SEU Mitigation and Validation of the LEON3 Soft Processor Using Triple Modular Redundancy for Space Processing

Michael Wirthlin, Andrew Keller, Chase McCloskey and Parker Ridd
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Deptartment of Electrical and Computer Engineering
Brigham Young University
Provo, UT 84606, USA

David S. Lee[1] and Jeffrey Draper[2]
[1]Sandia National Laboratories
Albuquerque, NM 87123, USA
Information Sciences Institute
[2]Ming Hsieh Department of Electrical Engineering
University of Southern California
Marina del Ray, CA, USA

ABSTRACT

Programmable processors are an essential component in most satellite payload electronics and handle a variety of functions including command handling and data processing. There is growing interest in implementing programmable processors within satellites on commercial FPGAs because of their reconfigurability, large logic density, and I/O bandwidth. Commercial FPGAs, however, are sensitive to ionizing radiation and systems developed for space must implement single-event upset mitigation to operate reliably. This paper investigates the improvements in reliability of a LEON3 soft processor operating on a SRAM-based FPGA when using triple-modular redundancy and other processor-specific mitigation techniques. The improvements in reliability provided by these techniques are validated with both fault injection and heavy ion radiation tests. The fault injection experiments indicate an improvement of 51× and the radiation testing results demonstrate an average improvement of 10×. Orbit failure rate estimations were computed and suggest that the TMR LEON3 processor has a mean-time to failure of over 70 years in a geosynchronous orbit.

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance; B.7.2 [**Design Aids**]: Automated TMR, Manual Mitigation

Keywords

FPGA; Soft-core Processor; LEON3; TMR; Fault-Tolerance

1. INTRODUCTION

Modern satellite systems depend on reliable, radiation-hardened (rad-hard) processors to perform various mission-critical operations such as payload processing, command handling, and satellite control. These rad-hard processors, however, are extremely expensive and are often based on architectures that are much older and slower than commercially available processors. A soft processor can be an attractive alternative to a rad-hard processor by providing processor-specific customization, the ability to add custom reliability techniques, and the ability to provide customized FPGA logic [1]. Further, modern FPGAs provide a significant amount of configurable logic and access to a large amount of serial I/O bandwidth.

Any design operating in an SRAM-based FPGA, including soft processors, are susceptible to ionizing radiation [2]. The presence of high energy protons, heavy ions, and galactic cosmic rays in the space orbits cause a number of problems for electronics, including FPGAs. This radiation can induce a number of negative effects including upsets in the internal state of the device. These upsets, known as single-event upsets or SEUs, can cause several problems in FPGA-based systems. First, SEUs can corrupt the configuration memory of the device causing the design configured on the device to operate incorrectly. Second, the internal state of the design's flip-flops or block memories may be corrupted resulting in data corruption. Third, the internal state of custom hardware blocks within the FPGA may be corrupted (PLLs, clock managers, DSPs, etc.).

Fortunately, a number of mitigation techniques have been developed to address these concerns. One of the most common ways of applying structural mitigation is using triple-modular redundancy or TMR [3]. TMR provides masking of single errors by triplicating the circuit and providing majority voters (see Figure 1). As long as two of the three copies of the circuit are operating correctly, the system will tolerate a failure in any of the three copies. Systems that employ TMR must also provide a mechanism for repairing the failures and resynchronizing the state of the three circuit copies. This repair and resynchronization is usually done through a technique called configuration scrubbing [4, 5]. TMR has been shown to provide significant improvements in FPGA reliability in harsh radiation environments [6].

While TMR along with scrubbing have been shown to improve reliability, applying TMR to the entire system may not be the most effective SEU mitigation approach. Complex systems with a variety of resources and different de-

FPGA'16, February 21-23, 2016, Monterey, CA, USA

DOI: http://dx.doi.org/10.1145/2847263.2847278

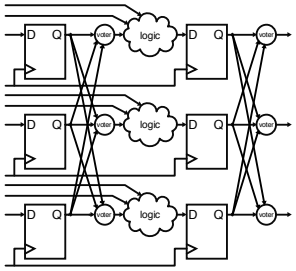

Figure 1: Triple Modular Redundancy (TMR).

sign approaches may respond more reliably to a variety of SEU mitigation approaches. This work investigates hybrid mitigation approaches that are targeted specifically for soft processors operating within an FPGA. In particular, this paper will describe a processor SEU mitigation technique that involves TMR, custom memory mitigation, and custom control structures.

