important SEU evaluation elements is the functional error detection mechanism. When evaluating the SEU sensitivity of a design, failure events are collected and recorded. To evaluate the sensitivity of a design to configuration upsets, there must be some way of know-

ing when the design has experience a functional failure. Many mechanisms can be used to detect functional failures but their primary goal is the same: to detect when an error has occurred within the design.

Many testing approaches separate the design under test from the other test fixture components so that the scope of fault injection or radiation testing only extends to the design under test and nothing else. This is typically done by placing the design under test on its own FPGA. Approaches that do not place the test control unit, error detection logic, golden output vectors, etcetera on the same FPGA as the design under test will be referred to as off-chip error detection approaches; whereas an on-chip error detection approach places additional test components on the same FPGA as the design under test.

3.3 SEU Sensitivity Testing Examples

The approaches discussed in this section specifically focus on estimating SEU sensitivity of SRAM-based FPGAs in harsh radiation environments and estimating the reliability benefits of SEU mitigation techniques. These approaches use an FPGA in fault injection or radiation testing to estimate the reliability of an FPGA design on a specific target FPGA. The reliability benefits of SEU mitigation techniques are estimated by comparing the estimated SEU sensitivity of a mitigated and an unmitigated version of the same design.

Fault injection and radiation testing are common methods for estimating the SEU sensitivity of a design operating on an FPGA. SEU sensitivity is defined as how prone to failure a given design is when an SEU occurs. The design operates on the FPGA while the FPGA undergoes fault injection or radiation testing. The rate of design failure with respect to injected faults or radiation exposure is the primary indicator used to determine SEU sensitivity. For fault injection, SEU sensitivity is typically measured in terms of mean upsets to failure (MUTF) but can also be measured in terms of the percentage of upsets that cause the design to fail. A lower MUTF or a higher percentage of *sensitive* bits means a greater SEU sensitivity. For radiation testing, SEU sensitivity is typically measured in terms of cross-section. The larger the cross section, the greater the SEU sensitivity.

Several testing approaches used in related works to estimate SEU sensitivity are presented in this section. Their test setup is describe and some results are included from experiments that use them. Benefits and limitations of each approach are discussed compared against each other.

3.3.1 FT-UNSHADES

FT-UNSHADES [41] emulates fault injections in ASIC circuitry using an FPGA as a fault injection emulation platform. This evaluation approach uses a formal verification tool to ensure that the ASIC netlist and its corresponding FPGA netlist maintain and guarantee equivalence, and then uses partial reconfiguration to inject faults into the user flip-flops. This approach places a golden and a *faulty* instance of the design on the same FPGA along with comparison and test control logic. Test vectors are loaded externally and faults are limited to flip-flops within the faulty instance of the design. Figure 3.2 shows the test setup used by FT-UNSHADES. Notice that only a single FPGA is used to implement this fault emulation approach.

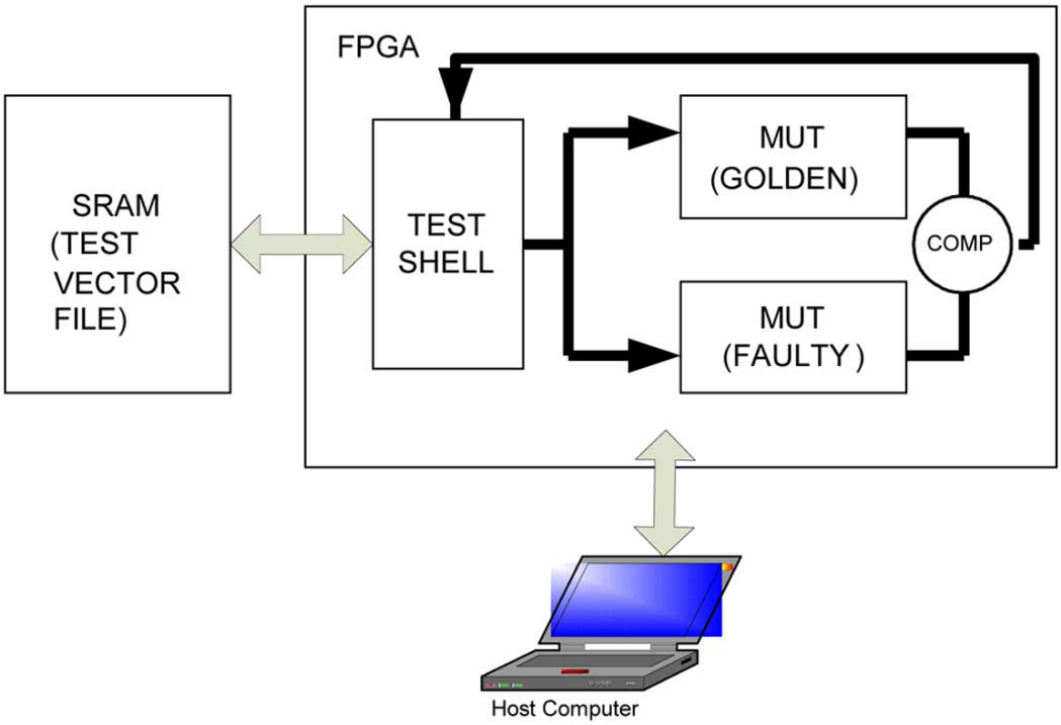


Figure 3.2: FT-UNSHADES test setup. Figure taken from [41].

Using this approach a LEON2 soft processor without mitigation was compared against an XTMR version (i.e., Xilinx automated TMR), and a fault tolerant (i.e., LEON3-FT) version. The most sensitive components of the processor were identified and protection provided by the mitigated version was verified [41]. This platform is used in [42] to prepare designs for radiation testing. The fault injection emulation approaches presented in [43] and [44] are similar to that of FT-UNSHADES.

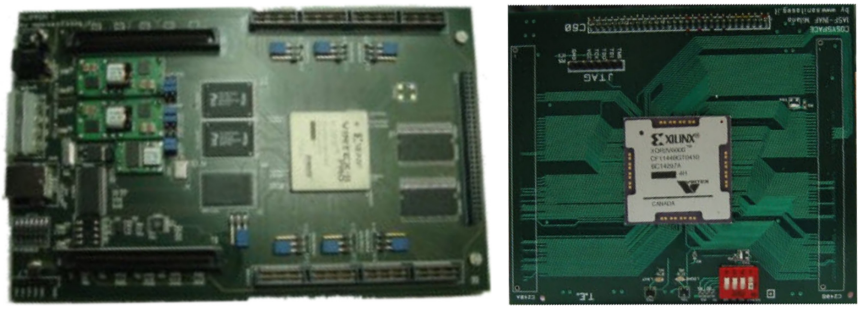
This is an on-chip error detection approach, and is similar to the on-chip error detection approach described in Chapter 4. It is on-chip because the test control logic, lockstep golden design instance and the error detection mechanism are all on the same FPGA. The biggest difference between this approach and the one described in Chapter 4 is that this approach limits faults to the design instance under test and the other approach allows faults anywhere on the FPGA.

Using an FPGA to emulate fault injection in ASIC circuitry accelerates ASIC fault-tolerance testing, but since faults are limited to flip-flips, only a small portion of FPGA architecture is included in an SEU sensitivity estimation obtained through this approach. This approach does not consider the effects of SEUs on logic or routing in an FPGA design. This approach is not designed for use in radiation testing. However, only a single FPGA is needed to implement this approach making it less expensive to implement than a multi-FPGA test setup. The approach could also feasibly be implemented on a ready made FPGA evaluation board.

3.3.2 FLIPPER

FLIPPER [45] is a fault injection approach that injects a fault using partial reconfiguration and uses stimuli and golden values derived from HDL simulation to determine when the design has failed. If the output of the FPGA does not match the expected golden values for the provided input stimulus, a failure has occurred. FLIPPER consists of two boards: the main board that manages fault injection, and a device under test board that contains the FPGA under test (see Figure 3.3).

In a case study, faults were injected every 18ms and allowed to accumulate until a functional fault was detected. For each injected fault, 26,000 testbench cycles are executed. Using the approach, the MUTF of various mitigation schemes were evaluated for an FPGA design. The plain design, without mitigation, had a MUTF of 337 accumulated faults. The Xilinx TMR version



(a) Main Board (b) DUT Board

Figure 3.3: FLIPPER test fixture consisting of two boards. Images taken from [45].

had a MUTF of 1,330 accumulated faults. Based on these results, Xilinx TMR without scrubbing reduced the SEU sensitivity of the evaluated design 3.9×.

Having a separate test board provides for extendability of the approach to newer devices and helps reduce reoccurring engineering costs [15]. The main board contains an FPGA which can be reused for new FPGAs, but at some point the main board would need to be updated with a newer FPGA. This approach leverages HDL simulation tools to provide stimulus and compare device outputs against golden output vectors. An advantage of using test vectors is that only a single target FPGA is needed.

3.3.3 SLAAC-1V

The SLAAC-1V board [46] is an FPGA fault injection and radiation testing approach that utilizes three FPGAs: one as the FPGA under test, one as a *golden* FPGA that runs in parallel lockstep with the FPGA under test and is failure-immune, and one for comparison, reconfiguration, and control logic of the other two. This approach allows for off-chip error detection of very complex designs without having to precompute golden test vectors. So long as the inputs are the same, the outputs of the golden and the device under test FPGAs should be the same also, except in the case of failure. This is an off-chip error detection approach because error detection logic is not on the same FPGA as the design under test. The golden lockstep design is also not on the same FPGA as the design under test.

A linear feedback shift register (LFSR) design was evaluated for SEU sensitivity using this approach. A fault was injected in the device under test every 267us. Between faults the circuit is

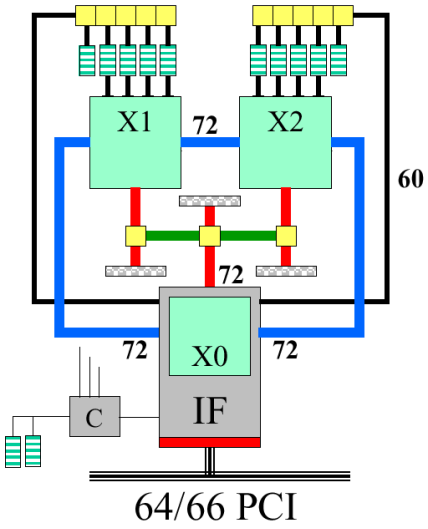


Figure 3.4: SLAAC-1V setup diagram. Figure taken from [46]. X1 and X2 are the golden and device under test FPGAs respectively. X0 is the comparator FPGA.

cycled and tested for discrepancies with the golden. All 5.9 million configuration bits on Xilinx Virtex XCV1000 were exhaustively fault injected. It was found that the tested LFSR design was relatively insensitive to injected faults.

The SLAAC-1V board was used in a later experiment that compared the SEU sensitivity found in fault injection against that found in proton radiation testing [25]. In this experiment, several versions of a multiplier and LFSR were tested. A 36-bit and 72-bit multiplier, and a 72-bit LFSR circuit demonstrated a 4.0%, 14.7%, and a 4.8% sensitivity rate respectively in fault injection, and a 4.9%, 17.2&, and 5.0% sensitivity rate in radiation testing respectively. This experiment helps to validate this approach by showing that fault injection data agrees with the radiation testing data.

In [10] several SEU mitigation techniques were compared using the SLAAC-V1 board approach to gather fault injection data. The fault injection campaign used in this experiment simulated configuration scrubbing by correcting the injected fault after checking for output errors. TMR and scrubbing in this experiment reduced the SEU sensitivity of a 36-bit counter and two finite state machine 2-3$\times$.

One advantage of this approach is that allows two test designs to operate in lockstep. For complex FPGA designs, comparing the output of two designs operating in lockstep can be easier

than comparing the output of a single instance against a set of golden output vectors. Unlike FT-UNSHADES, this approach places comparison logic on a separate FPGA. Doing so allows faults to be injected anywhere in the FPGA under test without disrupting the operation of the test setup. It also makes it possible to use the test setup in radiation testing. Some drawbacks of this approach are that multiple target FPGAs are needed and that a entire new board would need to be re-engineered to test newer FPGAs.

3.3.4 XRTC-V5FI

The XRTC-V5FI [47] approach consists of three main boards: the XRTC motherboard, the Xilinx Vertex 5 FPGA under test daughter card, and an external memory card (see Figure 3.5). There are two FPGAs on the mother board: the ConfigMon, and the FuncMon. The ConfigMon FPGA controls all of the reconfiguration and rapid fault injection, and the FuncMon is dedicated to detecting functional failures in the FPGA under test. Rather than compare the outputs of a design against external golden values to detect failures, this approach compares the outputs of two identical designs instances on the FPGA under test against each other. Several signals are compared externally between the two, these signals are sent to the FuncMon FPGA. It is not likely that a single fault will affect both instances in exactly the same way, and since the comparison logic is separated from the FPGA under test, failure of the error-detection logic is unlikely as well.

Because the comparison and error detection logic is not on the same chip as the FPGA design being tested, this approach is considered an off-chip error detection approach. However, the concept of having two instances of the same design on the FPGA under test, operating in lockstep, where both instances are vulnerable failure is an important aspect of the on-chip error detection approach presented in Chapter 4. Rather than have one design instance serve as the design under test and another instance serve as a golden copy that is failure-immune for comparison, both instances are placed under test. If either copy fails, than a failure has occurred. This is an important concept that is also central to DWC SEU mitigation [33]. It is the driving concept behind an on-chip error detection approach for SEU sensitivity estimation.

The XRTC-V5FI board has been used in several experiments. One experiment evaluated the SEU sensitivity of various soft-core processors [47]. This experiment used fault injection to exhaustively test each bit on the target FPGA for SEU sensitivity. Several benchmark programs

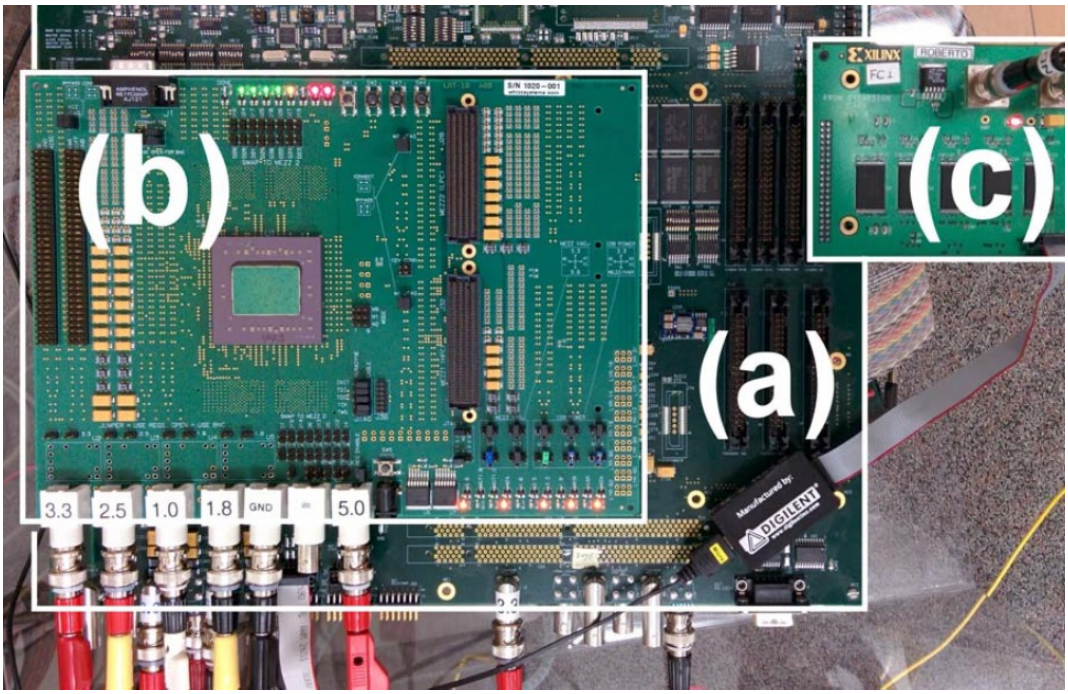


Figure 3.5: XRTC-V5FI test fixture. Image taken from [47]. The XTRC motherboard (a), a Xilinx V5QV daughter card(b), and a PROM memory card (c) are displayed.

were tested on each of the soft-core processors. In another experiment [28], three circuits were tested using this approach with fault injection and one design was further tested with radiation testing. It was found that TMR data from fault injection matched radiation data within the error bounds. A slight discrepancy was found between non-TMR fault injection data and radiation data, but that was attributed to radiation exposure during book keeping activities.

Like FLIPPER, the XRTC-V5FI SEU sensitivity testing approach uses a separate daughter card for the device under test. This reduces reoccurring engineering costs and allows the same motherboard to use when testing additional target FPGAs. Unlike the SLACC-V1 approach, which places identical FPGA design instances on two separate FPGAs, the XRTC-V5FI approach places identical FPGA design instance on the same FPGA. This reduces the number of target FPGAs needed to do lockstep comparison for functional error detection, but it also reduces the size of FPGA designs that can be tested using this approach since both instances must fit on the same FPGA together. This approach also allows for separate biasing of the test daughter card, which is helpful for SEE testing in an accelerated radiation beam tests [16].

3.3.5 Multiple Development Boards

[48] reduced the overhead of developing a custom test platform by using two ready-made Xilinx KC705 development boards to evaluate the feasibility of using the GTX transceivers on Xilinx 7-series FPGAs in high energy physics experiments. This experiment used one development board as a *service* board and the other development board as the device under test board. There were 13 multi-gigabit transceiver lanes between the two boards used to send and receive data. At least one custom crossover board was made to enable coupling of the available transceivers. Both boards had frame generator and frame checker logic operating on the FPGA fabric to continuously fill the 3.125 Gb/s of available bandwidth for each lane. It was found through proton radiation testing that the MTTF of a single lane is 2.02×10^7 seconds or 234 days of continuous operations. This was lower than the acceptable failure rate for the targeted application. However, some of the failures were attributed to unmitigated FPGA logic on the device under test board used to capture and log multi-lane errors. This experiment shows ready made FPGA boards being used for SEU sensitivity estimation and suggests that any on-chip test logic should have SEU mitigation techniques applied to it where possible.

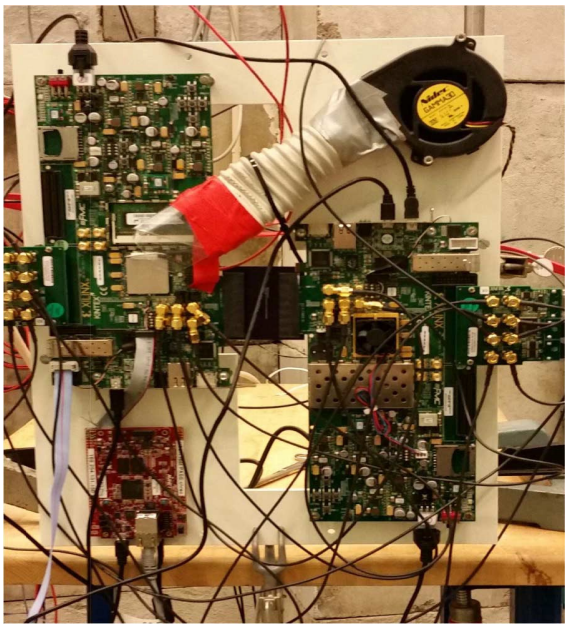


Figure 3.6: Multiple development board test setup. Image taken from [48].

One advantage of this approach is that it uses ready-made FPGA development boards, which can be more cost effective than designing custom test boards. A disadvantage of this approach is that two development boards are needed. Interfacing between two development boards can be challenging and would need to be redesigned for each new development board used. Also, as newer FPGAs are made available, the cost of a single development board may increase, making a multiple development board approach cost prohibitive.

3.4 Motivation for an On-Chip Error Detection Approach

Each of the approaches presented carry with them a number of advantages. Using on-chip error detection to estimate SEU sensitivity of FPGA designs is driven by the possibility of combining the advantages from all of these approaches into a low-cost single FPGA solution that is easier to implement. The on-chip error detection approach proposed by this thesis combines the following advantages from each of the approaches covered in this chapter.

- FT-UNSHADES – the entire test fixture is on a single FPGA.
- FLIPPER – test vectors can be used to stimulate a design and detect errors.
- SLAAC-1V – two or more complex designs can operate in lockstep for error detection.
- XTRC-V5FI – multiple designs instances can be placed on the same FPGA.
- Multiple Development Boards – ready-made development boards can be used.

Each of these aspects reduces cost and increases the flexibility of the on-chip error detection approach for estimating SEU sensitivity of FPGA designs. By only using a single FPGA and by using ready-made development boards, the cost of entry is greatly reduced, especially for higher end FPGAs. Being able to use test vectors or another design instance operating in parallel to detect errors makes this approach easier to adapt and implement for various designs. Like the other approaches covered in this chapter, valuable SEU sensitivity estimations can be collected using an on-chip error detection approach.

Another strong advantage of using on-chip error detection to estimate SEU sensitivity of a design is that many instances of a smaller FPGA design can be tested all at once. Smaller FPGA

designs are typically less sensitive to configuration upsets due to their reduced area usage in the FPGA. Gathering statically significant data from fault injection and radiation testing of a single instance can be challenging. This is due to the high number of random upsets that must occur before the design fails and the large number of failures that must be observed for the data to be significant. One way to work around this issue is to fill up the FPGA with many design instances and record whenever an instance fails. This effectively parallelizes testing and accelerates data acquisition for fault injection and radiation testing. This approach is hard to implement with off-chip error detection due to the limited number of I/O ports available on an FPGA. On-chip error detection compares the outputs of design instances internally without using I/O resources.

On-chip error detections comes with some disadvantages. Because multiple instances are placed on the same FPGA, designs that consume more than half of the available resources cannot be evaluated using this method. A false-positive is an error reported by the error detection logic that did not actually occur in the design under test. A false negative occurs when an error in the design under test goes undetected by the error detection logic. The error detection logic itself is vulnerable to failure (e.g. returning false-positives or false-negatives) because it too is on the FPGA under test and within the scope of fault injection and radiation testing. Few I/O resources are not included in the evaluation because outputs of the design under test are compared internally. Because no external golden value is available, it is difficult to validate the on-chip error detection logic. While off-chip error detection approaches may provide a more accurate evaluation, on-chip error detection is a useful means of roughly approximating the reliability benefits of SEU mitigation techniques and identifying issues that limit the benefits of SEU mitigation techniques.

In the on-chip error approach proposed, the components found in the test fixture of Figure 3.1 are all placed on the same FPGA as the design under test. The entire test fixture is placed within the scope of fault injection and radiation testing. This presents interesting implications for collecting good data. For example, since the error detection logic is also exposed to fault injection and radiation exposure, confidence of results may be negatively affected. Challenges and proposed solutions for using on-chip error detection to estimate the SEU sensitivity of FPGA designs are discussed in the next chapter. Test implementations for three FPGA designs using on-chip error detection logic are also included in the next chapter.

CHAPTER 4. ON-CHIP ERROR DETECTION APPROACH

On-chip error detection has the same basic elements as other SEU evaluation approaches, but it places most of the needed logic on the same FPGA as the design under test. The basic layout of an on-chip error detection test fixture is shown in Figure 4.1. Clocking and control signal are used to drive the design instances and the additional circuitry needed for on-chip error detection. A test state machine is used to orchestrate the parallel operation of design instances and provide for other needs. The design instances are copies of the design under test and functional error detection logic is used to determine if an error has occurred in any instance. To monitor the error status within the flow of fault injection or radiation testing, an external monitor interface is provided.

This chapter describes the implications of placing most of the test fixture on the same FPGA as the design under test. Doing so places the test circuitry within the scope of fault injection and radiation testing. To maximize the utility of this approach, special design consideration must be made. To reduce SEU induced failures within the on-chip error detection circuitry, SEU mitigation techniques such as TMR and scrubbing are applied to all SEU evaluation elements. State machines, functional error detection mechanisms, external interfaces, etc., all have some type of redundancy or fault tolerance built into them. To make sure that the test fixture and design under test are not halted by an SEU, a heartbeat is also made available for external monitoring. Every effort is made to maximize the accuracy of this approach. Implementing each of the basic components of an SEU sensitivity test fixture for on-chip error detection is discussed in this chapter. Designs used in the experiments of this thesis are also presented and the configuration of the on-chip error detection test fixture for each design is discussed.

4.1 Clocking and Control Signals

Clocking and control signals are typically high fanout nets within an FPGA that connect to many parts of an FPGA design and have a large impact on functionality. For on-chip error detec-

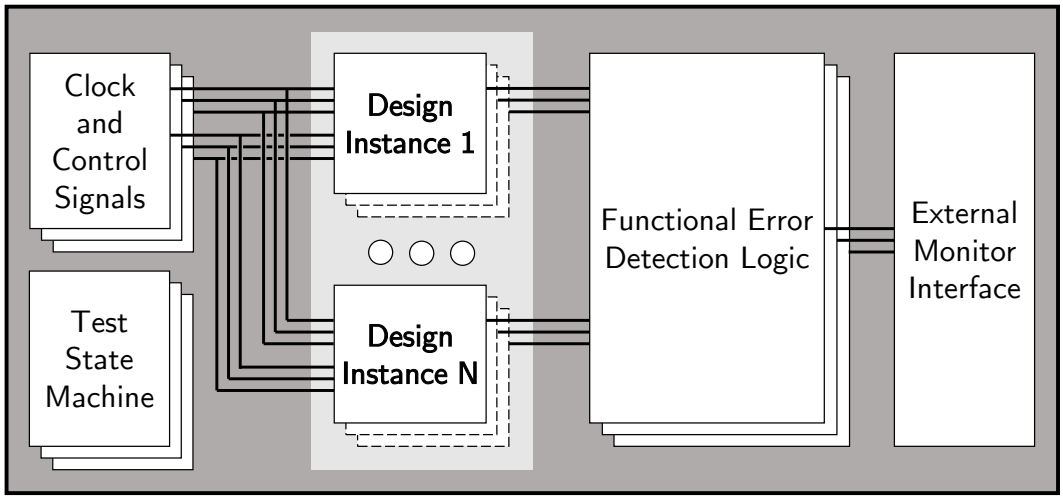


Figure 4.1: Components of On-Chip Error Detection

tion, clock signals drive SEU evaluation elements as well as all of the design instances. Control signals, such as reset or clock enable, can also cover a large area of the FPGA. Because these nets are so large, special care must be taken to make sure that when an SEU affects these signals it does not cause a system wide failure. If no redundancy is applied to these signals, they present a significant SPF vulnerability to the functionality of the system.

Using redundant clock signals improves on-chip error detection, but also introduces challenges. If redundant clock signals are used, it is important that all signals are in phase with each other. TMR in FPGAs requires that all three TMR domains operate synchronously with each other. If the TMR domains are out of phase with each other, majority voters will not be able to select the correct output between the TMR domains. TMR across clock domains requires special considerations as well [49]. For on-chip error detection, redundant clocks should be used where applicable.

Most FPGA development boards provide readily available clock sources that can be used for on-chip error detection through mitigation. A single clock source can be split into three in-phase clock networks by passing the incoming signal through three separate global buffers after entering the FPGA. This is similar to the mitigation approach used for global signals in [42] where singular signals are triplicated after being registered. This mitigation method can be used for other control signals as well and provides redundant in-phase clock networks with a reasonably low

impact on SEU sensitivity. Using this mitigation method removes the overhead of creating three separate sets of in-phase signals.

4.2 Test State Machine

A test state machine is used by on-chip error detection to control design instances and provide for other needs. The state machine will ensure that the design instances operate in lockstep. A heartbeat is also managed by the test state machine to indicate the continued operation of the test fixture and design instances. If the design instances are dependent on input stimulus or golden output vectors are available, the test state machine can manage provided input stimulus and select the appropriate golden output vector for comparison. The test state machine can also arbitrate between on-chip error detection and external monitoring by managing the error status registers and capturing samples for off-chip error verification.

A heartbeat is used by on-chip error detection to filter out common mode failures (e.g. design instances failing in exactly the same way). The heartbeat indicates the continued operation of the test fixture and design instances. It is possible for a single SEU to halt the operation of the test fixture, disrupt the operation of more than one design instance, or adversely affect the entire system. If an SEU affects all design instances in exactly the same way, it is likely that the design instances have ceased operation. In the case of halting design instances, functional error detection may not report an error because their output signals may still agree, but the heartbeat signal would reflect the failure. The heartbeat signal can be a counter that is incremented by the state machine only when the design instances have reached a certain checkpoint (e.g., finished a computation, completed a set of input vectors, concluded a program run). This counter is made available for external monitoring. Having a heartbeat signal indicates that the experiment is still operational.

If the design instances depend on input stimulus vectors or golden output vectors, the test state machine can control which input vector is provided to the design instances, or which output vector is used for functional error detection. These test vectors may be available off-chip from memory modules, JTAG or UART interfaces, etc., or they can also be placed on-chip with the test fixture. On-chip test vectors must be small enough to fit into the FPGA, and they must be protected by TMR and scrubbing or some other type of fault-tolerance. Mitigation is applied so that SEUs in on-chip test vectors have minimal influence on error detection results. The state machine can

act as a memory controller into provided test vectors and orchestrate the administration of the test vectors.

The test state machine also arbitrates the interaction between on-chip error detection and the external monitor interface. The error status register and heartbeat signal are made available for external monitoring. A design output sample may be wanted for external error verification, or the external monitor may want to reset the design instances. The test state machine can capture the heartbeat signal for external monitoring and reset the error state register after it has been sampled. It can also capture a design output sample at set intervals for external error verification and reset the design instances as requested by external monitoring. The state machine enhances the interaction of on-chip error detection with external observation of the experiment.

Like the other SEU evaluation components of on-chip error detection, the test state machine must also be mitigated to improve the functionality of the test fixture. There are many ways to mitigate a finite state machine. State encoding can be used or TMR can be applied to the entire state machine. TMR has demonstrated significant improvement in reliability of finite state machines over alternate encoding methods [10]. Figure 4.2 shows how TMR can be applied to the test state machine. The output signals and registers of the state machine are fed into redundant copies and the majority vote between all three copies are used in the next state logic of the state machine. Configuration scrubbing also benefits the reliability of the test state machine by repairing upsets as they occur. This prevents the accumulation of upsets that might break TMR. By mitigating the test state machine in this manner, error detection results are less likely to be adversely influenced by SEUs in the test state machine.

4.3 Design Instances

On-chip error detection is used to evaluate the effects of SEUs on a target design. The entire test fixture is built to detect failures within an FPGA design caused by an SEU in the design. Significant effort is made to ignore SEUs within the test fixture itself, as the effects of SEUs in the target design is of primary interest. The design under test is integral to the purpose of on-chip error detection and must be instanced within the test fixture.

One or more instances of the design under test may be placed within the on-chip test fixture. If input and output test vectors are provided, a single instance of the design under test can be used

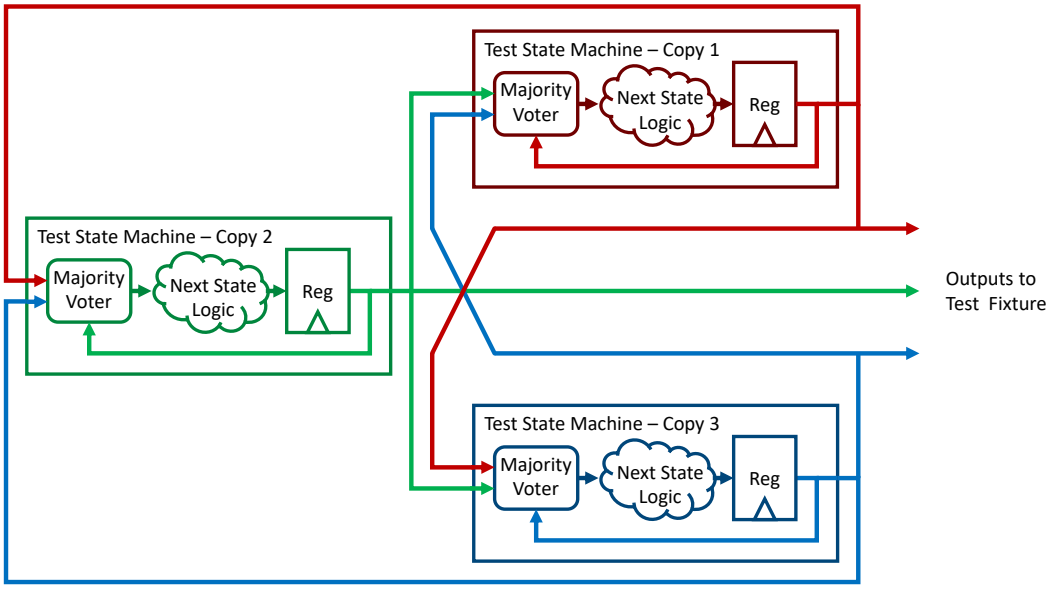


Figure 4.2: Test State Machine with TMR Applied

for SEU evaluation. When test vectors are not available, two or more instances of the design must be used. Either golden output vectors are used for comparison or the redundant copies of the design act as golden copies for each other. If ever the design instances disagree with the each other, or with the expected output vectors, a functional failure is detected. Typically, more than one design instance is used for on-chip error detection.

Placing more instances within the test fixture increases data acquisition but also increases functional error detection complexity. More design instances consume a larger area on the FPGA, thus as SEUs occur they are more likely to affect an instance of the design. Increasing the number of instances increases the SEU sensitivity of the entire test fixture; more failures can be observed with the same number of injected faults or exposed radiation. Thus smaller designs can be instanced hundreds of times in a single test fixture to accelerate data acquisition. Collecting data faster is useful because of time limitations and the cost of radiation testing. However, increasing the number of instances complicates functional error detection, control signal distribution, and timing closure as more signals need to be handled.

The design instances and the rest of the on-chip error detection test fixture must fit on the same FPGA. If a design consumes too many resources, there will not be enough available for

multiple design instances and the additional logic needed for on-chip error detection. This limits the number of designs that can be tested using this approach.

When evaluating multiple version of the same design, the same test fixture should be used for each evaluation. It is common to test an unmitigated version of a design and compare it against an SEU mitigated version [42]. Either multiple instances of the unmitigated design or multiple instances of the mitigated design are instanced together. Versions are never mixed for testing, (e.g., one instance of unmitigated and one instance of mitigated in the same test fixture). If a simplex design were being tested the triplicated control signals from the test state machine would be reduced to a singular signals to drive the design under test, and output comparison signals would fan out to redundant error detection mechanisms; only one of the triplicated clock networks would drive the design. For a TMR version of the same design, the same test fixture would be used; however, since the top level ports and output comparison signals of the TMR design are triplicated, triplicate test logic signals are not reduced to singular signals to drive the design under test. The same number of instances could be used when testing different versions of a design to keep the on-chip error detection overhead roughly the same.

4.4 Functional Error Detection

On-chip error detection uses comparison logic to detect functional failures within a design. Design instances are compared against each other, or outputs of the design are compared against golden test vectors. If any disagreement is found, an error is detected. The concepts covered here are very similar to those used by DWC [33]. Select signals from the design are used for comparison. Thanks to the flexibility of FPGA fabric, the design space for comparison schemes and comparison logic implementation is very large. This section covers important aspects and possible implementations of comparison logic for use in on-chip error detection.

4.4.1 Signal Selection

Select signals from an FPGA design are used for comparison, to detect functional failures. These signals will be compared against those of other design instances, or golden output vectors,

on a clock-by-clock basis to provide fast error detection. There is no standard as to which signals should be selected for comparison.

Not every signal within a design instance can be compared against those of other instances at the same time. Doing so would consume too many resources and not provide accurate results because not all of a design is active at once. If an error occurs within an FPGA design, it is likely that it will propagate to the outputs of a design [16], or be reflected in other significant signals such as the bus lines of a soft core processor. On-chip error detection selects significant signals of a design for comparison. These signals are compared on a clock-by-clock basis synchronous to the operation of the design. If a disagreement is found among these signals, the error status register is set and remains set to signify that an error has occurred.

Whichever signals are selected for comparison should reflect the operation of the design. Selected signals should be active when the design is active. The output signals of a design are a good candidate for selection, but do not necessarily reflect proper operation of the design. Consider for example the output signals of a soft core processor. If only the console output is compared, computations that never reach the console will not be compared. Too granular a selection may be an unwise choice as well. If intermediate computation is done out of order, but the result is the same, comparing individual registers may not produce accurate results either.

Selecting fewer signals for comparison reduces the size of functional error detection logic, but also reduces the chance of detecting an error. Selecting more signals for comparison increases the likelihood of detecting and error, but also increases the size of comparison logic. If many signals are compared, a large reduction network is needed to reduce the error flags to a single output [33].

4.4.2 Comparison Schemes

The selected signals of design instances can be compared in a variety of ways. Three of primary interest are: a one-to-one comparison scheme, a one-to-many comparison scheme, and a cascaded one-to-one comparison scheme. In a one-to-one comparison scheme there are only two instances of the design, and their selected signals are compared against each other. This is good choice for large designs. In a one-to-many comparison scheme, the selected signals of one instance are compared against those of many other instances. This is helpful when a small design

must be replicated many times to fill the FPGA. Finally, a cascaded one-to-one comparison scheme creates two chains of design instances, where the output of one instance feeds into the inputs of another, and only compared the selected signals of the last instances on each chain. This reduces the size of comparison logic by allowing errors to propagate through the individuals chains to the last instances on the chains. These three comparison schemes are later used in the evaluation of three benchmark designs.

4.4.3 Functional Error Detection Mechanisms

One of the most important parts of the on-chip error detection test fixture is the functional error detection mechanism. Having this mechanism on-chip provides great flexibility, but like the other components it too must be protected from the negative effects of SEUs that could skew the reported error results. There are two main methods that can be used to protect on-chip functional error detection from SEUs: through TMR or DWC. Figure 4.3 shows a functional error detection logic that if protected with TMR and another mechanism that is protected by DWC. Functional error detection logic compares and reduces the outputs of the design under test and the golden design to a single bit that indicates whether or not a functional error has occured. The TMR protected circuit triplicates the functional error detection logic and votes on the output of the three copies to mask any SEU within the error detection logic. The DWC protected circuit duplicates the functional error detection logic and masks SEU in the detection logic by only reporting an error when both copies report an error at the same time.

The protection method that reports the fewest false-positive or false-negative results is the one that should be used for on-chip error detection. Several of the experiments included in this thesis are set up to estimate the SEU sensitivity of a design using either a TMR protected or a DWC protected functional error detection mechanism. A comparison is made between the approaches based on the returned results.

4.5 External Monitor Interface

To facilitate external monitoring of the error status, an error status register is used to indicate whether or not an error has occurred. Functional error-detection logic operates on-chip at the

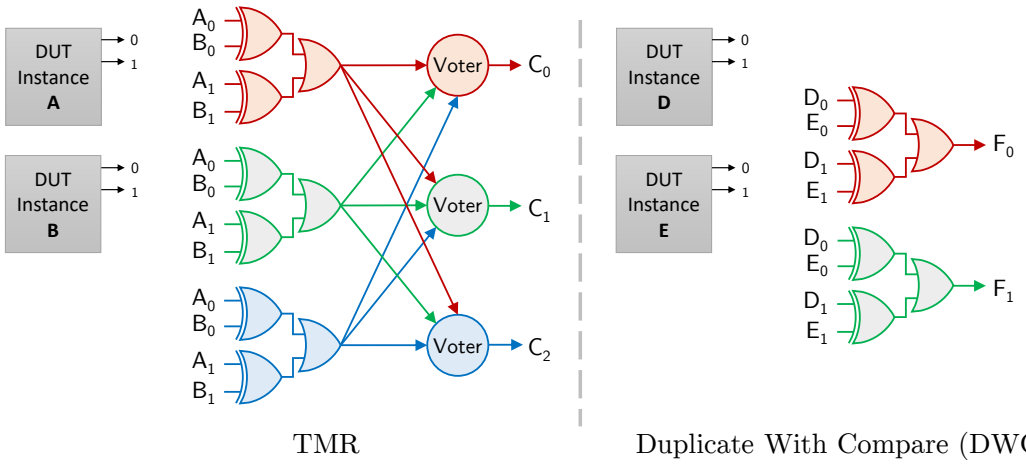


Figure 4.3: TMR Protected and DWC Protected Functional Error Detection Mechanisms

same rate as the design under test. An error may only be detected for a single clock cycle. The error status of the design under test is typically polled at a much slower rate through the external monitor interface. The error status register begins low and when an error is detected by the on-chip functional error detection logic the register is then set and remains set so that the next time the register is polled externally, the error can be reported. Like the other components of on-chip error detection, this register should also be protected through redundancy.

Any number of communication protocols can be used to externally monitor the error status of a design under test. JTAG and UART are simple communication standards that can be used to interface between the test fixture and an external host. For the experiments in this thesis, JTAG is used to periodically read the error status register of a design to collect data for analysis.

4.6 False Positive Bias

Due to high fan in and single point failures in the routing of reduction networks [33], an SEU in the reduction network can cause a false-positive to be reported (i.e., an error in the error detection logic that is reported as an error in the design under test). The following reasons suggest that an SEU in the functional error detection logic is more likely to cause a false-positive than to mask an actual error in the design under test (i.e., a false-negative). First, an SEU in the reduction network of the functional error detection logic may cause an error to be reported. Second, an SEU in the reduction network may prevent a portion of the comparison signals from being monitored,

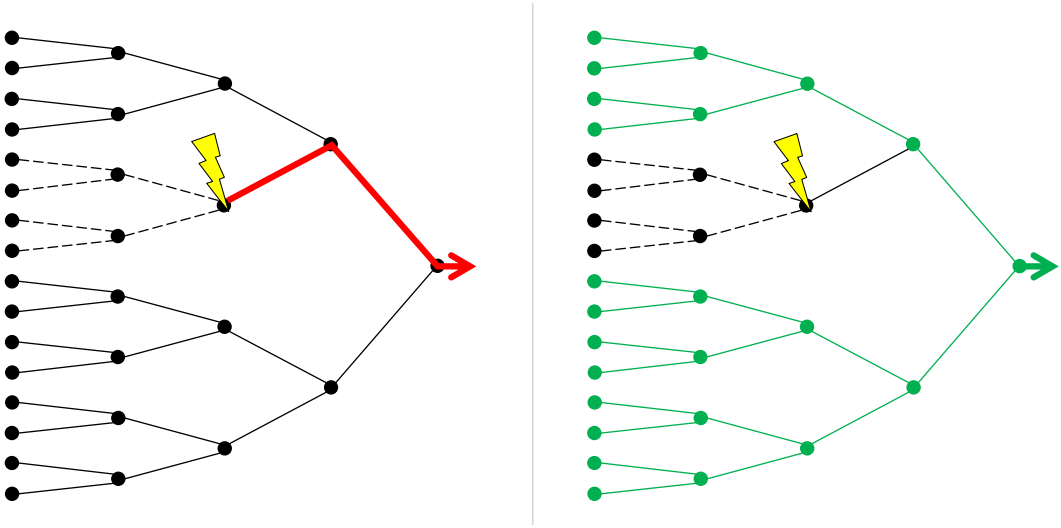


Figure 4.4: False-Positive Bias: An SEU in the functional error detection will either cause false-positive error to be reported (left), or disconnect a portion of the reduction network while the rest of network prevents a false-negative (right).

in which case the rest of the error detection network is still on-line. Theses cases are depicted in Figure 4.4. With part of the error detection logic disable, part of the network is still listening for errors. Error are detected on significant output signals of the design. When an SEU occurs within the design, a single fault is introduced. As this fault causes an error to propagate to the output signals being monitored for errors, it is likely that the original single fault will manifest as multiple bits disagreeing between design under test and golden values. Finally, when an error occurs within a design, it will likely manifest for more than a single clock cycle, which gives on-chip functional error detection logic more opportunity to detect the failure. Thus an SEU in the functional error detection logic may be more likely to cause a false-positive than a false-negative. The reduction network is protected by redundancy to try and minimize the number of false-positives that are reported.

False-positives in the results represent the SEU sensitivity of the test fixture logic. These results are mixed into the actual failures detected in the design. Thus a portion of the SEU sensitivity estimate reported by on-chip error detection is attributed to SEUs in the test fixture logic itself, and the SEU sensitivity estimate is actually an overestimate because of this influence.

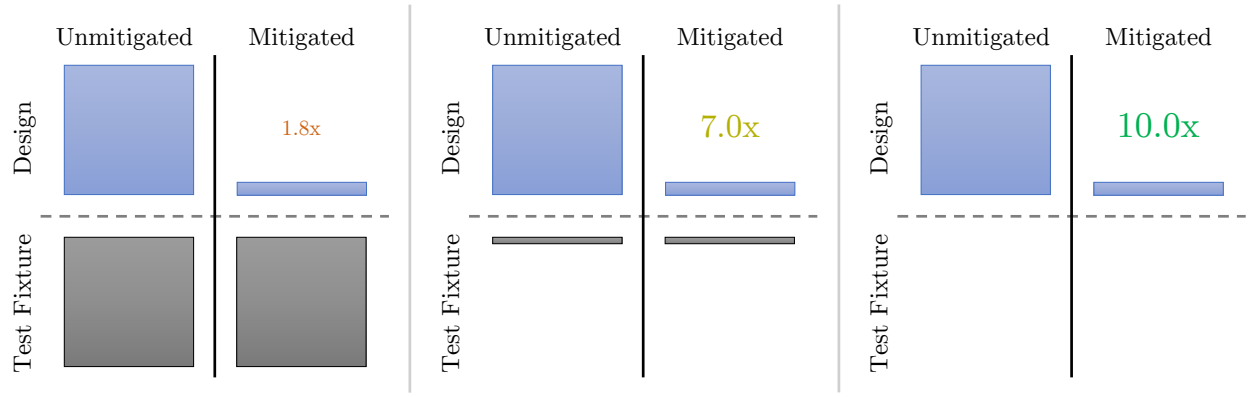


Figure 4.5: Minimum Influence on Results: Assuming a false-positive bias in the test logic, if the shared overhead of false-positives caused by SEUs in the test fixture is very large, the improvement demonstrated by applying SEU mitigation is limited (left). If the overhead were greatly reduced, more improvement would be demonstrated (center). Finally, if the overhead were removed completely, the most improvement would be demonstrated (right).

4.7 Minimal Influence on Results

When the SEU sensitivity of a design is tested with on-chip error detection, a portion of the SEU sensitivity reported is caused by SEUs in the test fixture logic itself. This sensitivity overhead influences the results and can make it difficult to estimate the difference in SEU sensitivity between the baseline unmitigated an mitigated versions of a design. A very large overhead shared between two different estimates makes the difference between the estimates look small. For example, say a design was estimated to be 20% sensitive (i.e., one in five SEUs causes a failure) and another design was estimated to be 11% sensitive (i.e., one in approximately nine SEUs causes a failure), but there is a shared test fixture sensitivity of 10%. When comparing the two designs, it appears that the second design is only 1.8× less sensitive than the first even though considering the shared overhead, the second design is actually 2× less sensitive than the first. This effect is lessened the smaller the overhead becomes. Figure 4.5 demonstrates what happens as the false-positive overhead of the test fixture is reduced.

In designing an on-chip test fixture, great care should be taken to minimize the SEU sensitivity of the test fixture itself so as to not skew the results. This is done by applying the techniques described in this chapter to the on-chip test fixture: triplicating the clock and control signals, triplicating the test control state machine, adding redundancy to the functional error detection mechanism and being selective in the number and importance of signals used for comparison to detect

failures. By reducing the influence of the test fixture on the reported the detected errors, the results become more meaningful and targeted to the actual design being tested.