This paper describes a SEU mitigation strategy applied to the LEON3 soft processor core operating on the Xilinx Kintex-7 FPGA. This mitigation strategy is validated using both fault injection and radiation testing with heavy ions. The results of this validation effort demonstrate that the techniques used to improve SEU reliability are improved by a factor of 51× using fault injection and 10× in radiation testing. Estimation of orbit failure rates suggest that at a minimum, these techniques will increase the mean-time to failure of this processor from 1.4 years to 76 years.

This paper will begin by reviewing related efforts in developing reliable soft processors within FPGAs. Next, the LEON3 soft processor core will be described along with the configuration used in this testing and the software used during the validation of the mitigation techniques. Next, the mitigation approaches applied to the LEON3 are described along with the increase in size and reduction in clock speed. The approach for validating the mitigation techniques are presented using fault injection and radiation testing. The impact of these techniques are compared with the non-mitigated processor and suggestions for future work are provided.

2. RELATED WORK

A number of projects have investigated methods for improving the reliability of soft processors operating in commercial SRAM FPGAs. A variety of both structural and temporal redundancy are used to improve soft processor response to single-event upsets. The LEON2 soft processor core was mapped to the Virtex II architecture and tested using fault injection [7]. This work compared three different version of the processor: the standard processor, the processor with TMR, and a "fault tolerant" architectural variation. A custom 16-bit "configurable fault-tolerant processor (CFTP)" was created for an FPGA-based computing satellite and uses TMR and scrubbing to mitigate against configuration upsets [8]. This platform was designed for use in the U.S. Naval Academy's MidSTAR-l satellites and launched in March of 2003.

A NIOS-II soft processor [9] was implemented on an Altera Cyclone-II device and implements time redundancy to address the effects of soft errors and fault injection. Experiments were completed to validate the benefits of this technique. A general approach for lock-step processing with dual soft processors has been proposed [10, 11] which uses traditional software reliability techniques such as lockstep, checkpointing, and rollback. This work was tested using the LEON processor with three variants: the default LEON processor, a TMR LEON processor, and the proposed dual-lockstep core with software fault tolerant techniques.

A triple modular small Xilinx PicoBlaze soft processor was tested on the Xilinx Virtex 5 architecture using a tile-based approach and resynchronization through partial reconfiguration [12]. A duplicated LEON3 was implemented with a special bus monitor to detect processor failures at run time [13]. The reliability of a TMR version of the Microblaze processor was tested with fault injection [14] and a related effort tested the reliability of the microblaze processor operating on the radiation hardened by design (RHBD) Virtex V5QV FPGA [15].

3. LEON3 PROCESSOR ARCHITECTURE

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) [16, 17]. It features a 7-stage integer pipeline with a Harvard architecture. The LEON3 is a popular option for processing in space environments. First, the LEON3 core utilizes a very small portion of available resources on a commercial FPGA such as the Xilinx Kintex-7 XC7K325T. The remaining resources are available for use in SEU mitigation techniques and for additional IP. Second, 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. Third, the LEON3 is well documented and supported by a large user group. Fourth, the processor is independent of any FPGA architecture and has been ported to several different FPGA architectures. Fifth, the processor is available as a radiation-hardened custom circuit to facilitate migration from a soft implementation to a higher-performance, more reliable custom implementation [18].

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. A 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 2 reflects the final configuration of both the LEON3 processor core and connected peripherals via the on-chip bus.

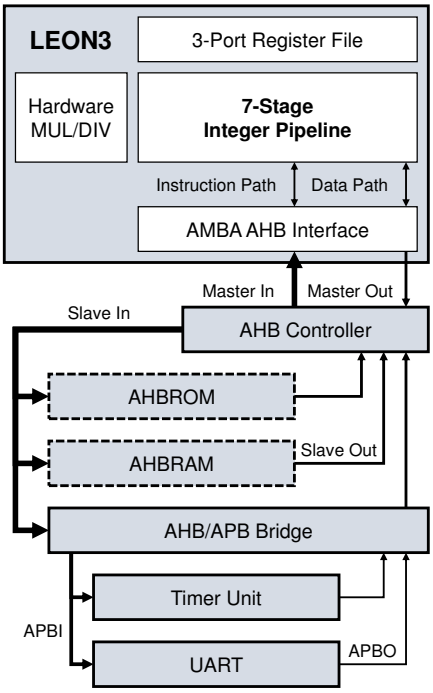

Figure 2: LEON3 System Architecture Under Test

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. The benchmark is composed of common instructions surveyed in system-level software such as operating systems and compilers [19]. This program assists in validation of functional correctness under testing when compared against another processor core running the same benchmark program in parallel. Once Dhrystone is loaded into RAM the LEON3 begins execution at the decompressed address. In this experiment Dhrystone is run for 10,000 iterations in a continuous loop.