4.8 On-Chip Error Detection Benchmark Design Testing

The experiments in this paper accomplish three main goals. First, they demonstrate the individual and combined benefits of complementary SEU mitigation techniques. Second, they compare results obtained from TMR protected and DWC protected functional error detection mechanisms to try and determine which approach is better. Finally, they compare the SEU sensitivity of FPGA designs on two different FPGA architectures with and without SEU mitigation techniques applied to the designs.

To accomplish these goals a set of three FPGA designs were selected: a LEON3 soft processor, a finite state machine called the ITC'99 B13 design, and a 128-bit AES encryption core. A recent study [42] has proposed a set of reliability benchmark designs to assist in research related to FPGA reliability. The B13 design is among the proposed set; but more importantly, this study suggests that FPGA designs held in common can be used to better understand the impact of factors that affect reliability. By comparing the SEU sensitivity of FPGA designs with and without SEU mitigation, on different FPGA architectures, or with variations in the test fixture used for estimation, the influence of these factors on SEU sensitivity can be exposed and better understood.

The sections that follow describe the FPGA designs used in the experiments of this thesis. General information is present about the design and its configuration for testing. Specifics are given as to how the on-chip error detection test fixture is configured for the particular design, and the kinds of reliability factors tested by each design are covered. The general idea is that the SEU sensitivities of three FPGA designs are tested with different SEU mitigation techniques applied, across two different FPGA architectures, with variations in the test fixtures used for SEU sensitivity estimations. With the data that is collected: the benefits of SEU mitigation techniques are demonstrated, a cross-architecture comparison is made of two FPGA architectures, and the best means of protecting on-chip error detection circuity from SEUs studied.

4.8.1 LEON3 Soft Processor

The LEON3 is an open-source 32-bit soft-core processor that is technology independent and can be implemented on SRAM-based FPGAs. The processor is well documented and has a strong user group for support [50]. For this work, only the core architecture of the LEON3 and a minimal set of peripherals are included in the experiment. Caching is also disabled to facilitate error detection. This configuration limits the scope of possible SEU failure modes. It also allows the experiment to focus on the benefits of SEU mitigation and repair techniques, as observed in fault injection and neutron radiation testing.

For the LEON3 experiments in this thesis, a one-to-one lockstep comparison scheme is used for on-chip error detection. Two independent instances of the LEON3 processor are placed on the same FPGA and compared against each other as they operate in lockstep. To provide a relatively low-level of fault detection, 108 bus signals[1] from each LEON3 processor are compared on a clock-by-clock basis. Because caching is disabled, bus activity is elevated and better reflects the functional state of the processor and connected peripherals. Rather than providing input vectors to stimulate the design and keep it active for error propagation, each LEON3 processor executes its own copy of the Dhrystone 2.1 benchmark in a continuous loop. The comparison and reduction logic used for functional error detection has TMR applied to it to protect against SEUs in functional error detection mechanism logic. The same testing fixture is used for all design variations.

Using on-chip error detection, the individual and combined benefits of complementary SEU mitigation techniques are demonstrated. To accomplish this, five variations of SEU mitigation techniques are applied to the LEON3 design: configuration scrubbing, TMR, TMR with configuration scrubbing, TMR with internal memory scrubbing of the ROM and RAM memory modules, and TMR with both configuration and internal memory scrubbing. The SEU sensitivity of each design variation is compared. The configuration scrubbing only variation is used as a baseline reference. The complementary nature of each SEU mitigation technique is observed as they interact with each other. SEU sensitivity is evaluated on a Xilinx Kintex 7 FPGA through fault injection and radiation testing. Results suggest that combining complementary SEU mitigation techniques offers the greatest reduction in SEU sensitivity.

[1] To processor – transfer done (1 bit), response type (2 bits), read data bus (32 bits); to peripherals – address bus (32 bits), read/write (1 bit), transfer type (2 bits), transfer size (3 bits), burst type (3 bits), write data bus (32 bits).

4.8.2 ITC'99 B13 Benchmark Design

The ITC'99 B13 Benchmarks is part of a suite of twenty-two benchmark designs that are representative of typical circuits that can be synthesized [51]. This set of designs was proposed for use as reliability benchmark designs by [42]. The B13 design is a finite state machine used to control a weather station. Although the circuit is small, previous research has been conducted using this design, which facilitates the comparison of mitigation approaches with other researchers [42].

Because the B13 design is smaller, 512 copies of the B13 are instanced on the target FPGA to accelerate data collection. Having more instances on the FPGA increases the probability of failure, which in turn increases the data collection rate for fault injection and radiation testing. A one-to-many lockstep comparison scheme is combined with on-chip test vectors for the B13 experiments. Outputs of design instances are compared against each other *and* a set of golden test vectors made available on the FPGA. The on-chip test control unit provides the input test vectors to the B13 instances in a continuous loop and resets the instances between iterations. To mitigate the effects of SEU in the input and output vectors, TMR and scrubbing are applied to the respective memory modules.

Several variation of this design are tested on two different FPGA architectures. On a Xilinx Kintex 7 FPGA, three SEU mitigation technique variations are tested: non-TMR, TMR, and TMR with configuration scrubbing. The functional failure error detection logic is protected with TMR. Results are compared against findings in related works and demonstrate the benefits of combining complementary SEU mitigation techniques. On the Altera Stratix V FPGA, the SEU sensitivities of two SEU mitigation variations are estimated using two different functional error detection mechanisms. It was found that a DWC protected functional error detection mechanism provided more favorable results than a TMR protected error detection mechanism for error detection. Results from the non-TMR and TMR with configuration scrubbing variations are compared against those of the Xilinx B13 experiments for a cross-architecture comparison.

A non-TMR version and a TMR version of a very simple FPGA design are also tested using an on-chip one-to-many comparison scheme with on-chip test vectors. This experiment was conducted on an Altera Stratix V FPGA and demonstrates that the amount of SEU reduction observable using on-chip error detection may depend on the resource utilization ratio between the

test fixture logic and the design under test. Fault injection results are compared against the B13 Altera fault injection results.

4.8.3 128-bit AES Cryptography Core

An open source 128-bit AES cryptography core is also used for testing. This is a larger FPGA design that is representative of a real-world use case for FPGAs. The design is fully pipelined and produces a valid output every clock cycle once the pipeline is filled.

The SEU sensitivity of this design was tested with on-chip error detection using a cascaded one-to-one lockstep comparison scheme. Two instances of the AES design are chained together in a feedback loop such that the output of one instance feeds into the input of the other. The outputs of two chains are compared against each other. The test control unit fills the pipelines of both chains with identical input vectors before feeding their outputs back into their respective inputs to create a feedback loop. This approach fills the FPGA and keeps the design highly active, which facilitates the detection of functional errors.

Like the B13 design, the SEU sensitivity of a non-TMR and a TMR with configuration scrubbing variations of this design are tested across two FPGA architectures. A TMR variation without configuration scrubbing is tested through fault injection on a Xilinx Kintex 7 FPGA. Also on the Kintex 7 FPGA, variations where the on-chip functional error detection mechanism is protected with TMR *and* DWC (first triplicated and then duplicated, such that a discrepancy must be detected by the majority output of both triplicated mechanisms in order to be considered a failure) are compared against results that just use DWC to protect the functional error detection mechanism from SEUs. On the Altera Stratix V, non-TMR and TMR with configuration scrubbing results are compared using DWC protected and TMR protected versions of the functional error detection mechanism. Results suggest that using DWC alone for error detection is the best approach for protecting functional error detection mechanisms from SEUs.

4.9 SEU Estimation Through Fault Injection and Radiation Testing

The design, comparison scheme, and functional error detection mechanism variations described in Section 4.8 are tested through either fault injection, radiation testing, or both. In the

chapters that follow, the approach and metrics used to collect data for fault injection and radiation testing are discussed. Collected data is also presented and compared with other results as appropriate. Outcomes of the experiments as related to the original goals are discussed.

CHAPTER 5. FAULT INJECTION

Fault injection is the process of artificially introducing a fault and observing the response. The type of fault injection referred to in this thesis involves mimicking the behavior of an SEU in configuration memory by altering configuration memory. Fault injection can be used to estimate the SEU sensitivity of an FPGA design and to validate SEU mitigation techniques. While it has limitations and does not replace the need for accelerated radiation testing, it does provide a relatively inexpensive and rapid means of estimating SEU sensitivity and observing the potential effects of SEUs [15].

There are many ways that fault injection can be done to emulate SEUs. In general, the values stored in the SRAM-cells of the FPGA, which determine its configuration and the state of the operating design, must be altered. One approach it to use partial reconfiguration. Partial reconfiguration modifies the values stored in the SRAM-cells while the FPGA is active; the design continues to run during this process. Using partial reconfiguration, an inverted bit value may be written to the device. Partial reconfiguration can be done internally with fault injection logic on the FPGA or can be done externally. Another method of performing fault injection is corrupting the bitstream used to program the device by inverting a value of a bit within the bitstream.

In this chapter, the approach and metrics used for fault injection in the experiments of this thesis are presented. Then results from all of the different experiments are presented. Appropriate comparisons are made and some conclusions are drawn.

5.1 Approach

Several different fault injection campaigns can be used to gather data from fault injection. The three main types are: exhaustive, targeted, and random. An exhaustive fault injection campaign injects a fault into every single configuration bit possible and tests each bit for sensitivity. A targeted fault injection campaign is selective in the bits it injects and tests. Examples of targeted

fault injection campaigns include only injecting faults into user flip-flops, or only injecting faults into the essential or critical bits as reported by the CAD tools. Random fault injection has an equal likelihood of upsetting any bit within the FPGA's configuration memory. This campaign is typically used to emulate radiation testing.

Fault injection tests consist of injecting faults and checking if a failure has occurred. To emulate the behavior that would occur within a radiation test, a *random* fault injection campaign was conducted to test all benchmark designs. This means that the configuration bits are selected at random for fault insertion; much like the upsets that occur in a well constructed radiation test. The primary goal of a fault injection experiment is to determine the *sensitive* CRAM bits within the FPGA that cause the benchmark design to fail when upset. After a randomly selected CRAM bit is upset, the behavior of the benchmark design is carefully monitored to see if any copy of the benchmark design deviates from the behavior of the other copy or copies of the design that also reside on the FPGA. If a deviation is detected, the CRAM bit is classified as *sensitive*; otherwise, the CRAM bit is classified as *non-sensitive*. A large number of CRAM bits must be upset and categorized to obtain sufficient statistical confidence.

Traditionally every configuration bit in a design would be tested; however, with newer FPGA devices, exhaustive testing of configuration bits becomes less feasible. The experiments conducted in this thesis all use a random fault injection campaign. This campaign follows the flow shown in Figure 5.1. Injected faults and detected failures are recorded and used for analysis. This test procedure follows the methodology of [52]. Since some of the variations of SEU mitigation techniques tested by fault injection do not include CRAM scrubbing, an additional check is made to see if CRAM scrubbing should be emulated. If scrubbing is enabled, the last injected fault is repaired before injecting another fault. Otherwise, faults are allowed to accumulate, emulating SEU behavior without CRAM scrubbing.

Compared to radiation testing, fault injection is relatively inexpensive and is able to collect data at relatively high-speeds. Figure 5.2 displays a fault injection platform used for the experiments conducted on a Xilinx Kintex 7 FPGA. This platform consists of a custom high-speed JTAG configuration manager, which injects faults via partial reconfiguration over JTAG. This platform can conveniently run in a local lab with little overhead. For the Altera FPGA designs, fault injec-

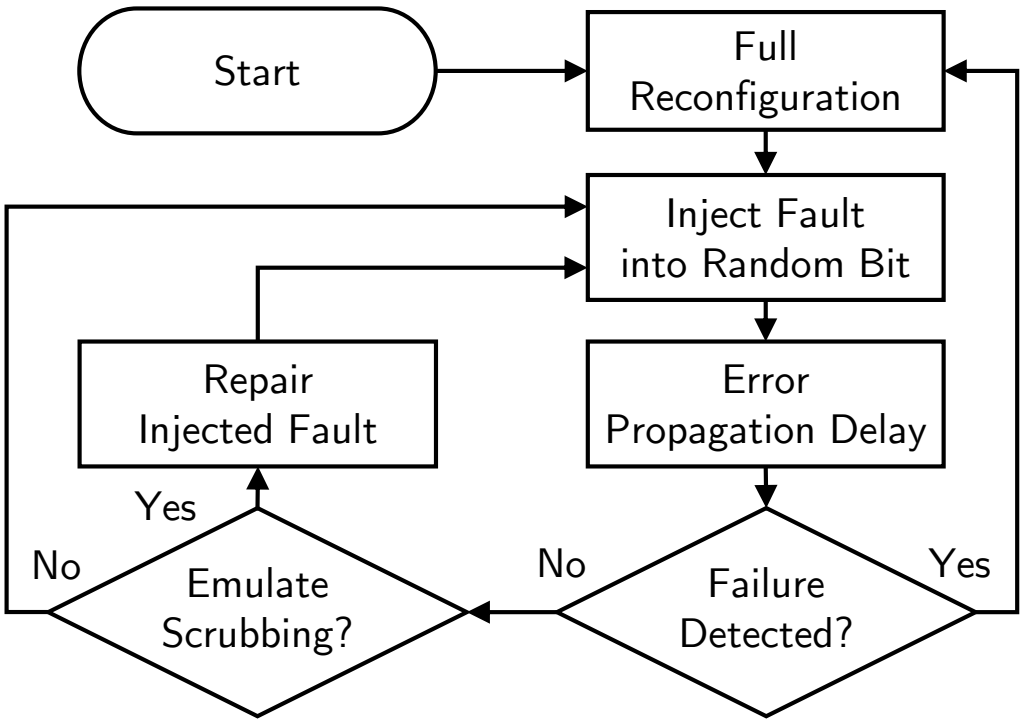


Figure 5.1: Fault Injection Flowchart

tion was performed using Altera's fault injection debugger and IP core instances [53]. Once the test is running, large amounts of data are collected without user intervention.

Only a portion of the configuration bits can be inverted via partial reconfiguration in Xilinx FPGAs and fault injection in Altera FPGAs. BRAM data, global configuration bits, and other bits on an active design can only be upset through radiation testing. Other SEEs, such as transient currents, can only be caused by high-energy radiation. However, fault injection testing accelerates radiation testing by granting insight into what to expect from radiation testing. Radiation tests are expensive; fault injection allows users to better prepare their designs and test methodologies for radiation testing.

Fault injection has its limitations and does not model all of the negative behavior of FPGAs operating in the presence of ionizing radiation (e.g., upsets in the proprietary internal state of an FPGA, user flip-flops, and block memories). Injected faults do not have the same characteristics that may occur within a radiation test beam or in actual space orbit (i.e., multi-cell upsets and random inter arrival times). As such, some of the failure modes that are seen in radiation testing will

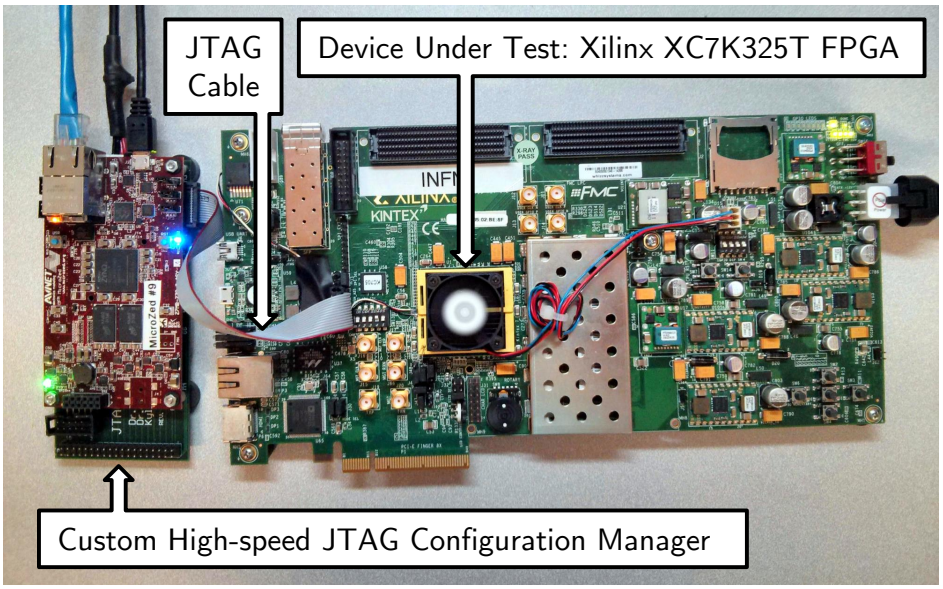


Figure 5.2: A custom high-speed JTAG controller (left) is used to perform automated fault injection testing on a Xilinx Kintex 7 FPGA (right).

not appear in fault injection. Improvement in reliability, between the unmitigated and mitigated design, seen by fault injection is expected to be higher than the improvement seen in radiation testing. In spite of these limitations, fault injection is a very helpful tool that provides important, preliminary information on the effectiveness of a given mitigation scheme, or the influence of FPGA architecture and test fixture configuration on SEU sensitivity.

The goals of fault injection for this work are: first, to estimate the configuration sensitivity of several SEU mitigation approach on different benchmark designs; second, explore the influence of protecting functional error detection logic with TMR and DWC; and finally, compare the SEU sensitivities of several design variations on two different FPGA architectures.

5.2 Metrics

The primary metric used in the fault injection experiments is design *sensitivity* or the percentage of CRAM bits that cause a design failure when upset. This design sensitivity metric is very similar to the design "cross section" metric that will be used for the neutron radiation test results. Rather than testing each CRAM bit, the sensitivity of the design is *estimated* [54] by dividing the number of observed failures (k) by the number of faults injected (n). This is equivalent to the

maximum likelihood estimator, $\hat{r}$, of the binomial distribution:

$$\hat{r} = \frac{k}{n}.$$

A related metric is the Mean Upsets To failure (MUTF) or the inverse of the design sensitivity (i.e., n/k).

The standard deviation of the maximum likelihood estimator is used to determine the 95% confidence interval between the sensitivity of the random sample and that of the design as a whole. The standard deviation is estimated using the following equation:

$$\sigma = \sqrt{\frac{k}{n^2}\left(1 - \frac{k}{n}\right)}. \tag{5.1}$$

The *percent error* is calculated by dividing the standard deviation by the maximum likelihood estimator. This value represents how tight the confidence interval is around the estimated sensitivity. Designs with lower sensitivity (i.e., fewer faults) require much more fault injection to get good results.

The Kintex 7 FPGA used in these experiments (XC7K325T) has approximately 25% fewer CRAM bits than the Altera Stratix V FPGA (5SGXEA7) used in these experiments. To provide a one-to-one comparison metric, the estimated number of sensitive bits in each design is used, with corresponding 95% confidence intervals. This value is the product of the sensitivity of the design and the total number of CRAM bits in the FPGA found in FPGA (72,823,424 for the Xilinx Kintex 7 FPGA; 98,681,196 for the Altera Stratix V FPGA). This metric helps assist in a fair comparison of the two FPGAs.

5.3 LEON3 Xilinx Results

Five variations of SEU mitigation techniques, applied to the LEON3 processor, are tested: configuration scrubbing, TMR, TMR with configuration scrubbing, TMR with internal memory scrubbing of the RAM and ROM modules, and TMR with both configuration and internal memory scrubbing. The estimated design sensitivity of each of the five LEON3 design variations is shown in Table 5.1. This is the SEU sensitivity for all of the logic on the target FPGA, which includes two

processors and the rest of the on-chip test fixture. Because the test fixture logic is also included in the estimated sensitivity, the SEU sensitivity of a design can only be overestimated. The same is true for the cross section of design obtained in neutron radiation testing. A rough SEU sensitivity estimation of a single design instance be determined by dividing the total SEU sensitivity of the test fixture and design instances by the number of design instances, two in this case.

Table 5.1: LEON3 Xilinx Fault Injection Results

Variation	1	2	3	4	5
Description	CRAM Scrubbing Baseline	TMR No Scrubbing	TMR & BRAM Scrubbing	TMR & CRAM Scrubbing	TMR, BRAM & CRAM Scrubbing
Faults Injected (n)	1,831,859	1,369,445	1,502,340	8,840,565	29,443,885
Observed Failures (k)	6,501	1,200	1,100	1,150	2,037
MUTF	282	1,141	1,366	7,687	14,455
Sensitivity	.355%	.0876%	.0732%	.0130%	.00692%
Percent Error	1.24%	2.89%	3.01%	2.95%	2.22%
(95% Conf. Interval)	(.346%, .363%)	(.0827%, .0926%)	(.0689%, .0775%)	(.0123%, .0138%)	(.00662%, .00722%)
Est. Sensitive Bits	258,440	63,813	53,321	9,473	5,038
(95% Conf. Interval)	(252,168; 264,711)	(60,204; 67,422)	(50,171; 56,471)	(8,926; 10,021)	(4,819; 5,257)
Improvement	1.00×	4.05×	4.85×	27.28×	51.30×
(95% Conf. Interval)	(.95×; 1.05×)	(3.74×; 4.4×)	(4.47×; 5.28×)	(25.17×; 29.66×)	(47.97×; 54.93×)

The configuration scrubbing variation is the baseline reference LEON3 processor that provides no spatial SEU mitigation techniques. Without any additional internal mitigation hardware, this design is the smallest of the different design variations and is expected to be the most sensitive to SEUs. It is expected that this design will be the most sensitive to fault injection and ionizing radiation and that all other design variations will provide greater SEU immunity. Although no structural mitigation was provided in this design, CRAM scrubbing was performed during the fault injection and radiation testing. This scrubbing was performed to protect the comparison and reduction error detection logic used by the functional error detection mechanism of the test fixture[1]. All other design variations are compared to the SEU response of this baseline unmitigated design.

The improvement in design SEU sensitivity, over the baseline design, is also shown in Table 5.1. This is calculated by dividing the sensitivity of the baseline design by the sensitivity of the design variation. All four LEON3 mitigated design variations demonstrate an improvement over the unmitigated baseline design. The TMR only variation demonstrates a 4× improvement over

[1]In hindsight, it would have been interesting to include a non-TMR version without CRAM scrubbing in the experiment.

the baseline – its improvement is limited since it allows data corruption to accumulate within the internal memories. Adding internal BRAM scrubbing (#3) further increases the improvement by preventing BRAM error accumulation. Although errors are not artificially injected into the BRAM, errors can enter the BRAM through errors in the surrounding logic. The use of CRAM scrubbing is very important as it prevents the accumulation of upsets within the configuration memory and facilitates the proper operation of TMR. The combination of all SEU mitigation techniques provides over 50× improvement in design sensitivity over the unmitigated baseline design.

These results, collected using an on-chip error detection approach, suggest that the mitigated LEON3 design is 51.3× *less* sensitive to upsets than the baseline design in spite of the fact that the mitigated design is 3.4× larger in resource utilization over the unmitigated design. These results indicate that the mitigation techniques described in Section 2.5 significantly reduce the sensitivity of the LEON3 processor to upsets in the configuration memory.

In spite of applying SEU mitigation techniques, however, there are a number of configuration bits in the mitigated design that are still sensitive to single-event upsets. This suggests that the design still contains a number of single points of failure, either in the test fixture logic, or in the designs themselves. Future work will investigate the cause of these remaining single points of failure to further improve the accuracy of on-chip error detection and the reliability of the design under test.

5.4 B13 Altera Results

The B13 benchmark design was tested on an Altera Stratix V FPGA using two different methods of protecting the on-chip functional error detection logic. The comparison and reduction logic is either triplicated or duplicated. Triplicating the logic allows the correct answer to be voted on but costs more area; duplication consumes fewer resources, but there is no way of telling which redundant copy is correct when the two instances disagree with each other. The SEU sensitivity of 512 instances of the B13 plus the on-chip test fixture logic is estimated using the two different methods of protecting the on-chip functional error detection logic. Table 5.2 show the results from testing an unmitigated and mitigated design variations using TMR protected error detection and DWC protected error detection.

Table 5.2: B13 Altera Fault Injection Results

Description	Unmitigated TMR Detection	TMR & Scrubbing TMR Detection	Unmitigated DWC Detection	TMR & Scrubbing DWC Detection
Faults Injected (n)	12,210	31,083	19,493	19,295
Observed Failures (k)	265	86	285	24
MUTF	46	361	68	804
Sensitivity Percent Error (95% Conf. Interval)	2.1704% 6.08% (1.912%, 2.429%)	.2767% 10.77% (.0827%, .0926%)	1.462% 5.88% (1.294%, 1.631%)	.1244% 20.40% (.0747%, .1741%)
Est. Sensitive Bits (95% Conf. Interval)	2,141,729 (1,886,675; 2,396,784)	273,030 (215,404; 330,655)	1,442,782 (50,171; 56,471)	122,744 (8,926; 10,021)
Improvement (95% Conf. Interval)	1.00× (.79×; 1.27×)	7.84× (5.71×; 11.13×)	1.00× (.79×; 1.26×)	11.75× (7.43×; 21.84×)

The results suggest that protecting the functional error detection logic with DWC provides more favorable results than using TMR to protect functional error detection logic for detecting errors. The unmitigated design is more sensitive when tested using TMR protected error detection than when using DWC protected error detection. It is possible that the TMR protected error detection logic returns more false positives than the DWC protected logic, thus increasing the sensitivity of the unmitigated and mitigated versions of the design. The overall improvement found using TMR to protect the error detection logic is 7.84×; whereas using DWC to protect the error detection logic demonstrates an 11.75× improvement in reliability. Based on these results, it appears that using DWC to protect error detection logic introduces fewer false-positives into the results. The results provided by DWC are more favorable and may be more accurate, but without some form of off-chip error validation (i.e., failure-immune validation) it cannot definitively be said that DWC provides more accurate results than TMR for error detection.

5.4.1 Comparison of a Simple DUT and the B13

To study the impact of size of the test fixture logic verses size of the design under test, the SEU sensitivity of a much smaller design under test was estimated using the same on-chip error detection test fixture as the B13 design. TMR was used to protect the error detection logic for this experiment. 2048 instances of simple design consisting of ten six-input look-up tables feeding 10 registers was tested. Random test vectors were used as input stimulus with corresponding

golden output vectors. This design is considerably smaller than the B13 finite state machine. This experiment is expected to demonstrate the negative influence of the test fixture overhead on obtained results. Table 5.3 shows the results obtained through fault injection on an Altera Stratix V FPGA.

Table 5.3: Simple DUT Altera Fault Injection Results

Description	Unmitigated TMR Detection	TMR & Scrubbing TMR Detection
Faults Injected (n)	658	11,345
Observed Failures (k)	30	100
MUTF	22	113
Sensitivity Percent Error (95% Conf. Interval)	4.56% 17.84% (2.97%, 6.15%)	.88 % 9.96% (.71%, 1.05%)
Est. Sensitive Bits (95% Conf. Interval)	4,499,143 (2,926,275; 6,072,010)	869,821 (700,089; 1,039,553)
Improvement (95% Conf. Interval)	1.00× (.48×; 2.07×)	5.17× (2.81×; 8.67×)

To provide a fair comparison between the B13 and the simple DUT designs, the results from the simple DUT experiment are scaled so that the sensitivity of 512 instances of each design are compared against each other instead of 2048 instances of the simple DUT against 512 of the B13. This is done by multiplying the MUTF of the simple DUT by 4 and dividing its sensitivity by 4. Table 5.4 shows a scaled comparison of the simple DUT and B13 fault injection data using TMR protected functional error detection logic.

The results obtained from on-chip error detection find that a simple DUT instance is less sensitive than a B13 instance for both the unmitigated and mitigated versions respectively. This is determined by comparing the scaled sensitivity of both designs and noting that in both the unmitigated and mitigated cases the sensitivity is lower for the simple DUT design. Considering that the simple DUT design is much smaller than the B13 design, this outcome is to be expected. The more interesting outcome is found by comparing the improvement between the unmitigated and mitigated version of each design. The simple DUT design shows an improvement of 5.17x whereas the B13 design shows an improvement of 7.84x (using TMR protected error detection

Table 5.4: Normalized Simple DUT and B13 Comparison Fault Injection Results

Description	Simple DUT Unmitigated	Simple DUT TMR & Scrubbing	B13 Unmitigated	B13 TMR & Scrubbing
MUTF	88	452	46	361
Sensitivity Percent Error	1.140% 17.84%	.2204% 9.96%	2.1704% 6.08%	.2767% 10.77%
Est. Sensitive Bits	1,124,786	217,455	2,141,729	273,030
Improvement	1.00×	5.17×	1.00×	7.84×

logic). These results suggest that as the sensitivity of a design decreases in relation to the test fixture logic overhead, the amount of improvement that can be demonstrated declines. In other words, there is a shared likelihood of false positives occurring from SEUs in the test fixture logic; as the sensitivity of the design decreases, the likelihood of false-positives begins to dominate, which makes it harder to see a difference between the sensitivity of the unmitigated and mitigated version of a design.

The error bounds in this experiment overlap in part because the improvement between the two designs are close together. The fact that the improvement from SEU mitigation is close in both designs suggests that the overhead of false-positive occurrences is significant. In other words, a significant portion of the estimated sensitivity in this experiment is due to SEUs in the error detection logic that resulted in false-positive failure reports. Further experimentation and off-chip error validation would strengthen these conclusions.

5.5 B13 Xilinx Results

On a Xilinx Kintex 7 FPGA, 512 instances of the B13 design were compared against each other and a set of golden output vectors using TMR protected functional error detection logic. Three variations of SEU mitigation were applied to the designs for testing. The "Unmitigated" version does not have TMR applied to the B13 design instances, but it does have configuration scrubbing applied to the prevent SEU accumulation in the on-chip error detection test fixture logic. Again, this version is expected to be the most sensitive version and is use as a baseline reference for the other variations of the design. In the second version, only TMR is applied to the design.

The third version combines TMR and configuration scrubbing. Fault injection was performed on this design to demonstrate the benefits of combining TMR and scrubbing, show what happens to TMR benefits with SEU accumulation (i.e., without configuration scrubbing), and to later be used in a cross-architecture comparison of SEU sensitivity on an Altera Stratix V FPGA and a Xilinx Kintex-7 FPGA. Result from the random fault injection campaign are shown in Figure 5.5.

Table 5.5: B13 Xilinx Fault Injection Results

Description	Unmitigated	TMR Only	TMR & Scrubbing
Faults Injected (n)	558,012	665,461	58,237,618
Observed Failures (k)	8,600	2,791	8,533
MUTF	65	235	6,825
Sensitivity Percent Error (95% Conf. Interval)	1.54% 1.07% (1.51%,1.57%)	0.45% 1.89% (.41%, .44%)	.0147% 1.08% (.0143%,.0150%)
Est. Sensitive Bits (95% Conf. Interval)	1,122,344 (1,098,807; 1,145,882)	310,087 (298,608; 321,567)	10,670 (10,444; 10,897)
Improvement (95% Conf. Interval)	1.0× (.96×; 1.04×)	3.62× (3.42×; 3.84×)	105.2× (100.84×; 109.72×)

Results in Table 5.5 demonstrate a 105.2× improvement in reliability between the unmitigated and the TMR with configuration scrubbing design. TMR alone only demonstrates a 3.6× reduction in SEU sensitivity. Without configuration scrubbing, the benefits of TMR drastically decline. It is interesting to note that a number of sensitive bits (i.e., bits that if upset will cause the design to fail), are still present in the design. Future work will investigate why these bits are sensitive and how to eliminate them.

5.6 AES Altera Results

Much like the fault injection test of the B13 design on an Altera Stratix V FPGA, the AES fault injection experiments on the Altera Stratix V FPGA compares two different approaches for protection of functional error detection logic. The first approach protects the detection with logic with TMR, but the triplicated clock network was optimized away by the CAD tools into a single clock network. The second approach uses DWC to protect the error detection logic, and the

clock network was forced to be triplicated. This experiment compares the effects of using TMR over DWC for protecting error detection logic, and demonstrates the importance of triplicating the clock network for TMR. Results from this experiment are shown in Figure 5.6.

Table 5.6: AES Altera Fault Injection Results

Description	Unmitigated TMR Detection Single Clock	TMR & Scrubbing TMR Detection Single Clock	Unmitigated DWC Detection Triplicated Clock	TMR & Scrubbing DWC Detection Triplicated Clock
Faults Injected (n)	3,903	3,987	14,544	124,045
Observed Failures (k)	400	177	1,451	17
MUTF	10	23	10	7,297
Sensitivity Percent Error (95% Conf. Interval)	10.249% 4.47% (9.297%, 11.200%)	4.439% 7.35% (3.800%, 5.079%)	9.977% 2.49% (9.490%, 10.463%)	.0137% 24.25% (.0072%, .0202%)
Est. Sensitive Bits (95% Conf. Interval)	10,113,369 (9,174,418; 11,052,319)	4,380,881 (3,749,967; 5,011,795)	9,845,051 (9,364,413; 10,325,688)	13,524 (7,096;19,952)
Improvement (95% Conf. Interval)	1.00× (.83×; 1.20×)	2.31× (1.83×; 2.95×)	1.00× (.91×; 1.10×)	727.97× (469.34×; 1,455.24×)

It is interesting to note that in this design there is not much difference between the sensitivity of the unmitigated design estimated by TMR protected error detection logic with a single clock and estimated by DWC protected logic with a triplicated clock. Both approaches returned a MUTF of approximately 10. For the B13 design, 512 sets of 10-bit signals are compared against each other and a set of golden inputs and the result is reduced to a single bit. That requires a 51,210 input reduction tree network. In the AES design, the functional error detection logic compares two sets of 128-bit signals, which requires only a 256 input reduction tree network. The reduction tree network needed by the AES design is much smaller than that of the B13 design, and thus the prevalence of false-positive failures is lessened [33]. Based on these considerations and the fact that each AES instance has much higher resource utilization than the on-chip error detection test fixture, it is assumed that false-positive overhead in the results of the AES design is very small.

By triplicating the clock network and using DWC to protect the error detection logic from reporting false-positives, the improvement benefit gained from applying TMR and configuration scrubbing increased from 2.31× to 728×. This is an enormous leap in improvement. These results suggest that by triplicating the clock networks, many single point failures are eliminated in the triplicated AES design logic. Using DWC over TMR to protect the functional error detection logic

contributed to this improvement increase in the results; although, it is difficult to say by how much. The most important outcome from this comparison is that triplicating the clock network reduces single point failures in the design, and that DWC protection of functional error detection logic provides more favorable results.

Much greater improvement is seen in the AES design than in the B13 design when TMR and configuration scrubbing are applied to the designs. There are many factors that influence the SEU sensitivity of a design including: synthesis, placement, and routing. One possible cause of the difference is the number of times all three TMR domains are forced to pass through the same logic function. In the B13 design, there are approximately 87,000 majority voters in the final on-chip test fixture circuit with the TMR'd design instances. In the AES design, there are approximately 768 majority voters in the final on-chip test fixture circuit with the TMR'd design instances. Each majority voter is a logic function that all three TMR domains must pass through, which presents a potential vulnerability for single points failures [32]. These results suggest that there are more single point failures in the mitigated B13 design than there are in the mitigated AES design on this particular FPGA architecture.

5.7 AES Xilinx Results

Like the B13 design on an Altera Stratix V FPGA, the AES design is tested on a Xilinx Kintex 7 FPGA using two variations of SEU mitigation for the on-chip functional error detection logic. The first variation is an aggressive form of SEU mitigation that consists of both TMR and DWC. In this variation, the functional error detection logic is first triplicated and then duplicated. Figure 5.3 depicts a TMR and DWC protected error detection logic. The outputs of design instances A and B are fed through a comparison and reduction network that is triplicated and duplicated. An error is only detected when two or more of the majority voter outputs (C_0, C_1, C_2; and D_0, D_1, D_2 respectively) of both copies of the triplicated comparison and reduction network agree that an error has occurred, at the same time. The second variation is a DWC protected functional error detection logic.

The original intent was that using an aggressive SEU mitigation of the functional error detection logic would reduce the likelihood of false-positives and improve the accuracy of on-chip error detection. This seemed plausible since the encoding of errors is stronger with this approach

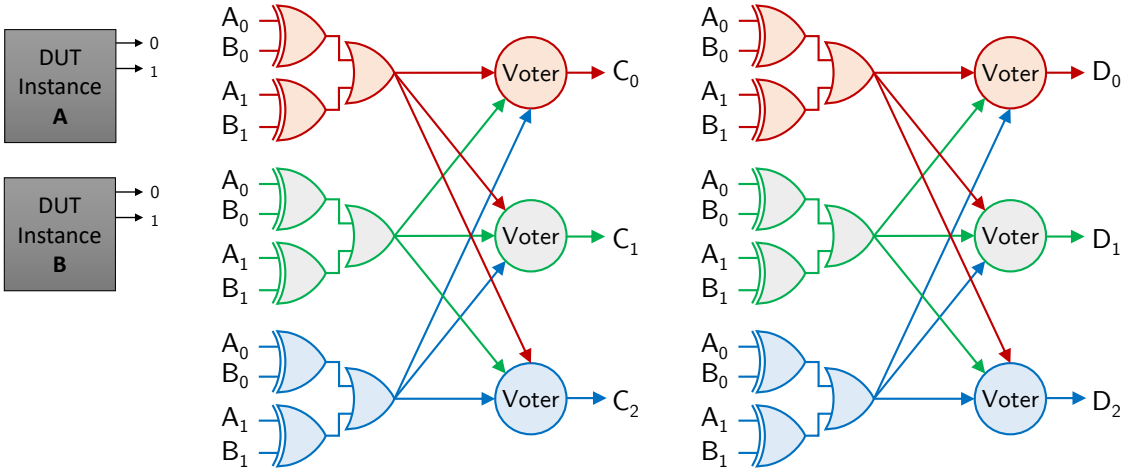


Figure 5.3: TMR and DWC Protected Function Error Detection Mechanism

(i.e., errors have to be detected by two triplicated functional error detection circuits). The estimated SEU sensitivity of the unmitigated and mitigated version of the AES design are shown in Table 5.7 and compared with findings gathered using DWC protected error detection logic.

Table 5.7: AES Xilinx Fault Injection Results

Description	Unmitigated TMR/DWC Detection	TMR & Scrubbing TMR/DWC Detection	Unmitigated DWC Detection	TMR DWC Detection	TMR & Scrubbing DWC Detection
Faults Injected (n)	832,113	1,493,221	278,287	974,499	5,312,513
Observed Failures (k)	10,000	1,779	19,233	23,971	5,024
MUTF	83	839	14	41	1,057
Sensitivity Percent Error (95% Conf. Interval)	1.20% .99% (1.18%, 1.22%)	.119% 2.37% (.114%, .125%)	6.9% .70% (6.8%, 7.0%)	2.46% .64% (2.43%, 2.49%)	.095% 1.41% (.092%, .097%)
Est. Sensitive Bits (95% Conf. Interval)	874,619 (857,580; 891,658)	86,707 (82,680; 90,734)	5,032,980 (4,964,351; 5,101,609)	1,791,331 (1,768,935; 1,813,728)	68,869 (66,965; 70,772)
Improvement (95% Conf. Interval)	1.00× (.96×; 1.04×)	10.09× (9.45×; 10.78×)	1.00× (.97×; 1.03×)	2.8× (2.7×; 2.9×)	73.1× (70.2×; 76.2×)

The data only demonstrates a 10× improvement between the unmitigated and mitigated design versions using TMR and DWC to protect the functional error detection logic; whereas the DWC protected functional error detection logic demonstrates a 73× reduction in SEU sensitivity between the unmitigated and mitigated design versions. An interesting thing to note is that the sensitivity of the unmitigated design reported using the TMR and DWC error detection protection

is much lower than that of the unmitigated version as reported by DWC protected error detection logic. It is not clear what causes these differences. It is possible that TMR and DWC protected logic misses error events (i.e., false-negatives) or better masks errors in the functional error detection logic, but without off-chip error validation, it is difficult to confirm.

The DWC protected functional error detection circuitry results are more in line with the Xilinx Kintex 7 results on the LEON3 and B13 circuit, and they are more favorable than the results obtained from the TMR and DWC protected error detection circuitry. For these reasons, the DWC protected results will be used for comparison with Altera Stratix V results. It is also a fitting comparison because the Altera Stratix V test also uses DWC to protect the functional error detection circuitry of its experiment.

Another interesting outcome of the AES experiments on the Xilinx Kintex 7 FPGA are the results of the TMR only version of the AES design using DWC protected error detection (see Table 5.7). Like the B13 TMR only design, much less improvement is gained by applying TMR alone than by applying TMR with configuration scrubbing. TMR alone demonstrates a 2.8× improvement; whereas TMR with configuration scrubbing demonstrates a 73× improvement over the baseline unmitigated version. This is the type of behavior that can be expected when SEUs are allowed to accumulate or occur faster than the scrubbing method can keep up with.

5.8 Comparison of Fault Injection Results on Two FPGA Architectures

Data from these experiments can be used to compare the SEU sensitivity of designs implemented on an Altera Stratix V (5SGXEA7)FPGA and a Xilinx Kintex 7 (XC7K325T)FPGA. These devices offer similar feature sets and are both built on 28nm process technology. Table 5.8 summarizes the fault injection results of the B13 and AES designs across both FPGA architectures.

Fault injection results demonstrate some surprising similarities between the two FPGA architectures. First, unmitigated versions of the tested designs have nearly the same number of estimated sensitive bits. Estimated sensitive bits are proportionally scaled to the number of CRAM bits on the device, which provides a metric for a one-to-one comparison of fault injection results across FPGA architectures. The B13 design has 1.4 million and 1.1 million estimated sensitive bits on the Stratix V and Kintex 7 respectively. Estimates are within 2× of each other for the AES design. Considering all of the variables involved, it is impressive that two FPGA architectures with

Table 5.8: Comparison of Fault Injection Data

Design		ITC'99 Benchmark B13		128-bit AES IP Core	
Description		Unmitigated	Mitigated	Unmitigated	Mitigated
Altera Stratix V	Percent Error	5.88%	20.40%	2.49%	24.25%
	MUTF	68	804	10	7,297
	Est. Sensitive Bits	1,442,782	122,744	9,845,051	13,524
	Improvement	1.00×	11.75×	1.00×	728×
Xilinx Kintex 7	Percent Error	1.07%	.015%	0.70%	1.41%
	MUTF	65	6,825	14	1,057
	Est. Sensitive Bits	1,122,344	10,670	5,032,980	68,869
	Improvement	1.00×	105.2×	1.00×	73.1×

separate CAD tools for synthesis, placement, and routing would implement FPGA designs with comparable SEU sensitivity. Second, with TMR and scrubbing applied to the designs, each FPGA architecture favored one design over the other for reduction of SEU sensitivity. Significantly better improvement and fewer estimated sensitive bits were found for the AES design on the Stratix V and for the B13 design on the Kintex 7 FPGA. This suggests that benefits from SEU mitigation are design and architecture dependent.

TMR and configuration scrubbing reduced SEU sensitivity in all cases. On the Altera Stratix V, the B13 design demonstrated an approximate 12× reduction in sensitivity and the AES design demonstrated an approximate 728× reduction in sensitivity. The Kintex 7 demonstrated a 117× and a 73× approximate sensitivity reduction for the B13 and AES designs respectively. There are still a number of estimated sensitive bits for each mitigated design and device suggesting that further improvement can be made in the mitigation approach.

Fault injection is relatively inexpensive and can be conducted very quickly in comparison to accelerated radiation beam testing, but fault injection is not a replacement for radiation testing [15]. Fault injection was used in these experiments to gather preliminary data and to prepare experiments for radiation testing. While fault injection provides useful information, radiation testing provides more accurate results and is the commonly accepted standard for estimating design sensitivity to SEUs [15]. The test fixtures that demonstrated the most favorable results are further tested with radiation testing. Where comparison was possible, using DWC to protect functional error detection logic provided the most favorable results and is thus selected for radiation testing over

TMR protected functional error detection logic where possible. The LEON3 and B13 design on a Xilinx Kintex 7 FPGA are the only design in radiation testing that use TMR protected functional error detection logic, all the others use DWC. In the next chapter, the approach and metrics used for neutron radiation testing as well as obtained results are discussed.

CHAPTER 6. RADIATION TESTING

Although more expensive and difficult to conduct, radiation testing provides a more accurate estimate of the sensitivity of the design to SEUs. This is because, unlike fault injection, radiation testing upsets *all* of the internal memory of the FPGA including BRAM and other user memories. SETs are also observed in radiation testing whereas they do not occur in fault injection [55]. To provide a more complete study, some of the design variation tested through fault injection are also tested with an accelerated neutron radiation beam. Priority is given to the best performing on-chip test fixtures, and the results are analyzed.

Neutron radiation testing was conducted at the Los Alamos Neutron Science Center (LANSCE) in November of 2015 and December of 2016 for the experiments included in this thesis[1]. This facility provides a wide spectrum spallation neutron beam source, which is commonly used to measure and report terrestrial neutron soft-errors in semiconductor devices [4]. All five variations of the LEON3 design and the three variations of the B13 were tested to demonstrate the individual and combined benefits of SEU mitigation techniques. The LEON3 data is compared against fault injection data and the B13 data is compared against findings in [42]. The B13 data is also used in a comparison of SEU sensitivities of designs implemented on different FPGA architectures. The B13 and AES designs are tested on an Altera Stratix V and a Xilinx Kintex 7 for comparison.

6.1 Approach

The neutron radiation experiments were conducted by exposing an active FPGA design to an accelerated neutron radiation beam; the total number of observed failures and the total amount of exposure are recorded for analysis. The development boards that the FPGA designs operate on are mounted normal to a two inch collimated neutron beam. Figure 6.1 shows the board setup used in the December 2016 test. The distance from the neutron source to the FPGA is carefully

[1]LANCSE graciously provided neutron beam time under proposal NS-2016-7268-F. Much gratitude is extended for access to the beam.

measured and compensated for in analysis. Three Xilinx Kintex 7 FPGAs and one Altera Stratix V FPGA were used in the December 2016 test, and three Xilinx Kintex 7 FPGAs were used in the November 2015 test.