The LEON3 processor begins the boot process by accessing a read only memory (AHBROM) composed of onboard BRAMs that contains a boot loader and the compressed application program.The boot loader sets the system clock and the baud-rate for UART and then decompresses the benchmark from AHBROM to the AHBRAM peripheral.The Dhrystone C code was cross-compiled for the SPARC V8 architecture and integrated into the LEON3 boot loader using GRTools-20150121 [20].

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

4. LEON3 SEU MITIGATION STRATEGY

The SEU mitigation strategy used in this work involves the combination of several different techniques. The primary mitigation technique used is triple modular redundancy (TMR). Automated tools are used to triplicate the LEON3 netlist and insert voters strategically throughout the design. In addition to TMR, the internal memories of the processor are manually replaced with hand-designed memories that include scrubbing to prevent the accumulation of errors within the memory. These techniques are augmented with configuration scrubbing which occurs in the background to prevent the accumulation of configuration errors. Each of these mitigation techniques used will be described in detail below.

4.1 Automated TMR

Triple Modular Redundancy uses redundant hardware to mask SEUs by triplicating the original module and inserting majority synchronizer voters (see Figure 1). If at least two of the circuit copies are correct, the output of the system will be correct as well. To avoid single point failures, voters themselves are often triplicated as well [21]. The granularity of TMR affects the amount of SEUs a design can tolerate before failure; 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 increased voter insertion [22]. There are a number of automated CAD tools that are capable of applying TMR to an FPGA design such as: Xilinx XTMR tool [23], Mentor Graphics Precision Rad-Tolerant Synthesis [24], and an open source tool called BL-TMR [25]. For this test, TMR will be leveraged as much as possible to increase reliability by applying full TMR at a fine granularity to the final LEON3 system configuration of Figure 2.

The BL-TMR open-source tool was used to perform fine granularity TMR of the LEON3 system architecture. It achieved this by manipulating the Xilinx ISE 14.7 generated EDIF netlist of the design allowing for Xilinx 7-series primitives, (e.g., LUT6, FDRE, MUXF8, etc.), to be triplicated with majority voters placed throughout the design. This tool places voters in feedback paths so as to resynchronize the triplicated processor when repaired through configuration scrubbing [21]. The resulting netlist is then instantiated in the test design harness for validation.

All components of the LEON3 system were triplicated except for the AHB memories as indicated by their dashed outline in Figure 2. The global clock was triplicated using three independent clock buffers enabling each LEON3 TMR domain to have its own unique clock. All ports of the LEON3 system module were triplicated and are utilized in the test design harness for validation.

4.2 Internal Memory Mitigation

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 multple upsets in memory words. This problem is especially prevalent on memories whose contents do not change very often (such as read only memories). Affected memories in Xilinx 7-series FPGAs include: BRAMS, LUTRAMs, SRLs, DRP registers, etc. In order to prevent accumulation of upsets that may invalidate TMR and other

memory reliability techniques, such as ECC, some form of memory scrubbing is needed [26]. The memories in this processor are manually modified to implement both fault masking and memory scrubbing.

Although the 7 series BRAMs offer ECC support [27], this ECC is not used in this processor for several reasons. First, the ECC is only available when the BRAM is configured in 64-bit mode. The memory used in the LEON3 is configured with a data width that is smaller than 64 bits. Second, the use of a single BRAM with ECC introduces single-point failures in the memory. For example, there are several signals such as the "write enable" that could be corrupted by a single SEU and completely break the operation of the single, ECC-enabled BRAM. While ECC can mask a single bit error when reading from BRAM, it does not actually repair the error within the BRAM. Therefore, ECC is not able to handle the accumulation of SEUs in BRAM unless the corrected data is written back into the memory. To ensure there are no single-point failures in the memory architecture and to prevent accumulation of upsets, triplicated BRAMs with self-scrubbing will be used.