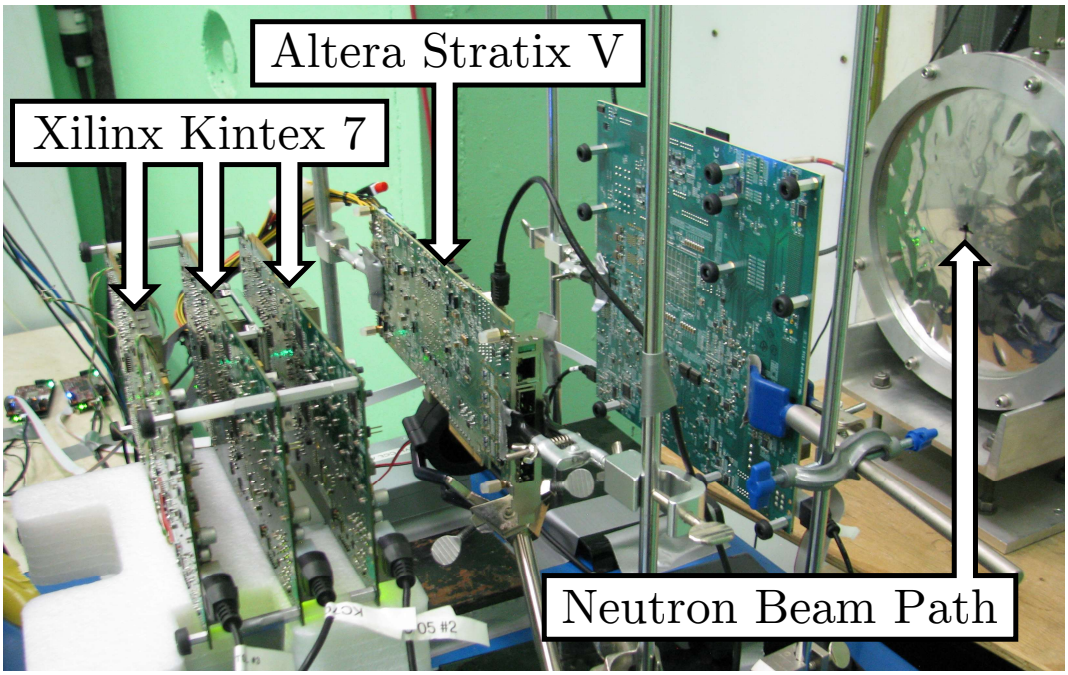


Figure 6.1: Neutron Beam Test Setup

Throughout the duration of a beam test, several test runs are conducted to collect data. A test run begins by first configuring the FPGAs with the design under test. Each design becomes active as soon as it is loaded and does not require any external stimulus (i.e. from outside of the FPGA) to remain active. The beam shutter is then opened. Whenever a functional failure is observed, it is recorded and the FPGA is reconfigured with the design under test. All of the configuration upsets that occur, as reported by configuration scrubbing, are also recorded. A test run ends with the closing of the beam shutter. Automating these tests is important because of the amount of time it takes to collect statistically significant amounts of data. Multiple beam runs are performed as needed to collect sufficient data to make good cross section estimates. A beam run consists of opening the beam shutter, observing failure events, and closing the beam shutter (see Appendix B).

6.2 Metrics

The main metric used to estimate the SEU sensitivity of a design in neutron radiation testing is the *cross section*. The cross section of a design is a hypothetical area that if a particle passes through, would result in an event, in this case a functional failure event [19]. A larger cross section signifies the design is more sensitive. The neutron cross section is estimated by dividing the number of errors observed in the test by the total fluence (neutrons per square centimeter, $\frac{n}{cm^2}$) of the radiation beam,

$$\sigma = \frac{N_{errors}}{Fluence} \tag{6.1}$$

and has units of cm^2. The design cross section measures the cross section of the entire FPGA device running one of the design variations under test, which includes the test fixture and any instances of the design. The cross section captures all failure mechanisms of the design including CRAM upsets, BRAM upsets, upsets into hidden FPGA state, and SEFIs. This cross section measurement will be used to compare all five LEON3 design variations against each other and facilitate the comparison of the neutron radiation tests against the fault injection tests (see Section 6.3). It is also used in the other comparisons made in this chapter. The 95% confidence interval is approximated using the standard deviation of the Normal distribution,

$$\frac{1.96 * \sqrt{Observed\ Failures}}{Total\ Fluence}.$$

Because the sensitive cross section of a design does not depend on the total number of CRAM bits contained in the FPGA, the design cross section can be used as a one-to-one comparison metric across architectures. The improvement shown is a ratio between the cross section of the unmitigated and the mitigated design.

To facilitate the comparison of results obtained during fault injection tests against results obtained during radiation testing, the number of "sensitive bits" is estimated for each of the design variations tested through neutron radiation. The number of "sensitive bits" is estimated by dividing the neutron cross section of the full design by the cross section of a single CRAM bit. The neutron cross section of a CRAM bit for a Xilinx Kintex 7 FPGA was previously measured at LANSCE at 6.99×10^{-15} [56]. For the Altera Stratix V FPGA, the cross section of a single CRAM bit is

estimated using upset data from the experiments in this thesis (See Section 6.5.1). The estimated number of sensitive bits for each design variation is included in the results.

This approach for estimating the number of sensitive bits is pessimistic and will overestimate the actual number of sensitive bits in a design. Any mechanism that causes the LEON3 processor to fail will be attributed to CRAM upsets. Such failure mechanisms may include upsets with BRAM bits, user Flip-Flops, SETs, and upsets within the hidden state of the FPGA. This estimate will manifest these types of failures as "sensitive CRAM bits" rather than their true failure mechanism. In spite of this limitation, this estimate is still useful in that it facilitates the comparison of sensitive bits estimated from both fault injection and radiation testing.

6.3 LEON3 Xilinx Results

The results from the neutron radiation test are shown in Table 6.1. The estimated cross section for each design variation is shown, including the number of failures observed, and the total fluence. The percent error is the standard deviation divided by the cross section and represents how tight the confidence interval is. Because of the limited neutron radiation test time, it was difficult to obtain sufficient testing statistics for all five design variations. In particular, the "TMR No Scrubbing" (#2) and the "TMR and CRAM Scrubbing" (#4) design variations have very wide confidence intervals.

Table 6.1: Neutron Radiation Data

Variation	1	2	3	4	5
Description	Unmitigated	TMR No Scrubbing	TMR & BRAM Scrubbing	TMR & CRAM Scrubbing	TMR, BRAM & CRAM Scrubbing
Failures	35	5	17	9	11
Fluence (n/cm^2)	1.34×10^{10}	1.56×10^{10}	1.06×10^{11}	9.30×10^{10}	2.06×10^{11}
Cross Section (cm^2)	2.61×10^{-9}	3.20×10^{-10}	1.60×10^{-10}	9.68×10^{-11}	5.34×10^{-11}
Percent Error	33.13%	87.65%	47.54%	65.33%	59.10%
(95% Conf. Interval)	(1.74×10^{-9}, 3.47×10^{-9})	(3.95×10^{-11}, 6.00×10^{-10})	(8.41×10^{-11}, 2.36×10^{-10})	(3.36×10^{-11}, 1.60×10^{-10})	(2.18×10^{-11}, 8.49×10^{-11})
Est. Sensitive Bits	380,822	46,691	23,345	14,124	7,792
(95% Conf. Interval)	(249,462; 496,648)	(5,647; 85,825)	(12,028; 33,826)	(4,801; 22,896)	(3,124; 12,149)
Improvement	1.00×	8.16×	16.27×	26.94×	48.85×
(95% Conf. Interval)	(.5×; 1.99×)	(2.91×; 87.96×)	(7.37×; 41.29×)	(10.9×; 103.45×)	(20.53×; 159×)

The estimate of the design cross section for each of the four LEON3 design variations is *less* than the estimated design cross section of the baseline unmitigated design suggesting that these mitigation approaches are successful. The use of successive SEU mitigation techniques provides a correspondingly lower design cross section, with the lowest design cross section obtained from design variation #5, which includes TMR, BRAM scrubbing of the ROM and RAM modules, and CRAM scrubbing. This combination of techniques reduced the estimated design cross section by $49\times$.

The results during radiation testing summarized in Table 6.1 are an improvement over the results obtained during a similar experiment [57]. In this previous experiment, the improvement in SEU sensitivity using all three mitigation techniques was only $10\times$ in spite of fault injection results suggesting $51\times$ improvement. The disparity between radiation testing and neutron testing suggested problems with the experimental setup. After thorough investigation, it was found that the CRAM scrubber was only scrubbing one third of the Kintex-7 325T FPGA. The CRAM scrubbing issue was resolved for these experiments and the fault injection and radiation testing are much closer in alignment.

Comparison to Fault Injection Results

Using the same five designs in a similar testing strategy facilitates comparison of the results between the fault injection experiments and the radiation testing experiments. Table 6.2 summarizes the key results from both sets of experiments for all five LEON3 design variations to facilitate side-by-side comparison (the results are copied from Tables 5.1 and 6.1). Before comparing the specific results from these tests it is important to note that the confidence intervals of the radiation test results are much larger than the confidence intervals of fault injection (as noted by the "percent error"). This disparity is due to the limited time available for radiation testing and slow rate of upsets obtained in radiation testing in comparison to fault injection.

The first method for comparing the two testing methods is to compare the "Improvement" of each of the SEU mitigation techniques. The improvement results indicate how much each design variation reduced the SEU sensitivity over the baseline unmitigated design. The improvement facilitates the *relative* benefit of each mitigation techniques for both fault injection and radiation testing. All SEU mitigation techniques provide improvement and the improvement increases as the

Table 6.2: Comparison of Fault Injection and Neutron Radiation Testing

Variation		1	2	3	4	5
Description		Unmitigated	TMR No Scrubbing	TMR & BRAM Scrubbing	TMR & CRAM Scrubbing	TMR, BRAM & CRAM Scrubbing
Fault Injection	Percent Error	1.24%	2.89%	3.01%	2.95%	2.22%
	Est. Sensitive Bits	258,440	63,813	53,321	9,473	5,038
	Improvement	1.00×	4.05×	4.85×	27.28×	51.30×
Neutron Radiation	Percent Error	33.13%	87.65%	47.54%	65.33%	59.10%
	Est. Sensitive Bits	380,822	46,691	23,345	14,124	7,792
	Improvement	1.00×	8.16×	16.27×	26.94×	48.85×

mitigation techniques are combined. In spite of the differences in the testing methodologies, the improvement for fault injection and radiation testing is surprisingly similar for most of the design variations (#1, #4, and #5). When all techniques are combined, design variation #5 has the highest improvement of roughly 50× for *both* fault injection and radiation testing. The improvement observed in both testing methodologies is plotted in Figure 6.2 including the confidence interval bounds.

TMR only on the unmitigated design (#2) demonstrated an improvement in reliability of 8.16× in radiation testing and 4.05× in fault injection. The greater improvement seen in radiation testing is likely due to the fact that radiation testing upsets more system state than fault injection (such as BRAM and user flip-flops) and TMR protects these additional failure mechanisms that are not tested with fault injection. This effect is more pronounced in the TMR and BRAM scrubbing variation (#3) in which radiation testing demonstrates a 16× improvement while the improvement for fault injection is only 4.9×. This suggests that internal BRAM memory scrubbing plays a significant component in improving the SEU sensitivity of FPGA systems that employ internal BRAM memory.

Configuration scrubbing plays a very important part in reducing SEU design sensitivity as suggested by design variations #4 and #5 that integrate active CRAM scrubbing. This suggests there is a greater chance of failure due to SEU accumulations in configuration memory than due to SEU accumulations in BRAM for this design, or in other words that the majority of bits utilized by the LEON3 design are part of static configuration memory. This would explain why configuration scrubbing has a greater positive impact on the reliability of the TMR design than internal memory scrubbing.

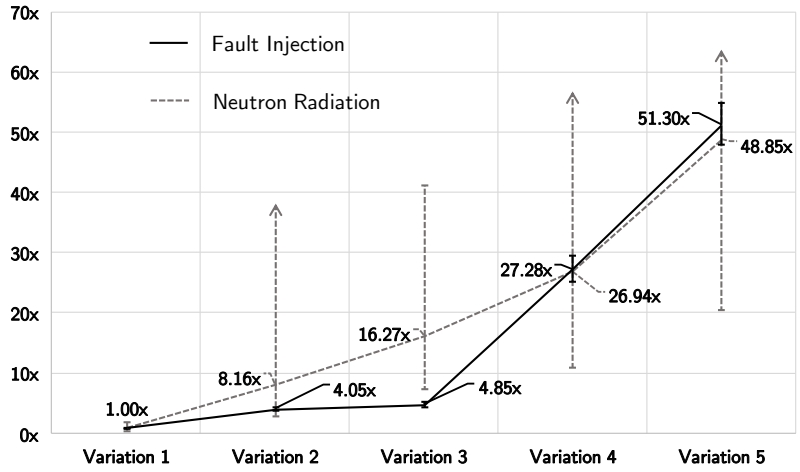


Figure 6.2: SEU Sensitivity Improvement for Each Design Variation

Another method for comparing the results between fault injection and radiation testing is to compare the estimated *sensitive bits* for each design variation. This approach facilitates *absolute* comparison of SEU sensitivity between fault injection and radiation testing. The number of sensitive bits is estimated for fault injection by multiplying the estimated design sensitivity by the total number of configuration bits; this number is estimated in radiation testing by dividing the *design* sensitive cross section by the cross section of a single configuration bit. The estimated number of sensitive bits for all design variations in both testing approaches is summarized in Table 6.2 along with the 95% confidence bounds.

For the unmitigated design (#1), radiation testing estimates a significantly higher number of sensitive bits than with fault injection. This result is expected since more internal state is upset during radiation testing than fault injection (BRAM, FFs, SETs, etc.). These data support the idea that BRAMs are excluded from testing in fault injection but included in radiation testing. These data also show the design as slightly more susceptible failure in the beam than in fault injection which is to be expected. This trend of higher estimated sensitive bits also is seen in design variations #4 and #5 as well[2].

[2]The estimated number of sensitive bits for designs #2 and #3 is lower for radiation testing than for fault injection. For reasons that are not fully understood, CRAM scrubbing has a greater effect during radiation testing than fault injection.

6.4 B13 Xilinx Results

Three variations of the B13 design were tested through radiation testing: configuration scrubbing, TMR, and TMR with configuration scrubbing. Like the LEON3 experiment, this experiment also shows in the individual and combined benefits of SEU mitigation techniques. One difference between the two experiments is that the B13 experiment uses a one-to-many lockstep on-chip functional error detection logic with on-chip golden test vectors whereas the LEON3 experiment uses a one-to-one lockstep comparison logic. The difference in SEU sensitivity between the SEU mitigation techniques can be seen using both methods. The results from neutron radiation testing are presented in Table 6.3.

Table 6.3: B13 Neutron Radiation Results

	Unmitigated	Mitigated W/O Scrubbing	Mitigated W/ Scrubbing
Failures	120	108	26
Fluence (n/cm^2)	1.30E+10	6.01E+10	1.47E+11
Fluence To Failure	1.08E+08	5.56E+08	5.66E+09
Cross Section (cm^2) (confidence int.)	9.26E-9 (7.6E-9,1.1E-8)	1.80E-9 (1.5E-9,2.1E-9)	1.77E-10 (1.1E-10,2.5E-10)
Improvement	1.0	5.1	52.4

It is not surprising that the benefits of TMR without scrubbing are significantly less than TMR with scrubbing as analytical models suggest that TMR without repair (scrubbing in this case) has a lower MTTF than single, non-TMR systems ($\frac{5}{6\lambda}$ v.s $\frac{1}{\lambda}$). The fact that TMR without scrubbing exhibits an improvement over the non-TMR system is most likely due to the fact that fine-grain feedback TMR is used rather than coarse-grain TMR with a single set of global output voters. More curious still is that the improvement in reliability for the TMR design without scrubbing is higher than what was observed in fault injection of 3.6$\times$ (See Table 5.5). The 95% confidence error bounds overlap between these two measurements, but it is possible that the improvement seen in neutron radiation is slightly skewed towards greater improvement because of the time it takes to reprogram the device during radiation exposure.

6.4.1 Comparison to Other Mitigation Techniques

Using the B13 neutron test results from [42], neutron cross sections for each version were calculated. The provided FITs were converted to cross sections using the fast neutron flux for New York City (13 neutrons/cm^2/hour). In the X-TMR/VERI-Place experiment 30 redundant copies of the B13 circuit were placed in each implementation. This experiment contains 512 redundant copies of the B13 circuit per implementation. For comparison between the two experiments (this experiment and [42]) cross-section data from both experiments were adjusted to represent the cross section of a single B13 circuit.

Table 6.4: Neutron Cross Section Comparison of Mitigation Techniques

Implementation	Cross Section (cm^2)	Improvement	Overhead
Unmitigated [42]	5.38E-9±7.69E-11	1.00×	1.00×
X-TMR [42]	4.41E-9±2.56E-11	1.22×	4.56×
VERI-Place [42]	3.44E-10±1.03E-11	15.67×	4.56×
Unmitigated	1.81E-11±3.30E-12	1.00×	1.00×
TMR w/o Scrubbing	3.51E-12±6.75E-13	5.15×	4.23×
TMR w/ Scrubbing	3.45E-13±1.35E-13	52.41×	4.23×

The single B13 designs cross section, normalized improvement in reliability and the overhead cost of the applied mitigation techniques are shown in Table 6.4. The normalized improvement in reliability is how many times smaller the cross section of the mitigated design is compared to the corresponding unmitigated design. It is interesting to note that the cross section of the unmitigated design in the X-TMR/VERI-Place experiment is much larger than that of the unmitigated design tested in this experiment.

The X-TMR/VERI-Place experiment was conducted on a Virtex-5LX50T Xilinx SRAM-based FPGA [42]. The neutron cross sections of a single CRAM bit in that FPGA compared to the Kintex 7 325T used in this experiment are fairly close together. It is not clear what is causing the difference in cross section of the unmitigated designs from each experiment.

X-TMR showed a relative improvement in reliability of 1.22× whereas the BL-TMR design demonstrated a 5.15× improvement in reliability. This is most likely due to the fine-grain

feedback TMR approach. Applying VERI-Place to X-TMR design demonstrated a 12.84× improvement in reliability [58]. Applying configuration scrubbing to the BL-TMR improved the designs reliability by an order of magnitude.

6.5 B13 and AES on Two Different FPGA Architectures

The total neutron fluence exposure of each design and the number of observed failures is reported in Table 6.5. As in fault injection, the two FPGAs are shown to be comparable for SEU sensitivity. The cross section of the B13 unmitigated design was very similar between the Stratix V and Kintex 7 FPGA. The same is true for the AES unmitigated design. As with fault injection, the mitigated version of each design demonstrates a greater improvement and a smaller sensitive cross section in opposite FPGAs. The AES design preferred the Stratix V FPGA and the B13 design preferred the Kintex 7 FPGA. TMR and scrubbing reduced the cross section of each design by a factor of 4× to 54×.

Table 6.5: Neutron Radiation Results

Design		ITC'99 Benchmark B13		128-bit AES IP Core	
Description		Unmitigated	Mitigated	Unmitigated	Mitigated
Altera Stratix V	Total Fluence	6.24×10^{10}	1.06×10^{11}	7.88×10^{9}	1.13×10^{11}
	Observed Failures	661	102	365	291
	Cross Section (cm^2) (95% Conf. Interval)	1.06×10^{-8} (9.79×10^{-9}; 1.14×10^{-8})	9.62×10^{-10} (7.75×10^{-10}; 1.15×10^{-9})	4.63×10^{-8} (4.16×10^{-8}; 5.11×10^{-8})	2.58×10^{-9} (2.29×10^{-9}; 2.88×10^{-9})
	Improvement (95% Conf. Interval)	1.00× (0.86×; 1.17×)	11.02× (8.53×; 14.72×)	1.00× (0.81×; 1.23×)	17.93× (14.43×; 22.34×)
Xilinx Kintex 7	Total Fluence	1.08×10^{11}	2.83×10^{11}	4.44×10^{10}	7.84×10^{10}
	Observed Failures	838	41	1,621	720
	Cross Section (cm^2) (95% Conf. Interval)	7.79×10^{-9} (7.26×10^{-9}; 8.32×10^{-9})	1.45×10^{-10} (1.00×10^{-10}; 1.89×10^{-10})	3.65×10^{-8} (3.48×10^{-8}; 3.83×10^{-8})	9.19×10^{-9} (8.52×10^{-9}; 9.86×10^{-9})
	Improvement (95% Conf. Interval)	1.00× (0.87×; 1.15×)	53.85× (38.44×; 77.48×)	1.00× (0.91×; 1.10×)	3.98× (3.53×; 4.50×)

6.5.1 Comparison of Device Cross Section

The neutron cross section for each FPGA device as a whole was also measured as part of this experiment. Table 6.6 shows the total fluence expose and number of detected configuration upsets for each FPGA. Both devices were found to have a similar cross section. Using the total number of CRAM bits, the cross section of a single CRAM cell is approximated. The neutron cross section single CRAM cell matches measurement reported reported by Xilinx [56][3].

Table 6.6: Device Neutron Radiation Data

Device	Altera Stratix V	Xilinx Kintex 7
Total Fluence	2.89×10^{11}	1.06×10^{12}
Total Upsets	138,040	551,952
Device Cross Section (95% Conf. Interval)	4.78×10^{-7} (4.75×10^{-7}; 4.80×10^{-7})	5.22×10^{-7} (5.20×10^{-7}; 5.24×10^{-7})
Total CRAM Bits	98,681,196	72,778,176
CRAM Bit Cross Section	4.84×10^{-15}	7.18×10^{-15}

6.6 Comparison to Fault Injection Results

Less improvement was demonstrated in the beam than in fault injection, however the relationships between SEU mitigation benefits still hold. Some degradation is to be expected as radiation can introduce more kinds of faults than fault injection can. However, test logs suggest that perhaps SEUs were accumulating faster than the scrubbing methods could handle. SEU accumulation was not tested in fault injection.

[3] As far as can be determined to date, there is no publicly available measurement of the neutron cross section estimate for a single CRAM bit in an Altera Stratix V FPGA.

CHAPTER 7. CONCLUSION

Estimating the SEU sensitivity of an FPGA design is important for many reasons. FPGA designs may fail due to upsets in configuration memory caused by ionizing radiation. The occurrence of an SEU does not however guarantee that an FPGA design will fail. The likelihood of an SEU causing a design to fail is different for every FPGA design. The SEU sensitivity of an FPGA design depends on the size of the design, the target FPGA architecture, the types of SEU mitigation techniques applied to it, etc. Knowing the SEU sensitivity of a design is very helpful in preparing mission-critical applications for deployment, for studying SEU mitigation techniques and improving them, and for analyzing the effects of different factors on reliability.

Several approaches have been developed to estimated the SEU sensitivity of FPGA designs through either fault injection or radiation testing. In general, the approaches consist of running an active design on an FPGA and comparing its outputs against a set of golden output vectors or against the outputs of a golden design running in lockstep. Most approaches have a test fixture that supports the active operation of the design and is able to detect functional errors. The examples covered placed supporting test fixture logic, including functional error detection logic, outside the scope of fault injection and radiation testing such that the test fixture and associated logic were failure-immune. This improves the accuracy of estimation, but comes at the cost of additional FPGAs or re-occurring engineering costs of creating custom test boards. Additional advantages and limitations of the several approaches were discussed.

An on-chip error detection SEU sensitivity estimation approach was presented that tried to incorporate the advantages of the other approaches into a low cost, single FPGA solution that was easier to implement. This solution consumes additional FPGA resources by placing most of the necessary test fixture components, including functional error detection logic and additional design instances, on the same FPGA as the design under test. All components are subject to fault injection and radiation testing. Implication of placing all test fixture logic within the scope of fault injection

and radiation testing were addressed in Chapter 4. The test fixture logic was protected with SEU mitigation techniques, and several variations of FPGA designs were loaded into the on-chip test fixture. The driving concept behind on-chip error detection is that when an SEU causes any design instance to fail, that failure can be detected by comparing design instances against each other or against a set of SEU mitigated golden test vectors also placed on-chip.

Several fault injection and radiation testing experiments were conducted using an on-chip error detection approach to estimate the SEU sensitivity of FPGA designs. Three main goals were accomplished through theses experiments. First, the individual and combined reliability benefits of three SEU mitigation techniques were demonstrated. Second, fault injection results obtained from TMR protected and DWC protected on-chip functional error detection logic were compared to determine which approach provided the most favorable results. Finally, the SEU sensitivities of FPGA designs on two different FPGA architectures were compared to perform an cross architecture comparison of design reliability and to evaluate the benefits of SEU mitigation techniques across architectures.

Five variations of SEU mitigation techniques were applied to the LEON3 soft processor. Functional errors were detected using a one-to-one lockstep on-chip comparison scheme. Improvement was measured in terms of sensitivity reduction for fault injection and cross section reduction for neutron radiation testing when compared to the baseline unmitigated design. The results from *both* fault injection and radiation testing demonstrate that each variation of SEU mitigation techniques reduce the SEU sensitivity of the LEON3, and that improvement increases as more mitigation techniques are combined. Similar results were found for the B13 design, only errors were detected using a one-to-many on-chip comparison scheme with an on-chip set of golden test vectors. Applying TMR alone provided a $8\times$ and $5\times$ reduction in cross section for the LEON3 and B13 design respectively. Applying TMR with configuration scrubbing provided a $49\times$ and a $52\times$ reduction in cross section respectively. On-chip error detection was used to demonstrate that combining complementary SEU mitigation techniques provides the greatest reduction in SEU sensitivity.

The functional error detection logic used to compare and reduce the outputs of multiple design instances to a single bit that signifies whether or not a failure occurs can be protected from SEU using TMR or DWC. TMR triplicates the comparison and reduction network and adds

voters to mask SEUs in the functional error detection logic. DWC duplicates the comparison and reduction network and only reports a failure when both duplicate copies detect a failure at the same time. SEU sensitivities of several FPGA designs were estimated through fault injection using on-chip functional error detection logic that was either protected by TMR or DWC. It was found, for error detection, that in most cases DWC provided more favorable results with a higher MUTF and greater improvement in reliability demonstrated between unmitigated and mitigated FPGA designs. For this reason, DWC protected functional error detection logic was used later in radiation testing.

Unmitigated and mitigated versions of two benchmark designs were tested for SEU sensitivity using on-chip error detection on two 28nm FPGAs with different architectures. Fault injection and radiation testing were used for comparison. Similarities were found between the two FPGAs in the SEU sensitivity of the unmitigated version of each design and the overall device cross section. The SEU mitigated versions of each design favored opposite FPGAs suggesting that benefits from SEU mitigation techniques are design and FPGA architecture dependent. It was found that these FPGAs are fairly comparable to each other for design SEU sensitivity and that TMR and scrubbing can reduce design sensitivity to SEUs anywhere from $12\times$ to $728\times$ in fault injection and $4\times$ to $54\times$ in radiation testings.

One drawback of the on-chip error detection experiments conducted in this thesis is that there is no way of differentiating between a true error, a false-positive, or a false-negative. With the information that was available, results were presented and analyzed. Future work is this field should consider providing a way of confirming the occurrence of an error in a definitive manner.

The on-chip error detection mechanisms used in these experiments are important for future work in fast error detection and recovery. As on-chip error detection is further researched it can be used to improve fast error detection and recovery, which has lower overhead than TMR. All of the detected SEUs that cause a failure to be reporter matter even if they are reporting a false-positive. Ideally there would be no way for a single SEU to cause a false-positive error detection report. But there are known issue in placement and routing that create these vulnerabilities [31, 32]. The research conducted in this thesis may be helpful in identifying single point failures in on-chip error detection mechanisms and assisting in improving their implementation.

REFERENCES

[1] Caulfield, A. M., et al., 2016. "A cloud-scale acceleration architecture." In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–13. 1, 5

[2] Wirthlin, M., 2015. "High-reliability FPGA-based systems: Space, high-energy physics, and beyond." *Proceedings of the IEEE,* **103**(3), March, pp. 379–389. 1, 5

[3] Eormonds, L., Barnes, C., and Scheick, L., 2000. An introduction to space radiation effects. 1, 7, 8

[4] JESD89A, J. S., 2006. "Measurement and reporting of alpha particle and terrestrial cosmic ray-induced soft errors in semiconductor devices." *JEDEC solid state technology association*. 1, 6, 8, 21, 68

[5] Katz, R., LaBel, K., Wang, J., Cronquist, B., Koga, R., Penzin, S., and Swift, G., 1997. "Radiation effects on current field programmable technologies." *IEEE Transactions on Nuclear Science,* **44**(6), December, pp. 1945–1956. 1

[6] Quinn, H., and Graham, P., 2005. "Terrestrial-based radiation upsets: a cautionary tale." In *13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05)*, pp. 193–202. 1, 6, 7

[7] Lesea, A., Drimer, S., Fabula, J. J., Carmichael, C., and Alfke, P., 2005. "The Rosetta experiment: atmospheric soft error rate testing in differing technology FPGAs." *IEEE Transactions on Device and Materials Reliability,* **5**(3), Sept, pp. 317–328. 1, 6, 7

[8] Pratt, B., Caffrey, M., Graham, P., Morgan, K., and Wirthlin, M., 2006. "Improving FPGA design robustness with partial TMR." In *2006 IEEE International Reliability Physics Symposium Proceedings*, pp. 226–232. 1

[9] Carmichael, C., Fuller, E., Blain, P., and Caffrey, M., 1999. "SEU mitigation techniques for Virtex FPGAs in space applications." In *Proceeding of the Military and Aerospace Programmable Logic Devices International Conference (MAPLD)*, Citeseer, p. C2. 2

[10] Morgan, K. S., McMurtrey, D. L., Pratt, B. H., and Wirthlin, M., 2007. "A comparison of TMR with alternative fault-tolerant design techniques for FPGAs." *IEEE Transactions on Nuclear Science,* **54**(6), Dec, pp. 2065–2072. 2, 14, 29, 38

[11] Rollins, N., Fuller, M., and Wirthlin, M., 2010. "A comparison of fault-tolerant memories in SRAM-based FPGAs." In *Aerospace Conference, 2010 IEEE*, pp. 1–12. 2, 17, 18

[12] Johnson, J. M., and Wirthlin, M., 2010. "Voter insertion algorithms for FPGA designs using triple modular redundancy." In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, ACM, pp. 249–258. 2, 14

[13] Sterpone, L., and Violante, M., 2005. "Analysis of the robustness of the TMR architecture in SRAM-based FPGAs." *Nuclear Science, IEEE Transactions on,* **52**(5), oct., pp. 1545 – 1549. 2

[14] Stoddard, A., Gruwell, A., Zabriskie, P., and Wirthlin, M., 2016. "A hybrid approach to FPGA configuration scrubbing." *IEEE Transactions on Nuclear Science,* **PP**(99), pp. 1–1. 2, 17

[15] Quinn, H., et al., 2013. "Fault simulation and emulation tools to augment radiation-hardness assurance testing." *IEEE Transactions on Nuclear Science,* **60**(3), June, pp. 2119–2142. 2, 19, 23, 28, 51, 66

[16] Quinn, H., 2014. "Challenges in testing complex systems." *IEEE Transactions on Nuclear Science,* **61**(2), April, pp. 766–786. 2, 7, 8, 10, 19, 21, 22, 23, 24, 31, 41

[17] Silburt, A. L., Evans, A., Burghelea, A., Wen, S. J., Ward, D., Norrish, R., and Hogle, D., 2008. "Specification and verification of soft error performance in reliable internet core routers." *IEEE Transactions on Nuclear Science,* **55**(4), Aug, pp. 2389–2398. 5

[18] Baumann, R. C., 2001. "Soft errors in advanced semiconductor devices - part I: The three radiation sources." *IEEE Transactions on Device and Materials Reliability,* **1**(1), Mar, pp. 17–22. 6

[19] Battezzati, N., Sterpone, L., and Violante, M., 2011. "Reconfigurable field programmable gate arrays: Failure modes and analysis." In *Reconfigurable Field Programmable Gate Arrays for Mission-Critical Applications*. Springer, pp. 37–83. 6, 7, 8, 9, 70

[20] Wang, F., and Agrawal, V. D., 2008. "Single event upset: An embedded tutorial." In *21st International Conference on VLSI Design (VLSID 2008)*, pp. 429–434. 7, 8

[21] Hussein, J., and Swift, G. Mitigating single-event upsets White Paper: 7 Series FPGAs WP395, Xilinx, Inc. 8

[22] Chang, M. L., 2008. "Device architecture." *HAUCK, S.; DEHON, A. Reconfigurable computing: the theory and practice of FPGA-based computation. United States: Elsevier*. 10

[23] Graham, P., Caffrey, M., Zimmerman, J., Sundararajan, P., and Johnson, E., 2003. "Consequences and categories of SRAM FPGA configuration SEUs." In *In Proceedings of the International Conference on Military and Aerospace Programmable Logic Devices*, Citeseer. 1, 11, 12

[24] Quinn, H., Graham, P. S., Morgan, K., Krone, J., Caffrey, M. P., and Wirthlin, M., 2008. "An introduction to radiation-induced failure modes and related mitigation methods for Xilinx SRAM FPGAs.." In *ERSA*, pp. 139–145. 1, 11

[25] Wirthlin, M., Johnson, E., Rollins, N., Caffrey, M., and Graham, P., 2003. "The reliability of FPGA circuit designs in the presence of radiation induced configuration upsets." In *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2003. FCCM 2003.*, pp. 133–142. 12, 29

[26] Neumann, P. G., 1994. *Computer-related risks*. Addison-Wesley Professional. 13

[27] Carmichael, C., 2001. Triple module redundancy design techniques for Virtex FPGAs Tech. rep., Xilinx Corporation, November 1, XAPP197 (v1.0). 14

[28] Ostler, P. S., Caffrey, M. P., Gibelyou, D. S., Graham, P. S., Morgan, K. S., Pratt, B. H., Quinn, H., and Wirthlin, M., 2009. "SRAM FPGA reliability analysis for harsh radiation environments." *IEEE Transactions on Nuclear Science,* **56**(6), Dec, pp. 3519–3526. 14, 19, 31

[29] Niknahad, M., Sander, O., and Becker, J., 2012. "Fine grain fault tolerance – a key to high reliability for FPGAs in space." In *Aerospace Conference, 2012 IEEE*, pp. 1–10. 14

[30] Manuzzato, A., Gerardin, S., Paccagnella, A., Sterpone, L., and Violante, M., 2007. "Effectiveness of TMR-based techniques to mitigate alpha-induced SEU accumulation in commercial SRAM-based FPGAs." In *9th Eur. Conf. Radiation and Its Effects on Components and Systems*, pp. 1–7. 15

[31] Sterpone, L., Violante, M., and Rezgui, S., 2006. "An analysis based on fault injection of hardening techniques for SRAM-based FPGAs." *IEEE Trans. Nucl. Sci,* **53**(4), Aug, pp. 2054–2059. 15, 81

[32] Quinn, H., et al., 2007. "Domain crossing errors: Limitations on single device triple-modular redundancy circuits in Xilinx FPGAs." *IEEE Transactions on Nuclear Science,* **54**(6), Dec, pp. 2037–2043. 15, 63, 81

[33] Johnson, J., Howes, W., Wirthlin, M., McMurtrey, D. L., Caffrey, M., Graham, P., and Morgan, K., 2008. "Using duplication with compare for on-line error detection in FPGA-based designs." In *2008 IEEE Aerospace Conference*, pp. 1–11. 16, 30, 40, 41, 43, 62

[34] Herrera-Alzu, I., and Lopez-Vallejo, M., 2013. "Design techniques for Xilinx Virtex FPGA configuration memory scrubbers." *Nuclear Science, IEEE Transactions on,* **60**(1), Feb, pp. 376–385. 16

[35] Berg, M., Poivey, C., Petrick, D., Espinosa, D., Lesea, A., LaBel, K., Friendlich, M., Kim, H., and Phan, A., 2008. "Effectiveness of internal versus external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis." *IEEE Transactions on Nuclear Science,* **4**(55), pp. 2259–2266. 16

[36] Li, Y., Nelson, B., and Wirthlin, M., 2013. "Reliability models for SEC/DED memory with scrubbing in FPGA-based designs." *IEEE Transactions on Nuclear Science,* **60**(4), Aug, pp. 2720–2727. 17

[37] Mukherjee, S. S., Emer, J., Fossum, T., and Reinhardt, S. K., 2004. "Cache scrubbing in microprocessors: myth or necessity?." In *10th Proc. IEEE Pacific Rim Int. Symp. Dependable Computing*, pp. 37–42. 18

[38] Schwank, J. R., Shaneyfelt, M. R., and Dodd, P. E., 2013. "Radiation hardness assurance testing of microelectronic devices and integrated circuits: Radiation environments, physical mechanisms, and foundations for hardness assurance." *IEEE Transactions on Nuclear Science,* **60**(3), June, pp. 2074–2100. 21

[39] Association, E. I., et al., 1996. "Test procedures for the measurement of single-event effects in semiconductor devices from heavy ion irradiation." *EIA/JEDEC Standard*(57). 21

[40] Abate, F., Sterpone, L., Lisboa, C. A., Carro, L., and Violante, M., 2009. "New techniques for improving the performance of the lockstep architecture for SEEs mitigation in FPGA embedded processors." *IEEE Transactions on Nuclear Science,* **56**(4), Aug, pp. 1992–2000. 24

[41] Aguirre, M. A., Tombs, J. N., Munoz, F., Baena, V., Guzman, H., Napoles, J., Torralba, A., Fernandez-Leon, A., Tortosa-Lopez, F., and Merodio, D., 2007. "Selective protection analysis using a SEU emulator: Testing protocol and case study over the Leon2 processor." *IEEE Transactions on Nuclear Science,* **54**(4), Aug, pp. 951–956. 26, 27

[42] Quinn, H., Robinson, W. H., Rech, P., Aguirre, M., Barnard, A., Desogus, M., Entrena, L., Garcia-Valderas, M., Guertin, S. M., Kaeli, D., Kastensmidt, F. L., Kiddie, B. T., Sanchez-Clemente, A., Reorda, M. S., Sterpone, L., and Wirthlin, M., 2015. "Using benchmarks for radiation testing of microprocessors and FPGAs." *IEEE Transactions on Nuclear Science,* **62**(6), Dec, pp. 2547–2554. 27, 36, 40, 46, 48, 68, 76, 90

[43] Sterpone, L., and Violante, M., 2007. "A new partial reconfiguration-based fault-injection system to evaluate SEU effects in SRAM-based FPGAs." *IEEE Transactions on Nuclear Science,* **54**(4), Aug, pp. 965–970. 27

[44] Lopez-Ongil, C., Garcia-Valderas, M., Portela-Garcia, M., and Entrena, L., 2007. "Autonomous fault emulation: A new FPGA-based acceleration system for hardness evaluation." *IEEE Transactions on Nuclear Science,* **54**(1), Feb, pp. 252–261. 27

[45] Alderighi, M., Casini, F., D'Angelo, S., Pastore, S., Sechi, G. R., and Weigand, R., 2007. "Evaluation of single event upset mitigation schemes for SRAM based FPGAs using the FLIPPER fault injection platform." In *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, pp. 105–113. 27, 28

[46] Johnson, E., Wirthlin, M., and Caffrey, M., 2002. "Single-event upset simulation on an FPGA." In *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, CSREA Press, pp. 68–73. 28, 29

[47] Harward, N. A., Gardiner, M. R., Hsiao, L. W., and Wirthlin, M., 2015. "Estimating soft processor soft error sensitivity through fault injection." In *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines*, pp. 143–150. 30, 31

[48] Cannon, M., Wirthlin, M., Camplani, A., Citterio, M., and Meroni, C., 2015. "Evaluating Xilinx 7 series GTX transceivers for use in high energy physics experiments through proton irradiation." *IEEE Transactions on Nuclear Science,* **62**(6), Dec, pp. 2695–2702. 32

[49] Li, Y., Nelson, B., and Wirthlin, M., 2010. "Synchronization techniques for crossing multiple clock domains in FPGA-based TMR circuits." *IEEE Transactions on Nuclear Science,* **57**(6), Dec, pp. 3506–3514. 36

[50] Cobham Gaisler LEON3 processor
http://www.gaisler.com/index.php/products/processors/leon3. 47, 87

[51] Corno, F., Reorda, M. S., and Squillero, G., 2000. "RT-level ITC'99 benchmarks and first ATPG results." *IEEE Design Test of Computers,* **17**(3), Jul, pp. 44–53. 48

[52] Quinn, H., et al., 2009. "A test methodology for determining space readiness of Xilinx SRAM-Based FPGA devices and designs." *IEEE Trans. Instrum. Meas.,* **58**(10), Oct, pp. 3380–3395. 52

[53] Intel, I., 2016. *Altera Fault Injection IP Core User Guide*. 53

[54] Ramachandran, P., et al., 2008. "Statistical fault injection." In *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, pp. 122–127. 54

[55] Spilla, A., et al., 2011. "Run-time soft error injection and testing of a microprocessor using FPGAs." In *Proc. Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen*. 68

[56] Xilinx Inc. Device reliability report, second half 2015. 70, 78

[57] Wirthlin, M., Keller, A., Lee, D. S., Draper, J., McCloskey, C., and Ridd, P., 2016. "SEU mitigation and validation of the LEON3 soft processor using triple modular redundancy for space processing." In *Proc. 2016 ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, pp. 205–214. 72

[58] Desogus, M., Sterpone, L., and Codinachs, D. M., 2014. "Validation of a tool for estimating the effects of soft-errors on modern SRAM-based FPGAs." In *On-Line Testing Symposium (IOLTS), 2014 IEEE 20th International*, pp. 111–115. 77

APPENDIX A. DETAILS ON THE BENCHMARK DESIGNS USED

This appendix includes more detailed information on each of the designs tested. Many versions of each design were tested that implemented various SEU mitigation techniques and error detection mechanisms. Only a selection of the design variations are covered in this appendix

A.1 LEON3

The LEON3 is an open-source soft processor core distributed by Cobham Gaisler AB as part of their GRLIB IP Library. It conforms to the IEEE Standard for a 32-bit Microprocessor Architecture and Version 8 of the Scalable Processor Architecture (SPARC V8) [50]. It features a 7-stage integer pipeline with a Harvard architecture. Additional peripherals may be easily incorporated into a LEON3 system due to its bus centric system-on-chip design. As part of the GRLIB IP Library, the LEON3 connects to additional IP cores via an on-chip bus. The GRLIB IP library supports the AMBA-2.0 AHB/APB bus – a widely used, royalty free industry standard.

The LEON3 used in this experiment originated from the LEON3/GRLIB Release 1.4.0-b4154. For this test, a minimal configuration was given to the "leon3-xilinx-kc705" design – which targets the Xilinx Kintex-7 KC705 development board. The stripped-down configuration of the LEON3 processor was chosen for this experiment to test the sensitivity of the core internal architecture and to simplify the construction of the test. This simplified configuration *excluded* the following default architectural components:

- Instruction and Data Caches
- Interrupt Controller
- Memory Management Unit (MMU)
- Debug Support Unit
- External Memory Controllers

All unnecessary I/O peripherals were excluded and all instruction and data memory was held in internal BRAM resources to avoid the need for an external memory controller. In addition, the PLL clock controllers were removed and a 200 MHz external clock was internally divided by four to create a 50 MHz global clock. Figure A.1 reflects the final configuration of both the LEON3 processor core and connected peripherals via the on-chip bus.

The LEON3 processor is programmed to execute the Dhrystone Version 2.1 benchmark. Dhrystone is designed to test integer performance of a processor like that of the 7-stage integer pipeline found in the LEON3. Executing this program assists in error detection as the processors outputs are compared against another processor core running the same benchmark program in lockstep. In this experiment, Dhrystone is run for 10,000 iterations in a continuous loop.

The Dhrystone output is sent across UART for external monitoring. The benchmark executes continuously to guarantee that the LEON3 processor remains active throughout the test. The

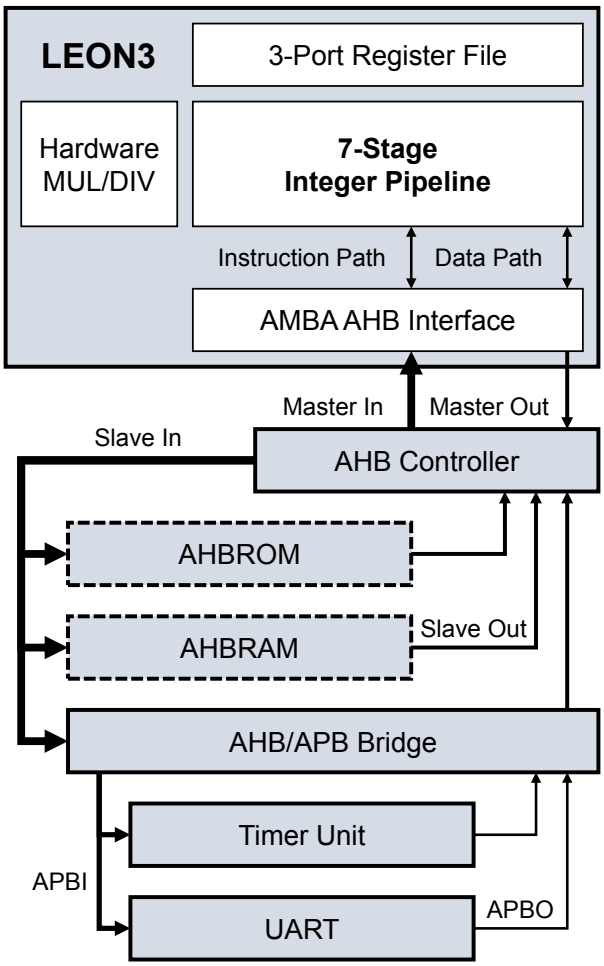


Figure A.1: LEON3 System Architecture Under Test

caches are disabled to force the processor to communicate more frequently on the bus and facilitate more frequent error checking.

A.2 Test Fixture

The test fixture implemented on a single FPGA for the one-to-one lockstep comparison mechanism of the LEON3 design is shown in Figure A.2. The bus signals from the two identical designs instances are compared in a clock-by-clock cycle fashion using a large XOR reduction network that sets an error status register high if any bit pair disagreement is found between the two instances. The XOR reduction network for this design is triplicated to prevent false-positive errors (i.e. errors within the test logic that are reported as errors within the design under test).

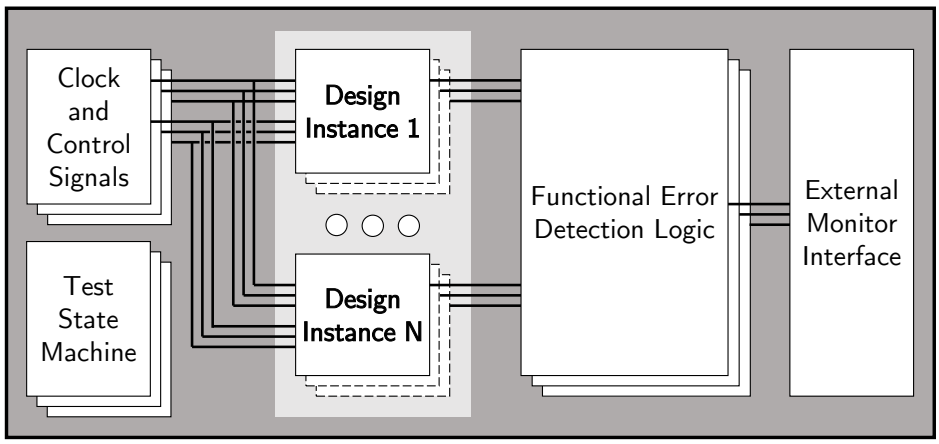


Figure A.2: LEON3 On-Chip Test Fixture

A.3 Resource Utilization

Table A.1 shows the average resource utilization of: the test fixture by itself, a single non-TMR LEON3 processor core, a single TMR LEON3 processor core, and a single TMR processor core with supporting logic for BRAM scrubbing. The table also includes the overall resource utilization of all three design versions, which includes the logic resources for the test fixture logic and two instances of the LEON3. The percentage next to each metric is the utilization percentage of available device resources for the Xilinx Kintex 7 XC7325T FPGA. The overhead of the SEU mitigation technique compared to the non-TMR version is also shown. Figure A.3 shows the placement and routing of the unmitigated, non-TMR design version compared to the mitigated, TMR with BRAM scrubbing version.