By providing memory scrubbing, memories affected by SEUs will not accumulate upsets and TMR can be used to mask single errors that occur in the memories. The two memories that were protected in this processor are a single-port read only BRAM configuration for the AHBROM peripheral and a single-port BRAM read/write configuration for the AHBRAM peripheral. In each case scrubbing logic is added that takes advantage of the unused second BRAM memory port (see Figure 3). This scrubber "cleans" the memory contents by alternating between a read and write operation through each address of the memory. The complete contents of the BRAM are scrubbed every 400 μs. Data reads from the triplicated memories are voted upon to obtain a corrected value for the current address. This value is then written back to the memory. Scrubbing is disabled in the AHBRAM while it is being written to by the processor to avoid memory conflicts.

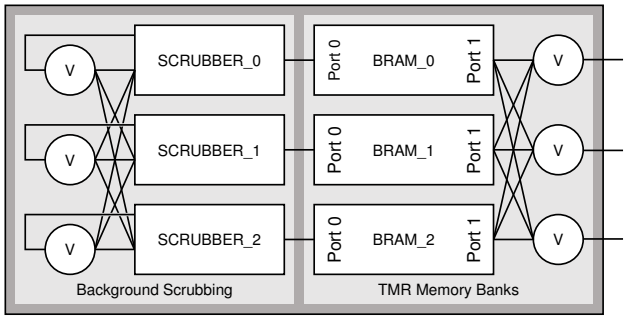

Figure 3: Memory Mitigation with TMR and Memory Scrubbing.

These scrubbing memories were created manually in HDL using inferred BRAMs. The BL-TMR tool merged the netlist of the manually mitigated scrubbing memories with the triplicated netlist of the LEON3 system.

4.3 Configuration Scrubbing

An important component of the mitigation strategy used in this system is configuration scrubbing [4, 5]. Configuration scrubbing involves the rapid and continuous writing of the configuration memory to repair upsets. The reliability of a system can be significantly improved when both configuration scrubbing (repair) and TMR (fault masking) are used together. Without configuration scrubbing, the benefits of TMR are limited (especially for long missions).

Configuration scrubbing is usually performed external to the FPGA and does not impact the design of the mitigated LEON3 processor. A JTAG configuration scrubbing mechanism is used for this experiment and will be described in more detail in Section 6.

4.4 Testing Infrastructure

Two different versions of the LEON3 processor were developed. The first is the unmitigated version which implements the LEON3 processor without any modification. This unmitigated version will serve as a reference to identify baseline reliability. The second is a mitigated version and implements the TMR and memory mitigation described above. The improvements in reliability will be compared against the baseline unmitigated processor.

Both the unmitigated and mitigated LEON3 processors were tested under fault injection and a heavy ion radiation beam. In order to validate the LEON3 processor under these tests a mechanism must be put in place to identify when the LEON3 processor fails. One common approach is to provide a "golden" copy of the design and compare the "golden" copy of the processor with the design under test or "DUT". While this is an effective mechanism, it is difficult to synchronize two separate systems making it a cumbersome strategy to implement.

The strategy used for detecting failures in this experiment is to instance *two* copies of the LEON3 system inside a test design harness and provide an internal cycle-by-cycle comparison of the processors from *within* the FPGA. Fault tolerant detection circuitry is added to detect when the two processors disagree (see Figure 4). When a failure occurs in one of the two processors, the faulty processor will not match the non-faulty processor and the detection circuitry will identify the error. Once an error is identified a triplicated register is set and remains set so external monitoring methods may record the failure.

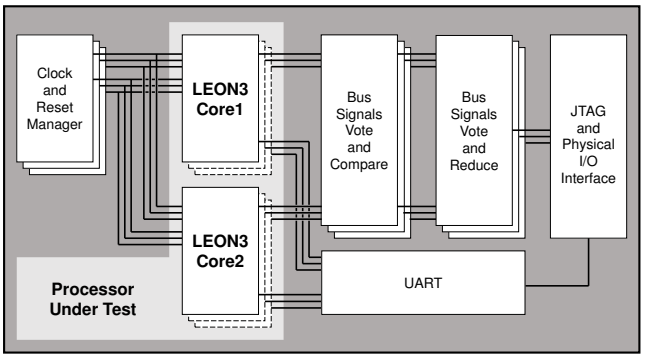

Figure 4: Internal Testing Infrastructure.

The following signals were used to compare the processor execution state and monitor the health of the test design harness:

- Global system clock heartbeat and a heartbeat for each of the triplicated clock domains
- Triplicated disagreement signals set only when processors disagree
- Triplicated failure detection signals asserted if any disagreement occurred
- Triplicated activity signals for each processor core indicating that the bus signals being compared are changing over time (i.e., that the processor is active).
- UART output of the Dhrystone (2.1) benchmark for monitoring correct behavior.