Table A.1: LEON3 Design Variation Resource Utilization

Average Resource Utilization (Per Test Fixture or Single LEON3)				
Resource	Test Fixture	Non-TMR	TMR	TMR w/BRAM Scrubbing
Slice	1,886 (3.70%)	1,397 (2.74%)	5,401 (10.6%) 3.87×	6,667 (13.1%) 4.77×
FF	2,726 (0.67%)	1,950 (0.48%)	5,850 (1.44%) 3.00×	6,165 (1.51%) 3.16×
LUT	3,274 (1.61%)	4,088 (2.01%)	16,041 (7.87%) 3.92×	18,094 (8.88%) 3.00×
BRAM	0 (0.00%)	50 (11.2%)	150 (33.7%) 3.00×	150 (33.7%) 3.00×
DSP	0 (0.00%)	1 (0.12%)	3 (0.36%) 3.00×	3 (0.36%) 3.00×
Total Resource Utilization (Test Fixture and Two LEON3s)				
Slice	–	4,546 (8.92%)	12,746 (25.0%) 2.80×	15,294 (30.0%) 3.36×
FF	–	6,626 (1.63%)	14,426 (3.54%) 2.18×	15,056 (3.69%) 2.28×
LUT	–	11,471 (5.63%)	35,311 (17.3%) 3.08×	39,453 (19.4%) 3.44×
BRAM	–	100 (22.5%)	300 (67.4%) 3.00×	300 (67.4%) 3.00×
DSP	–	2 (0.24%)	6 (0.71%) 3.00×	6 (0.71%) 3.00×

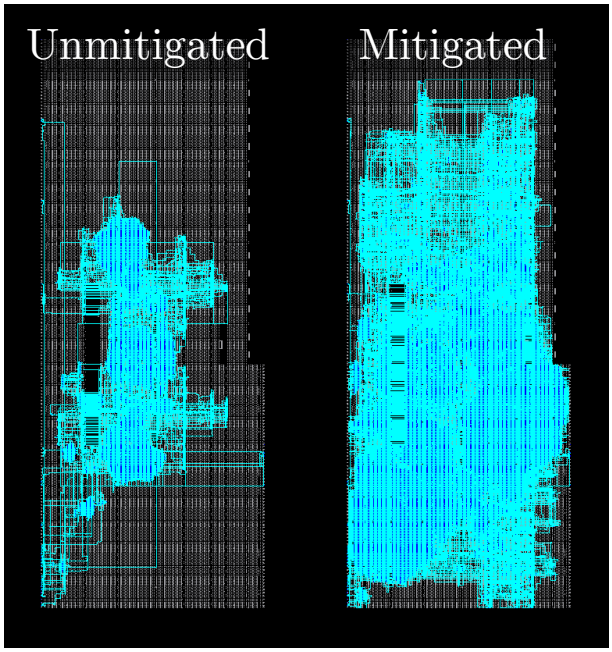


Figure A.3: LEON3 Design Test Infrastructure FPGA Layout

A.4 ITC'99 B13 Benchmark Design

The B13 is a small finite state machine that interacts with a weather station. It is part of a larger set of hardware benchmarks and has been used in other research as a reliability benchmark for radiation testing of FPGAs [42]. Included with this benchmark design is a set of approximately 7,600 input test vectors with corresponding golden output test vectors. For experiments with this design, these test vectors were integrated into the test fixture used to estimate SEU sensitivity. The test vectors themselves were protected from SEUs by applying TMR and internal BRAM scrubbing to them.

A.4.1 Test Fixture

The test fixture implemented for the B13 design is depicted in Figure A.4. This test fixture allows multiple instances of the B13 to be tested at the same time. Since the B13 is so small, if only one or two instances were placed within the test fixture, then the overall SEU sensitivity would be very small. This would require long fault injection and radiation test times. To accelerate testing, 512 instances of the B13 are instanced within the test fixture. The same test fixture setup was used in the Xilinx and Altera designs. An XOR reduction network was used to detect errors in both FPGAs. TMR was applied to the reduction network in the Xilinx design implementations, and DWC was applied to the reduction network in the Altera design implementations.

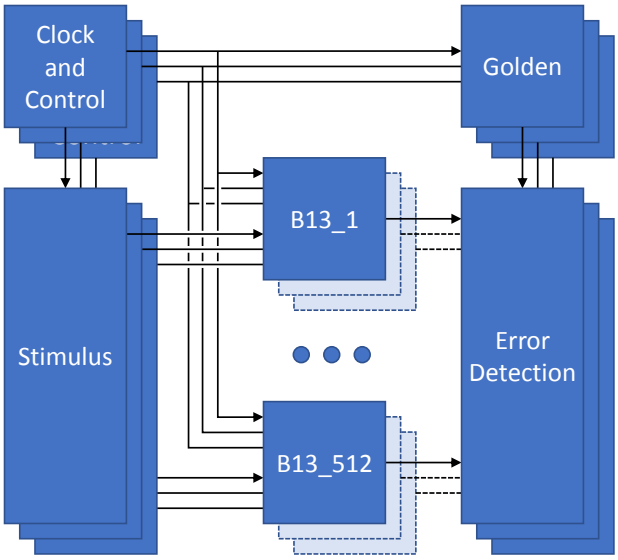


Figure A.4: B13 On-Chip Test Fixture

A.5 Resource Utilization

The resource utilization of a single B13 instance in a Xilinx Kintex 7 FPGA is shown in Table A.2. This table includes approximate timing penalties for applying TMR. The penalties are significant because of how small the design is in comparison to the number of inserted feedback synchronization voters. The resource utilization for the test infrastructure and 512 instances of the B13 on a Xilinx Kintex 7 FPGA is shown in Table A.3. Both tables compare the unmitigated design without TMR applied to it against a TMR version of the design. The overhead of applying TMR is shown in the "Comparison" columns of each table. Figure A.5 shows the FPGA layout of the entire test fixture with all 512 B13 instances and compares the unmitigated and mitigated versions.

Table A.2: Single B13 Implementation on a Xilinx Kintex-7 FPGA

	Unmitigated	Mitigated	Comparison
Flip-flops	52	156	3.00×
LUTs	36	107	2.97×
Slices	15	53	3.53×
Max Copies on Device	3,396	961	0.283×
Min Clock Period	1.364 ns	3.366 ns	2.47×
Max Clock Frequency	733.14 MHz	297.09 MHz	0.405×

Table A.6 shows the resource utilization of the B13 test fixture with 512 instances of the B13 on an Altera Stratix V FPGA. An adaptive logic module (ALM) in an Altera device is similar to a slice in a Xilinx FPGA. A slice in a Xilinx Kintex 7 FPGA consists of four 6-input look-up

Table A.3: Implementation of 512 B13 copies on a Xilinx Kintex-7 FPGA

	Unmitigated	Mitigated	Comparison
Flip-flops	27,025 (12.17%)	80,273 (19.69%)	2.97×
LUTs	24,807 (12.17%)	137,350 (67.39%)	5.54×
Slices	8,757 (17.19%)	37,037 (72.69%)	4.23×
BRAM 36K	12 (2.70%)	12 (2.70%)	1.0×
BRAM 18K	6 (0.67%)	6 (0.67%)	1.0×

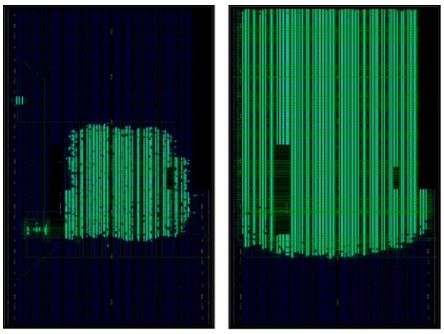


Figure A.5: Xilinx B13 Design Test Infrastructure FPGA Layout: Unmitigated (Left), Mitigated (Right).

tables, carry-chain logic, and eight flip-flops. An ALM in an Altera Stratix V FPGA consists of an 8-input combinational logic unit, carry-chain addition logic, and 4 flip-flops. A slice represents approximately 3× more computational resources than an ALM. Comparing the utilization metrics between Table A.3 for Xilinx and Table A.4 for Altera, approximately the same amount of resources are used for each of the design versions between the two FPGAs. Figure A.6 shows the resource block utilization and wire resources utilization of the unmitigated and mitigated versions of the B13 design instances and test fixtures.

There is an interesting difference between the Xilinx and Altera implementation of internal memory scrubbing for the test vectors used to keep the designs active and detect output errors. Xilinx's synthesis tool inferred true dual-port memory modules and dedicates one port to the reading and writing of memory for scrubbing, and the other port to the reading of memory for design stimulus and output verifications. Altera's synthesis tool inferred simple dual-port memory modules where one port is dedicated to reading and one port is dedicated to writing. Two copies of the test vector memory modules were made. One copy's read port was dedicated to scrubbing logic that wrote corrected memory words back to both copies using their respective write port's. The other copy's read port was dedicated to reading test vectors for use by the test fixture. Both implementations are a valid approach.

Table A.4: Implementation of 512 B13 copies on an Altera Stratix V FPGA

	Unmitigated	Mitigated	Comparison
ALMs	30,255 (12.9%)	96,926 (41.3%)	3.20×
Registers	34,559 (3.7%)	91,903 (9.8%)	2.67×
Total Wire Utilization	2.7%	18.1%	6.70×

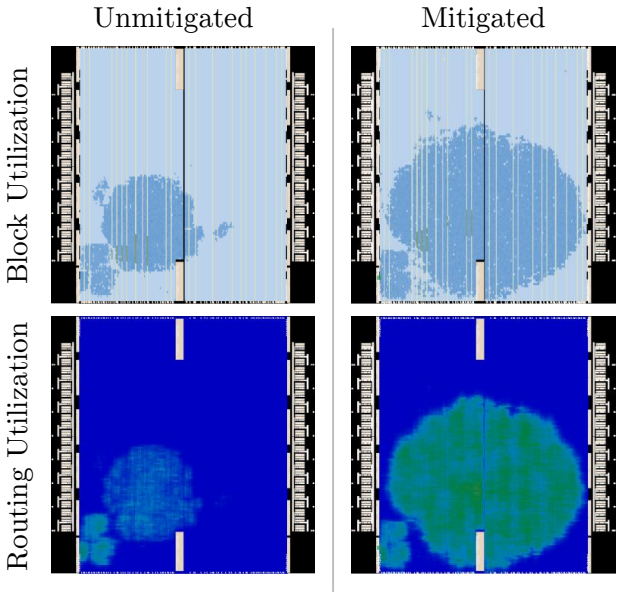


Figure A.6: Altera B13 Design Test Infrastructure FPGA Layout

A.6 AES 128-bit Cryptography IP Core

The AES 128-bit cryptography core is an open source core from OpenCores.org. The implementation used in this thesis is fully pipelined. Once the pipeline is full, this implementation produces valid data every clock cycle. The on-chip test fixture used to estimate the SEU sensitivity of this core is designed to keep the IP core as active as possible while still monitoring for errors on a clock-by-clock cycle basis.

A.7 Test Fixture

Figure A.7 depicts the on-chip test fixture setup for the AES cryptography core design. This is a one-to-one comparison scheme of two chains of design instances where each chain consists of two design instances. The purpose of chaining design instances together is so that the correct output of one instance depends on the correct output of the other. This reduces the number of bits that must be compared to determine if a functional error has occurred. Rather than having to compare the outputs of each pair of design instances against each other, the output of the two chains can be compared. This reduces the reduction network size by a factor of two. A feedback loop is created within each chain by connecting the output of the last instance on the chain to the

Table A.5: AES Test Fixture Resource Utilization on an Xilinx Kintex 7 FPGA

	Unmitigated	Mitigated	Comparison
Slices	13,030 (25.6%)	37,821 (74.2%)	2.90×
LUTs	44,860 (22.0%)	139,124 (68.3%)	3.10×
Registers	25,787 (6.3%)	73,787 (18.1%)	2.86×

input of the first instance of the chain. The test control unit fills the pipeline of the chain until the output is valid and then it feeds the output of the chain to its input. This keeps the design active using self generated test vectors.

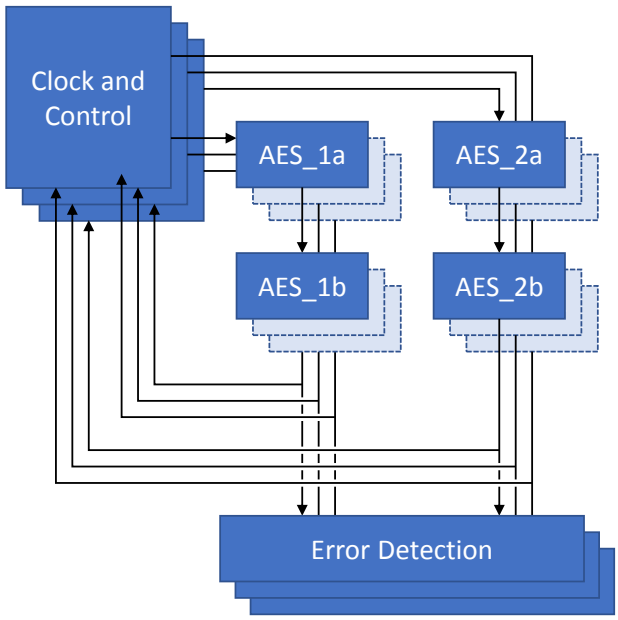


Figure A.7: AES On-Chip Test Fixture

A.8 Resource Utilization

Table A.5 shows the resource utilization of the AES test fixture with unmitigated and mitigated AES encryption core instances on a Xilinx Kintex 7 FPGA. Figure A.8 displays the FPGA layout for both design versions. The test logic in this test fixture is protected from upsets in the error detection mechanism though the use of DWC.

Table A.6 shows the resource utilization of the AES test fixture with unmitigated and mitigated AES encryption core instances on an Altera Kintex 7 FPGA. Figure A.9 displays the FPGA layout for both design versions; the block utilization and the routing resource utilization is displayed. The test logic in this test fixture is also protected from upsets in the error detection mechanism though the use of DWC.

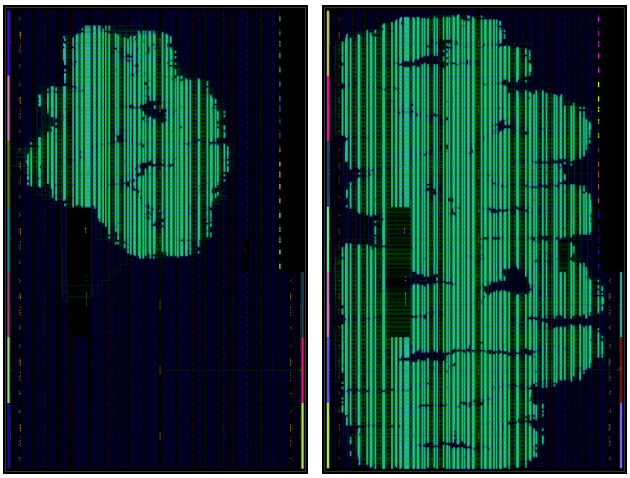


Figure A.8: Xilinx AES Design Test Infrastructure FPGA Layout:Unmitigated (Left), Mitigated (Right).

Table A.6: AES Test Fixture Resource Utilization on an Altera Stratix V FPGA

	Unmitigated	Mitigated	Comparison
ALMs	64,288 (27.4%)	183,554 (78.2%)	2.86×
Registers	33,711 (3.6%)	87,983 (9.4%)	2.61×
Total Wire Utilization	6.4%	19.4%	3.03×

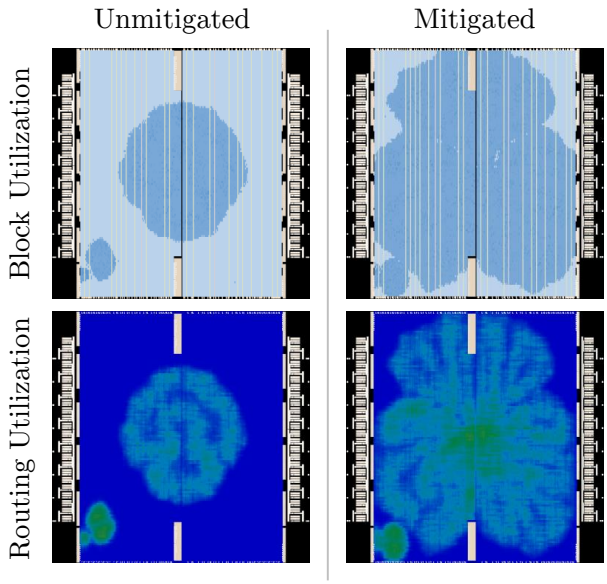


Figure A.9: Altera AES Design Test Infrastructure FPGA Layout

APPENDIX B. FAULT INJECTION AND RADIATION TESTING LOGS

Fault injection data was collected following the approach outlined in Section 5.1 and radiation testing data was collected following the approach outlined in Section 6.1. This appendix includes samples from the data logs collected during fault injection and radiation testing. This data was analyzed to estimate the SEU sensitivity of each FPGA design tested.

B.1 Altera Fault Injection

The following log is from a fault injection run on the B13 non-TMR design implemented on an Altera Stratix V FPGA using DWC protected on-chip error detection logic.

```
[INFO] Brigham Young University - Altera Sensitivity Testing Tool
[INFO] Command: b13NONTMR_TESTBENCH_DWC\test.py -f
[INFO] Configuration:
          SOF File      - b13NONTMR_TESTBENCH_DWC\b13_test_top.sof
          SMH File      - None
          Inject Faults - True
          Weight        - 100.0.0
          Cable         - DE5 Standard [USB-1]
1479413308.52 : [INFO] Initializing Test Run...
1479413308.52 : [INFO] Programming design file b13NONTMR_TESTBENCH_DWC\b13_test_top.sof . . .
1479413334.93 : Info: ******************************************************************
1479413334.94 : Info: Running Quartus Prime Fault Injection Debugger
1479413334.94 :     Info: Version 16.0.0 Build 211 04/27/2016 SJ Standard Edition
1479413334.94 :     Info: Copyright (C) 1991-2016 Altera Corporation. All rights reserved.
1479413334.94 :     Info: Your use of Altera Corporation's design tools, logic functions
1479413334.94 :     Info: and other software and tools, and its AMPP partner logic
1479413334.94 :     Info: functions, and any output files from any of the foregoing
1479413334.94 :     Info: (including device programming or simulation files), and any
1479413334.94 :     Info: associated documentation or information are expressly subject
1479413334.94 :     Info: to the terms and conditions of the Altera Program License
1479413334.94 :     Info: Subscription Agreement, the Altera Quartus Prime License Agreement,
1479413334.94 :     Info: the Altera MegaCore Function License Agreement, or other
1479413334.94 :     Info: applicable license agreement, including, without limitation,
1479413334.94 :     Info: that your use is for the sole purpose of programming logic
1479413334.94 :     Info: devices manufactured by Altera and sold by Altera or its
1479413334.94 :     Info: authorized distributors.  Please refer to the applicable
1479413334.94 :     Info: agreement for further details.
1479413334.94 :     Info: Processing started: Thu Nov 17 13:08:28 2016
1479413334.94 : Info: Command: quartus_fid --cable="DE5 Standard [USB-1]"
                --index=@1=b13NONTMR_TESTBENCH_DWC\b13_test_top.sof#pi --weight=100.0.0 -n 0
1479413334.94 : Info (208808): Using programming cable "DE5 Standard [USB-1]"
1479413334.94 : Info (209060): Started Programmer operation at Thu Nov 17 13:08:43 2016
1479413334.94 : Info (209016): Configuring device index 1
1479413334.94 : Info (209017): Device 1 contains JTAG ID code 0x029030DD
1479413334.94 : Info (209007): Configuration succeeded -- 1 device(s) configured
1479413334.94 : Info (209011): Successfully performed operation(s)
1479413334.94 : Info (209061): Ended Programmer operation at Thu Nov 17 13:08:54 2016
1479413334.94 : Info (208551): Program signature into device 1.
1479413334.94 : [INFO] Design loaded, checking jtag state...
1479413335.69 : JTAG_READ: 0b00000000000000000000000000001111
1479413335.69 : JTAG STATS - Heartbeat: 0 (True) Mismatch w/Others: False (True)
```

```
               Disagree w/Golden: False (True)
1479413335.69 : [INFO] JTAG reading properly, test is running . . .
1479413336.03 : Info (208521): Injects 1 error(s) into device(s)
1479413338.69 : Info (208540): Reading EMR array
1479413338.69 : Info (208543): No frame error detected in device 1.
1479413338.69 : Info (11623): 1 frame error(s) corrected in device 1.
1479413338.69 :     Info (11620):   Corrected error #1 : Single error in frame 0x0606 at bit 0x1CF3.
1479413339.40 : JTAG_READ: 0b00000011000000110000001100001111
1479413339.40 : JTAG STATS - Heartbeat: 3 (True) Mismatch w/Others: False (True)
               Disagree w/Golden: False (True)
b13NONTMR_TESTBENCH_DWC Total tests:     1 Total detected failures:     0 Duration (sec):     31
1479413339.89 : Info (208521): Injects 1 error(s) into device(s)
1479413342.53 : Info (208540): Reading EMR array
1479413342.53 : Info (208543): No frame error detected in device 1.
1479413342.53 : Info (11623): 1 frame error(s) corrected in device 1.
1479413342.53 :     Info (11620):   Corrected error #1 : Single error in frame 0x1F4A at bit 0x0A06.
1479413343.26 : JTAG_READ: 0b00000011000000110000001100001111
1479413343.26 : JTAG STATS - Heartbeat: 9 (True) Mismatch w/Others: False (True)
               Disagree w/Golden: False (True)
b13NONTMR_TESTBENCH_DWC Total tests:     1 Total detected failures:     0 Duration (sec):     35

...

b13NONTMR_TESTBENCH_DWC Total tests:     5 Total detected failures:     4 Duration (sec):   2175
1479415484.18 : Info (208521): Injects 1 error(s) into device(s)
1479415486.83 : Info (208540): Reading EMR array
1479415486.83 : Info (208543): No frame error detected in device 1.
1479415486.95 : Info (11623): 1 frame error(s) corrected in device 1.
1479415486.95 :     Info (11620):   Corrected error #1 : Single error in frame 0x0FF8 at bit 0x00D1.
1479415487.69 : JTAG_READ: 0b00101010001010100010101000101101
1479415487.69 : JTAG STATS - Heartbeat: 295 (True) Mismatch w/Others: True (True)
               Disagree w/Golden: False (True)
1479415487.69 : MISMATCH W/OTHERS FAILURE DETECTED!
1479415488.03 : Info (208540): Reading EMR array
1479415488.03 : Info (208543): No frame error detected in device 1.
b13NONTMR_TESTBENCH_DWC Total tests:     5 Total detected failures:     5 Duration (sec):   2180
```

B.2 Xilinx Fault Injection

The following log is from a fault injection run on the B13 non-TMR design implemented on a Xilinx Kintex 7 FPGA using TMR protected on-chip error detection logic.

```
[INFO] Brigham Young University - Frame Based Scrubber Sensitivity Testing Tool
[INFO] Command: ./b13_nonTMR_scrub/test.py -r
[INFO] Configuration:
          Bit File      - b13_nonTMR_scrub/BF_b13_nonTMR_512x_noReset.bit
          Golden File   - b13_nonTMR_scrub/BF_b13_nonTMR_512x_noReset.data
          Inject Faults - False
1476805704.50 : [INFO] Performing powerCycle
1476805704.50 : [INFO] Power cycle complete
1476805704.50 : [INFO] Initializing Test Run...
1476805704.50 : [INFO] Programming design file b13_nonTMR_scrub/BF_b13_nonTMR_512x_noReset.bit . . .
Finished Full Configuration
1476805707.37 : [INFO] Design loaded, checking jtag state...
1476805707.37 : JTAG READ: 0b00000000000000000000000000000000
1476805707.38 : JTAG STATS - Fail State: 0 (True)
1476805707.38 : [INFO] JTAG reading properly, test is running . . .
Hardware Version: 7Z010 4.3
1476805707.81 : [INFO] Injected 1-bit fault at address 0x00042281, word 3, bit 15
1476805707.81 : JTAG READ: 0b00000000000000000000000000000000
1476805707.81 : JTAG STATS - Fail State: 0 (True)
b13_nonTMR_scrub Total tests:     1 Total detected failures:     0 Duration (sec):      3
1476805707.83 : [INFO] Injected 1-bit fault at address 0x00002707, word 60, bit 24
1476805707.83 : JTAG READ: 0b00000000000000000000000000000000
```

```
1476805707.83 : JTAG STATS - Fail State: 0 (True)
b13_nonTMR_scrub Total tests:     1 Total detected failures:     0 Duration (sec):      3

...

b13_nonTMR_scrub Total tests: 117935 Total detected failures: 117930 Duration (sec): 362959
1477168663.66 : [INFO] Injected 1-bit fault at address 0x00000F20, word 22, bit 2
1477168663.66 : JTAG READ: 0b00000000000000000000000000000111
1477168663.66 : JTAG STATS - Fail State: 1 (True)
1477168663.66 : [ERROR] FAILURE DETECTED!
b13_nonTMR_scrub Total tests: 117935 Total detected failures: 117931 Duration (sec): 362959
1477168663.66 : [INFO] Initializing Test Run...
1477168663.66 : [INFO] Programming design file b13_nonTMR_scrub/BF_b13_nonTMR_512x_noReset.bit . . .
Finished Full Configuration
1477168666.08 : [INFO] Design loaded, checking jtag state...
1477168666.08 : JTAG READ: 0b00000000000000000000000000000000
1477168666.08 : JTAG STATS - Fail State: 0 (True)
1477168666.08 : [INFO] JTAG reading properly, test is running . . .
1477168666.08 : [INFO] Injected 1-bit fault at address 0x00041201, word 52, bit 4
1477168666.08 : JTAG READ: 0b00000000000000000000000000000000
1477168666.08 : JTAG STATS - Fail State: 0 (True)
```

B.3 Altera Neutron Radiation Testing

The following log is from a neutron radiation test beam run on the AES encryption core TMR design implemented on an Altera Stratix V FPGA using DWC protected on-chip error detection logic. The design is programmed onto the FPGA and initialized before the beam run begins. The beam run begins with the beam shutter opening. The beam shutter blocks the accelerated neutron beam from the flight path (see Figure 6.1). When the design is first programmed onto the FPGA with the shutter closed, the log shows that no upsets are detected in the configuration memory. Once the shutter is opened, upsets in configuration memory are reported. Once a failure is detected in the design under test, any additional configuration upsets are recorded and the FPGA is re-programmed with the design under test to reset the design. The reset is performed so that additional failures can be observed.