Each bit of the bus signal state was compared between the two processors using triplicated comparison circuitry giving a total of 104 bits of comparison status. Triplicated reduction circuitry reduced these 104 bits to a single triplicated bit. If ever two or more copies of this triplicated bit go high (i.e., the bus signals of the processors do not agree across two or more copies of the comparison logic), an additional register is set and remains set to *catch* the disagreement so that the failure may be recorded externally from the chip.

Two methods of off chip status retrieval were incorporated into this test. First, the status signals were made available via JTAG using an instanced Xilinx Boundary Scan primitive (BSCAN). This allows the status to be queried as part of the configuration scrubbing or fault injection, enabling dynamic control of the test (e.g., automated recording and full device reconfiguration upon failure for fault injection). Second, the triplicated status signals were tied to physical I/O ports and monitored via Xilinx FPGA Mezzanine Card XM105 Debug Cards using an additional Xilinx KC705 Evaluation Board. A Xilinx Virtual I/O Module was instanced on this out-of-beam monitor device and Xilinx ChipScope was used to allow real time monitoring of the DUT during radiation testing. Figure 8 shows the physical configuration of the DUT with the monitoring board. Once off chip the triplicated status signals may be voted upon to determine the correct status of the design under test (i.e., one of the three failures signals may erroneously be set due to an SEU in the comparison logic and should not be considered as a LEON3 system failure).

4.5 Design Implementation Results

Both the unmitigated and mitigated designs were mapped to the Xilinx XC7K325T device on the KC705 evaluation board. The FPGA utilization of both designs is summarized in Table 1. The mitigated version of the LEON3 uses 3.3× more slices than the unmitigated version. The mitigated dual LEON version uses roughly 30% of the FPGA resources suggesting that up to six mitigated LEON3 processors could fit within the XC7K325T device. The internal FPGA layout of both designs is shown in Figure 5.

Although both processors operate at 50 MHz during the tests, the unmitigated processor has a higher maximum clock rate than that of the mitigated processor. The maximum operating frequency of the two designs is shown in Table 2. To understand the effect of the detection logic on maximum operating speed, both designs are implemented in two forms: with and without the detection circuitry. In both cases, the unmitigated circuit operates faster than the mitigated circuit. With detection circuitry added, the mitigated circuit is much slower than that of the unmitigated circuit (.55×) suggesting that the critical path is within the detection circuitry.

Resource Utilization	Testing Overhead	LEON3 Core 1	LEON3 Core 2	Total	Device NonTMR/TMR
Slices	1753	1383	1410	4546	50950
(TMR)	1960	6567	6767	15294	8.9%/30.0%
Slice Reg	2726	1950	1950	6626	407600
(TMR)	2726	6165	6165	15056	1.6%/3.7%
LUTS	3324	4077	4069	11470	203800
(TMR)	3265	18046	18051	39362	5.6%/19.3%
LUTRAM	1	15	15	31	64000
(TMR)	1	45	45	91	.048%/.142%
BRAM	0	50	50	100	445
(TMR)	0	150	150	300	22.5%/67.4%
DSP48E1	0	1	1	2	840
(TMR)	0	3	3	6	.238%/.714%
BUFG	4	0	0	4	32
(TMR)	4	0	0	4	12.5%/12.5%

Table 1: Dual-LEON3 Design Utilization

	NonTMR	TMR
No Detection	164 MHz (6.10 ns)	140 Mhz (7.15 ns)
Detection	132 MHz (7.56 ns)	73.7 MHz (13.56 ns)

Table 2: Post-PAR Timing Summary

5. FAULT INJECTION

A useful way of learning more about the SEU sensitivity of an FPGA design and to understand the benefits of a mitigation technique is to apply artificial fault injection within the configuration memory [29]. Fault injection involves *intentionally* inserting corrupt data into the configuration memory by partially configuring FPGA frames with configuration data in which one or more configuration bits are opposite from their original values (faults are usually injected one at a time). After the faulty configuration data has been applied, the behavior of the circuit is monitored to detect deviations from the expected output. If the circuit deviates from its expected value, then the corresponding upset configuration bit is labeled as *sensitive*. Those configuration bits that do not impact the circuit behavior are labeled *insensitive*. Fault injection can be used to measure the relative sensitivity of various designs to upsets in the configuration memory.

Fault injection has its limitations and does not model all of the negative behavior of FPGAs operating in the presence of ionizing radiation (flip-flops or upsets in the proprietary internal state of an FPGA). As such, a number of failure modes seen in radiation testing will not appear in fault injection. 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. The goals of fault injection for this work are first, estimate the configuration sensitivity of both the unmitigated LEON3 processor and the mitigated processor. second, validate the ability of the test infrastructure to detect processor errors, and validate the testing infrastructure and software before radiation testing.

5.1 Fault Injection Setup

The fault injection tool used for this project is based on custom tool called the "JTAG Configuration Manager" or JCM. The JCM is a Linux-based embedded system that provides the ability to generate high-speed JTAG sequences

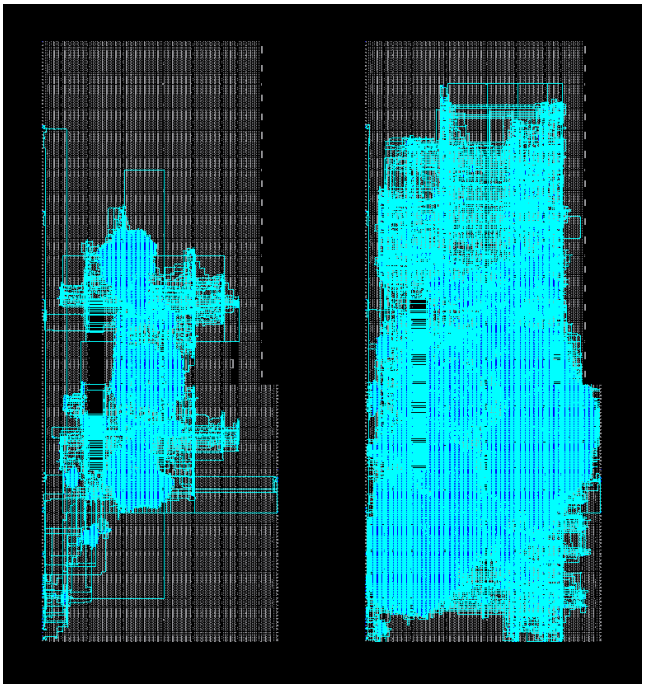

Figure 5: FPGA Layout of Unmitigated Design (left) and the Mitigated design (right)

for FPGA configuration, readback, and configuration scrubbing. The JCM is capable of configuring an individual frame through JTAG in under 85 μs. The injection of faults can be controlled programmatically by the host Linux system to customize the fault injection campaign to the goals of the experiment. A picture of the JCM fault injection system and the KC705 board is shown in Figure 6.

The fault injection campaign applied to the LEON3 was designed to emulate the random nature of configuration upsets that are expected in a radiation beam. The fault injector is programmed to upset a single configuration bit at a random location and to continue inserting upsets until the processor system fails. The specific steps of this fault injection campaign are as follows:

1. Randomly select a configuration bit and invert the value of the configuration bit through partial configuration.
2. Wait 1 ms to allow the effects of the upset to propagate through the system.
3. Monitor the behavior of the two processor comparator circuit through the JTAG BSCAN interface.
    - If the processor is unaffected by the upset, repair the configuration bit and proceed.
    - If the processor behavior is affected by the upset, reconfigure the FPGA to restore the correct operating state.

The JCM injects faults at an average rate of 120 faults per second.

5.2 Fault Injection Results

The fault injection system was applied to both LEON3 designs: the unmitigated design and the SEU mitigated design.

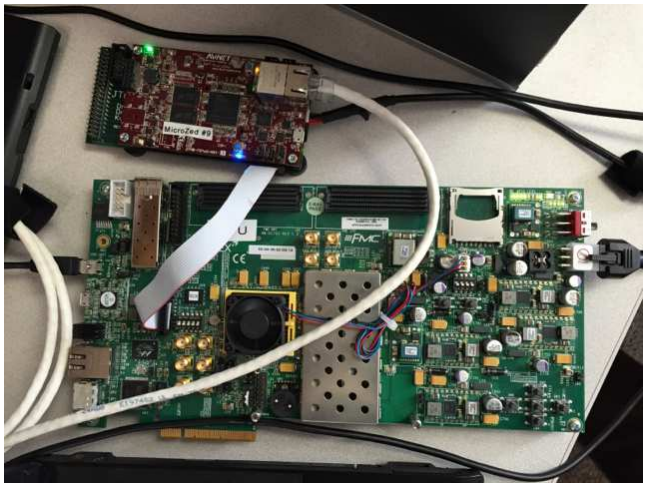

Figure 6: JTAG Configuration Manager. The board at the bottom is the Xilinx KC705 which contains the Kintex DUT. The small board on the top is the Linux-based JTAG Configuration Manager. A standard 14-wire JTAG cable connects the JCM to the KC705.