During a single beam run, multiple failures may occur and be observed. The FPGA design is reset through a full reconfiguration between every observed failure. The beam shutter is not closed between failures. It is closed to end a beam run. Beam runs lasted anywhere from a few minutes to about half a day. Runs would be cut short if the experiment was malfunctioning, or the board setup needed to be changed. Longer runs were terminated to switch the design being tested or to analyze data. Only a single design was tested per board per beam run. It is important to note that it is possible for the beam to be down even though the shutter is open. When this occurs configuration upsets cease until the beam comes back up as long as the shutter is open.

Between most readings of the configuration memory status, multiple detected upsets and corrected upsets were reported. This data suggests that the rate of upsets in configuration memory may have been faster than the correction rate of the scrubber. This means it may be possible that multiple upsets were allowed to accumulate between scrub cycles, which would break TMR. This may explain why less reduction in SEU sensitivity is seen for SEU mitigation in radiation testing than in fault injection.

```
[INFO] Brigham Young University - Altera Sensitivity Testing Tool
[INFO] Command: test.py
[INFO] Configuration:
         SOF File      - aes_128.sof
```

```
          SMH File      - None
          Inject Faults - False
          Weight        - 100.0.0
          Cable         - DE5 Standard [USB-1]
1481427668.94 : [INFO] Performing powerCycle
1481427679.25 : [INFO] Power cycle complete
1481427683.06 : [INFO] Initializing Test Run...
1481427683.06 : [INFO] Programming design file aes_128.sof . . .
1481427712.24 : Warning (292006): Can't contact license server "27001@alteralic2.ee.byu.edu"
                -- this server will be ignored.
1481427712.25 : Info: *******************************************************************
1481427712.25 : Info: Running Quartus Prime Fault Injection Debugger
1481427712.25 :     Info: Version 16.0.0 Build 211 04/27/2016 SJ Standard Edition
1481427712.25 :     Info: Copyright (C) 1991-2016 Altera Corporation. All rights reserved.
1481427712.25 :     Info: Your use of Altera Corporation's design tools, logic functions
1481427712.25 :     Info: and other software and tools, and its AMPP partner logic
1481427712.25 :     Info: functions, and any output files from any of the foregoing
1481427712.25 :     Info: (including device programming or simulation files), and any
1481427712.25 :     Info: associated documentation or information are expressly subject
1481427712.25 :     Info: to the terms and conditions of the Altera Program License
1481427712.25 :     Info: Subscription Agreement, the Altera Quartus Prime License Agreement,
1481427712.25 :     Info: the Altera MegaCore Function License Agreement, or other
1481427712.25 :     Info: applicable license agreement, including, without limitation,
1481427712.25 :     Info: that your use is for the sole purpose of programming logic
1481427712.25 :     Info: devices manufactured by Altera and sold by Altera or its
1481427712.25 :     Info: authorized distributors.  Please refer to the applicable
1481427712.25 :     Info: agreement for further details.
1481427712.25 :     Info: Processing started: Sat Dec 10 20:41:23 2016
1481427712.25 : Info: Command: quartus_fid --cable="DE5 Standard [USB-1]"
                --index=@1=aes_128.sof#pi --weight=100.0.0 -n 0
1481427712.25 : Info (208808): Using programming cable "DE5 Standard [USB-1]"
1481427712.25 : Info (209060): Started Programmer operation at Sat Dec 10 20:41:39 2016
1481427712.25 : Info (209016): Configuring device index 1
1481427712.25 : Info (209017): Device 1 contains JTAG ID code 0x029030DD
1481427712.25 : Info (209007): Configuration succeeded -- 1 device(s) configured
1481427712.25 : Info (209011): Successfully performed operation(s)
1481427712.25 : Info (209061): Ended Programmer operation at Sat Dec 10 20:41:50 2016
1481427712.25 : Info (208551): Program signature into device 1.
1481427712.25 : [INFO] Design loaded, checking jtag state...
1481427713.31 : JTAG_READ: 0b00000111000001110000011100001111
1481427713.31 : JTAG STATS - Heartbeat: 7 (True) Mismatch w/Others: False (True)
                Disagree w/Golden: False (True)
1481427713.34 : [INFO] JTAG reading properly, test is running . . .
1481427714.76 : Info (208540): Reading EMR array
1481427714.76 : Info (208543): No frame error detected in device 1.
1481427716.08 : JTAG_READ: 0b00001111000011110000111100001111
1481427716.08 : JTAG STATS - Heartbeat: 15 (True) Mismatch w/Others: False (True)
                Disagree w/Golden: False (True)

...

1481455465.72 : JTAG STATS - Heartbeat: 117 (True) Mismatch w/Others: False (True)
                Disagree w/Golden: False (True)
1481455469.40 : Info (208540): Reading EMR array
1481455469.40 : Info (208544): 3 frame error(s) detected in device 1.
1481455469.52 : [INFO] Error needing scrub detected.
1481455469.52 :     Info (208545):   Error #1 (0x0174F92171) : Single error in frame 0x2E9F at bit 0x1217.
1481455469.52 : [INFO] Error needing scrub detected.
1481455469.52 :     Info (208545):   Error #2 (0x0067986941) : Single error in frame 0x0CF3 at bit 0x0694.
1481455469.52 : [INFO] Error needing scrub detected.
1481455469.52 :     Info (208545):   Error #3 (0x00B49207E1) : Single error in frame 0x1692 at bit 0x207E.
1481455469.52 : Info (11623): 4 frame error(s) corrected in device 1.
1481455469.52 : [INFO] Error needing scrub NOT FOUND!
1481455469.52 :     Info (11620):   Corrected error #1 : Single error in frame 0x2D84 at bit 0x0312.
1481455469.52 : [INFO] Error needing scrub NOT FOUND!
1481455469.52 :     Info (11620):   Corrected error #2 : Single error in frame 0x0F8F at bit 0x26FC.
1481455469.52 : [INFO] Error needing scrub NOT FOUND!
1481455469.52 :     Info (11620):   Corrected error #3 : Single error in frame 0x13B9 at bit 0x0C48.
```

```
1481455469.52 : [INFO] Error needing scrub NOT FOUND!
1481455469.52 :     Info (11620):   Corrected error #4 : Single error in frame 0x13BA at bit 0x0C48.
1481455473.46 : Info (208525): External scrubbing
1481455474.61 : JTAG_READ: 0b10010000100100001001000000001111
1481455474.61 : JTAG STATS - Heartbeat: 144 (True) Mismatch w/Others: False (True)
                Disagree w/Golden: False (True)

...

1481455483.09 : JTAG STATS - Heartbeat: 169 (True) Mismatch w/Others: False (True)
                Disagree w/Golden: False (True)
1481455486.86 : Info (208540): Reading EMR array
1481455486.86 : Info (208544): 4 frame error(s) detected in device 1.
1481455486.86 : [INFO] Error needing scrub detected.
1481455486.86 :     Info (208545):   Error #1 (0x00ED580D81) : Single error in frame 0x1DAB at bit 0x00D8.
1481455486.86 : [INFO] Error needing scrub detected.
1481455486.86 :     Info (208545):   Error #2 (0x00ED600D81) : Single error in frame 0x1DAC at bit 0x00D8.
1481455486.86 : [INFO] Error needing scrub detected.
1481455486.86 :     Info (208545):   Error #3 (0x01054173D1) : Single error in frame 0x20A8 at bit 0x173D.
1481455486.86 : [INFO] Error needing scrub detected.
1481455486.86 :     Info (208545):   Error #4 (0x007D097301) : Single error in frame 0x0FA1 at bit 0x1730.
1481455491.01 : Info (208525): External scrubbing
1481455492.26 : JTAG_READ: 0b11000100110001001100010000101101
1481455492.26 : JTAG STATS - Heartbeat: 196 (True) Mismatch w/Others: True (True)
                Disagree w/Golden: False (True)
1481455492.26 : MISMATCH W/OTHERS FAILURE DETECTED!
1481455496.03 : Info (208540): Reading EMR array
1481455496.03 : Info (208544): 3 frame error(s) detected in device 1.
1481455496.06 : [INFO] Error needing scrub detected.
1481455496.06 :     Info (10914):   Error #1 (0x00F06001FF) : Uncorrectable multi-bit error in frame 0x1E0C.
1481455496.06 : [INFO] Error needing scrub detected.
1481455496.06 :     Info (208545):   Error #2 (0x00AD325871) : Single error in frame 0x15A6 at bit 0x2587.
1481455496.06 : [INFO] Error needing scrub detected.
1481455496.06 :     Info (208545):   Error #3 (0x00CA4196D1) : Single error in frame 0x1948 at bit 0x196D.
1481455496.06 : Info (11623): 3 frame error(s) corrected in device 1.
1481455496.06 : [INFO] Error needing scrub NOT FOUND!
1481455496.06 :     Info (11620):   Corrected error #1 : Single error in frame 0x1151 at bit 0x26D0.
1481455496.06 : [INFO] Error needing scrub NOT FOUND!
1481455496.06 :     Info (11620):   Corrected error #2 : Single error in frame 0x1AF3 at bit 0x0C34.
1481455496.06 : [INFO] Error needing scrub NOT FOUND!
1481455496.06 :     Info (11620):   Corrected error #3 : Single error in frame 0x1C7B at bit 0x1C58.
1481455496.06 : [INFO] Initializing Test Run...
1481455496.06 : [INFO] Programming design file aes_128.sof . . .
1481455524.73 : Warning (292006): Can't contact license server "27001@alteralic2.ee.byu.edu"
                -- this server will be ignored.
1481455524.73 : Info: *******************************************************************
1481455524.73 : Info: Running Quartus Prime Fault Injection Debugger
1481455524.73 :     Info: Version 16.0.0 Build 211 04/27/2016 SJ Standard Edition
1481455524.73 :     Info: Copyright (C) 1991-2016 Altera Corporation. All rights reserved.
1481455524.73 :     Info: Your use of Altera Corporation's design tools, logic functions
1481455524.73 :     Info: and other software and tools, and its AMPP partner logic
1481455524.73 :     Info: functions, and any output files from any of the foregoing
1481455524.73 :     Info: (including device programming or simulation files), and any
1481455524.73 :     Info: associated documentation or information are expressly subject
1481455524.73 :     Info: to the terms and conditions of the Altera Program License
1481455524.73 :     Info: Subscription Agreement, the Altera Quartus Prime License Agreement,
1481455524.73 :     Info: the Altera MegaCore Function License Agreement, or other
1481455524.73 :     Info: applicable license agreement, including, without limitation,
1481455524.73 :     Info: that your use is for the sole purpose of programming logic
1481455524.73 :     Info: devices manufactured by Altera and sold by Altera or its
1481455524.73 :     Info: authorized distributors.  Please refer to the applicable
1481455524.73 :     Info: agreement for further details.
1481455524.73 :     Info: Processing started: Sun Dec 11 04:24:56 2016
1481455524.73 : Info: Command: quartus_fid --cable="DE5 Standard [USB-1]"
                --index=@1=aes_128.sof#pi --weight=100.0.0 -n 0
1481455524.73 : Info (208808): Using programming cable "DE5 Standard [USB-1]"
1481455524.73 : Info (209060): Started Programmer operation at Sun Dec 11 04:25:11 2016
1481455524.73 : Info (209016): Configuring device index 1
```

```
1481455524.73 : Info (209017): Device 1 contains JTAG ID code 0x029030DD
1481455524.73 : Info (209007): Configuration succeeded -- 1 device(s) configured
1481455524.73 : Info (209011): Successfully performed operation(s)
1481455524.73 : Info (209061): Ended Programmer operation at Sun Dec 11 04:25:23 2016
1481455524.73 : Info (208551): Program signature into device 1.
1481455524.73 : [INFO] Design loaded, checking jtag state...
1481455525.99 : JTAG_READ: 0b00000111000001110000011100001111
1481455525.99 : JTAG STATS - Heartbeat: 7 (True) Mismatch w/Others: False (True)
                Disagree w/Golden: False (True)
1481455525.99 : [INFO] JTAG reading properly, test is running . . .
```

B.4 Xilinx Neutron Radiation Testing

The following log is from a neutron radiation test beam run on the B13 TMR design implemented on an Xilinx Kintex 7 FPGA using TMR protected error detection logic. Note that multiple design failures are observed within a single beam run with a full reconfiguration in between observed failures. Again, multiple configuration upsets are detected and repaired within a single scrub cycle. This indicated that SEU accumulation has occurred, or that a single energetic ionizing particle has upset more than a single bit. SEU accumulation can break redundancy protection. This could explain why less reduction in SEU sensitivity is seen for SEU mitigation in radiation testing than in fault injection.

```
[INFO] Brigham Young University - Frame Based Scrubber Sensitivity Testing Tool
[INFO] Command: test.py
[INFO] Configuration:
          Bit File      - BF_b13_TMR_512x_noReset.bit
          Golden File   - BF_b13_TMR_512x_noReset.data
          Inject Faults - False
1476053797.99 : [INFO] Performing powerCycle
1476053810.40 : [INFO] Power cycle complete
1476053810.41 : [INFO] Initializing Test Run...
1476053810.41 : [INFO] Programming design file BF_b13_TMR_512x_noReset.bit . . .
Finished Full Configuration
1476053813.00 : [INFO] Design loaded, checking jtag state...
1476053813.00 : JTAG READ: 0b00000000000000000000000000000000
1476053813.00 : JTAG STATS - Fail State: 0 (True)
1476053813.00 : [INFO] JTAG reading properly, test is running . . .
Hardware Version: 7Z010 4.3
1476053813.06 : [INFO] Beginning scrub cycle number 1 ...
1476053815.54 : Readback Finished
1476053815.58 : [INFO] Finshed scrub cycle 1, Upsets: 0, Time Elapsed: 2.52
1476053815.58 : JTAG READ: 0b00000000000000000000000000000000
1476053815.58 : JTAG STATS - Fail State: 0 (True)
b13_TMR_scrub Total tests:     1 Total detected failures:     0 Duration (sec):     17
1476053815.59 : [INFO] Beginning scrub cycle number 2 ...
1476053818.07 : Readback Finished
1476053818.11 : [INFO] Finshed scrub cycle 2, Upsets: 0, Time Elapsed: 2.52
1476053818.11 : JTAG READ: 0b00000000000000000000000000000000
1476053818.11 : JTAG STATS - Fail State: 0 (True)
b13_TMR_scrub Total tests:     1 Total detected failures:     0 Duration (sec):     20

...

1476075586.67 : JTAG READ: 0b00000000000000000000000000000000
1476075586.67 : JTAG STATS - Fail State: 0 (True)
b13_TMR_scrub Total tests:     8 Total detected failures:     6 Duration (sec):  21441
1476075586.68 : [INFO] Beginning scrub cycle number 8446 ...
1476075589.16 : Readback Finished
1476075589.20 : [INFO] Upset(s) Detected, Frame: 0x00022B82, Word: 85, Bit(s): 0x08000000 (1)
1476075589.21 : [INFO] Upset(s) Detected, Frame: 0x00022B83, Word: 85, Bit(s): 0x04000000 (1)
1476075589.21 : [INFO] Upset(s) Detected, Frame: 0x00400019, Word: 55, Bit(s): 0x00001000 (1)
```

```
1476075589.22 : [INFO] Upset(s) Detected, Frame: 0x00402892, Word: 55, Bit(s): 0x00200000 (1)
1476075589.22 : [INFO] Upset(s) Detected, Frame: 0x00402894, Word: 55, Bit(s): 0x00600000 (2)
1476075589.23 : [INFO] Upset(s) Detected, Frame: 0x00402895, Word: 55, Bit(s): 0x00200000 (1)
1476075589.23 : [INFO] Upset(s) Detected, Frame: 0x00440612, Word: 62, Bit(s): 0x00000001 (1)
1476075589.23 : [INFO] Finshed scrub cycle 8446, Upsets: 8, Time Elapsed: 2.55
1476075589.23 : JTAG READ: 0b00000000000000000000000000000000
1476075589.23 : JTAG STATS - Fail State: 0 (True)
b13_TMR_scrub Total tests:     8 Total detected failures:     6 Duration (sec):  21443

...

b13_TMR_scrub Total tests:    14 Total detected failures:    12 Duration (sec):  35064
1476089210.35 : [INFO] Beginning scrub cycle number 13813 ...
1476089212.83 : Readback Finished
1476089212.87 : [INFO] Upset(s) Detected, Frame: 0x00021C23, Word: 36, Bit(s): 0x00000100 (1)
1476089212.88 : [INFO] Upset(s) Detected, Frame: 0x0002260C, Word: 71, Bit(s): 0x00000010 (1)
1476089212.88 : [INFO] Upset(s) Detected, Frame: 0x00401B0A, Word: 84, Bit(s): 0x00100000 (1)
1476089212.89 : [INFO] Upset(s) Detected, Frame: 0x00401B0B, Word: 84, Bit(s): 0x00200000 (1)
1476089212.89 : [INFO] Finshed scrub cycle 13813, Upsets: 4, Time Elapsed: 2.54
1476089212.89 : JTAG READ: 0b00000000000000000000000000000000
1476089212.89 : JTAG STATS - Fail State: 0 (True)
b13_TMR_scrub Total tests:    14 Total detected failures:    12 Duration (sec):  35067
1476089212.89 : [INFO] Beginning scrub cycle number 13814 ...
1476089215.37 : Readback Finished
1476089215.42 : [INFO] Upset(s) Detected, Frame: 0x00020822, Word: 67, Bit(s): 0x01000000 (1)
1476089215.42 : [INFO] Finshed scrub cycle 13814, Upsets: 1, Time Elapsed: 2.53
1476089215.42 : JTAG READ: 0b00000000000000000000000000000111
1476089215.42 : JTAG STATS - Fail State: 1 (True)
1476089215.42 : [ERROR] FAILURE DETECTED!
1476089215.42 : [INFO] Beginning scrub cycle number 13815 ...
1476089217.90 : Readback Finished
1476089217.95 : [INFO] Upset(s) Detected, Frame: 0x00000505, Word: 59, Bit(s): 0x00000010 (1)
1476089217.95 : [INFO] Upset(s) Detected, Frame: 0x00040D0D, Word: 53, Bit(s): 0x00000040 (1)
1476089217.96 : [INFO] Upset(s) Detected, Frame: 0x00040D0E, Word: 53, Bit(s): 0x00000100 (1)
1476089217.96 : [INFO] Upset(s) Detected, Frame: 0x00040D0F, Word: 53, Bit(s): 0x00000080 (1)
1476089217.96 : [INFO] Upset(s) Detected, Frame: 0x00401CA1, Word: 46, Bit(s): 0x00000020 (1)
1476089217.97 : [INFO] Finshed scrub cycle 13815, Upsets: 5, Time Elapsed: 2.55

b13_TMR_scrub Total tests:    14 Total detected failures:    13 Duration (sec):  35072
1476089217.97 : [INFO] Initializing Test Run...
1476089217.97 : [INFO] Programming design file BF_b13_TMR_512x_noReset.bit . . .
Finished Full Configuration
1476089220.57 : [INFO] Design loaded, checking jtag state...
1476089220.57 : JTAG READ: 0b00000000000000000000000000000000
1476089220.57 : JTAG STATS - Fail State: 0 (True)
1476089220.57 : [INFO] JTAG reading properly, test is running . . .
1476089220.57 : [INFO] Beginning scrub cycle number 13816 ...
1476089223.05 : Readback Finished
1476089223.09 : [INFO] Upset(s) Detected, Frame: 0x00020B90, Word: 74, Bit(s): 0x00000080 (1)
1476089223.10 : [INFO] Upset(s) Detected, Frame: 0x00020B91, Word: 74, Bit(s): 0x00000040 (1)
1476089223.10 : [INFO] Upset(s) Detected, Frame: 0x00402498, Word: 70, Bit(s): 0x00003000 (2)
1476089223.11 : [INFO] Upset(s) Detected, Frame: 0x00402499, Word: 70, Bit(s): 0x00003000 (2)
1476089223.11 : [INFO] Upset(s) Detected, Frame: 0x00420A94, Word: 69, Bit(s): 0x00010000 (1)
1476089223.12 : [INFO] Upset(s) Detected, Frame: 0x00420EA2, Word: 48, Bit(s): 0x00000100 (1)
1476089223.12 : [INFO] Finshed scrub cycle 13816, Upsets: 8, Time Elapsed: 2.55
1476089223.12 : JTAG READ: 0b00000000000000000000000000000000
1476089223.12 : JTAG STATS - Fail State: 0 (True)
b13_TMR_scrub Total tests:    15 Total detected failures:    13 Duration (sec):  35077
1476089223.12 : [INFO] Beginning scrub cycle number 13817 ...
1476089225.60 : Readback Finished
1476089225.64 : [INFO] Upset(s) Detected, Frame: 0x00001120, Word: 10, Bit(s): 0x00000002 (1)
1476089225.65 : [INFO] Upset(s) Detected, Frame: 0x00021A89, Word: 100, Bit(s): 0x00200000 (1)
1476089225.65 : [INFO] Finshed scrub cycle 13817, Upsets: 2, Time Elapsed: 2.53
1476089225.65 : JTAG READ: 0b00000000000000000000000000000000
1476089225.65 : JTAG STATS - Fail State: 0 (True)
b13_TMR_scrub Total tests:    15 Total detected failures:    13 Duration (sec):  35080
```