The results from this fault injection campaign are summarized in Table 3. This table lists the number of faults injected into each design (n) and the number of observed failures on the design (k). From these results the mean-upsets to failure (MUTF) can be estimated as n/k.

	Unmitigated	Mitigated
Faults Injected (n)	1,831,859	29,443,885
Observed Failures (k)	6,501	2,037
MUTF	282	14,455
Sensitivity (95% Conf. Interval)	.355% (.346%,.363%)	.00692% (.00662%,.00722%)
Est. Sensitive Bits	240,539	4,689
Improvement	1.0	51.3

Table 3: Fault Injection Results.

The sensitivity of the design or the estimated percentage of configuration bits with each FPGA design that are sensitive to upsets, is estimated by using the maximum likelihood estimator, $\hat{r}$, of the Binomial distribution:

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

The standard deviation of the maximum likelihood estimator is:

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

The standard deviation of the estimator can be used to determine the 95% confidence interval bounds of the sensitivity estimate. The number of total sensitive bits in the design can be estimated by multiplying the sensitivity estimate, $\hat{r}$, by the total number of configuration bits (67,779,264) in

block 0 of the configuration bitstream (i.e., the configuration bits associated with the logic and routing).

These results suggest that the mitigated LEON3 design is 51.3× *less* sensitive to upsets than the unmitigated design in spite of the fact that the mitigated design is 3.4× larger than the unmitigated design. These results indicate that the mitigation techniques described in Section 4 significantly reduce the sensitivity of the LEON3 processor to upsets in the configuration memory.

In spite of these mitigation techniques, however, there are a number 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. There are several known single points of failure including the clock network and external I/O. In addition to these known single points of failure, it is possible that some configuration bits in the routing may corrupt more than one net associated with different TMR domains as suggested by [30]. Future work will investigate the cause of these remaining single-points of failure to further improve the reliability.

6. HEAVY ION RADIATION TEST

The results from the fault injection campaign suggest that the mitigation approaches are working and the configuration sensitivity is significantly reduced. The next step in the validation of the mitigated LEON3 processor is to test the processor with a high energy radiation beam. Radiation testing provides a number of advantages over fault injection. First, a radiation beam can upset *any* internal state of the FPGA including that states not tested by fault injection (user flip-flops, block memory, and internal proprietary FPGA state). Second, radiation testing will induce other faults such as single-event transients, multi-cell upsets, and possibly single-event latch-up. Because of these additional failure modes, it is expected that the improvements of the mitigation design will be *lower* with radiation testing than with fault injection.

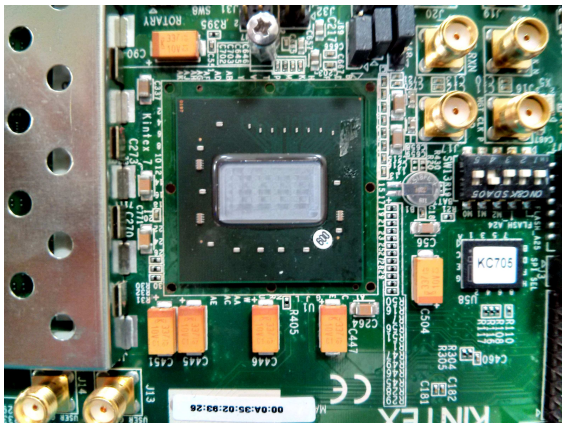

Figure 7: Delidded and Thinned XC7K325T Die.

6.1 Radiation Test Setup

A radiation test was performed on both the mitigated and non-mitigated LEON3 designs with heavy ions at the Texas A&M K500 Cyclotron in August of 2015. A sample XC7K325T device was prepared for the test by removing the lid and thinning the substrate to allow the ions to penetrate the substrate and reach the active region of silicon (see Figure 7). During radiation testing many configuration bits (and other state) will be upset and configuration scrubbing must be employed to rapidly restore the configuration memory to its proper state. The JTAG Configuration Manager described in Section 5 was used to provide active, high-speed configuration scrubbing during the test. Figure 8 shows the KC705 test board mounted on a base plate along with the configuration scrubber, power monitor, and external I/O connector. Next to the base plate is a second KC705 board that receives the I/O signals for review by the test operator. The base plate mounted in front of the beam cap is shown in Figure 9.

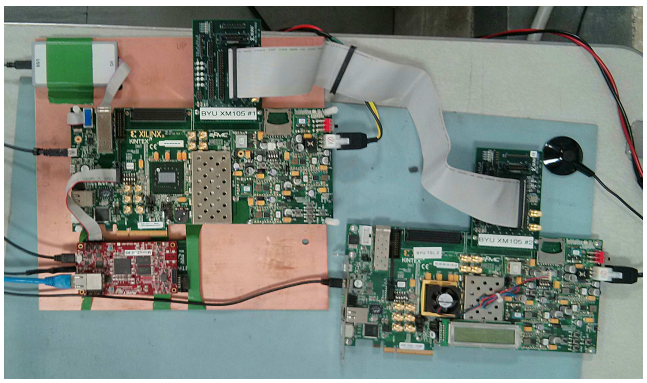

Figure 8: Experiment setup for radiation testing. The radiation test setup is on the left and the remote monitor is on the right.

The organization of the radiation test is very similar to the fault injection testing. The FPGA is configured with either the mitigated or non-mitigated LEON3 design and executes the Dhrystone benchmark. The correctness of the processor execution is monitored through the external I/O signals as well as the standard UART output of the processor. There are a few minor modifications to the radiation test. First, the configuration scrubber is enabled to actively repair configuration upsets when the beam is turned on. Second, a secondary processor correctness interface is added to detect processor failure in the event that the JTAG interface is disabled by radiation. This interface involves several digital I/O signals that are sent to a remote data acquisition board for manually monitoring the processor status.

The procedure for each test is as follows. First, the FPGA is configured and the scrubber initialized to make sure the system is running properly without the radiation beam. Second, the shutter for the radiation beam is removed to enable the radiation beam and induce faults in the circuit. Once the beam is enabled, upsets will occur throughout the FPGA and the configuration scrubber is repairing and logging these upsets. At some point, the radiation will cause the processor to fail. When this failure is detected by the operator, the operator closes the beam shutter and records the beam fluence that accumulated during the beam run. The accumulated fluence is the primary data associated with each run (fluence to failure) and multiple runs are performed to provide an average fluence to failure with greater confidence.

To obtain an accurate estimate of the failure rate of the

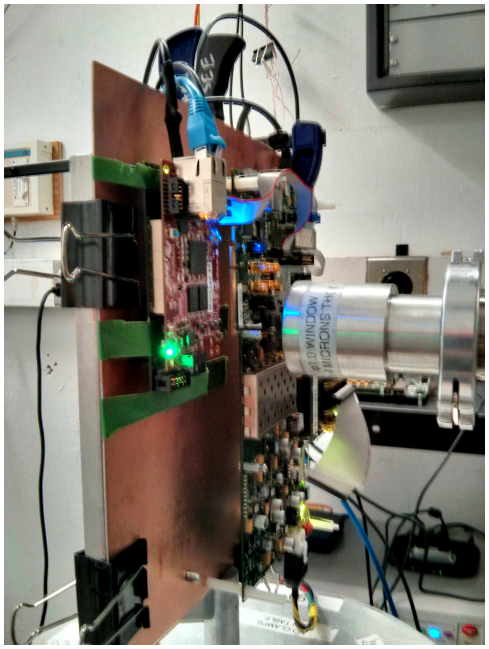

Figure 9: Heavy Ion Radiation Test Setup in Beam.

LEON3 processors in a space environment, the heavy ion radiation test must be performed at multiple energies as the space environment is a complex environment with a range of particle flux and energies (see Figure 11). The parameter used in this type of testing is Linear Energy Transfer or LET which describes the amount of energy that an ionizing particle transfers to the material per unit distance ($MeV\ cm^2/mg$). Different values of LET can be obtained by changing the ion used in the beam and inserting degrader material into the beam path.

6.2 Radiation Test Results

Both the unmitigated and mitigated designs were tested at six different LET values as shown in Table 4. As described above, multiple runs were performed at each LET to measure the accumulated beam fluence to failure. The average cross section at each LET was determined by dividing the number of failures observed by the total fluence at the given LET. As shown in the fourth column, the mitigated designs tolerated on average ten times as much fluence as the unmitigated design suggesting that the mitigation techniques provided significant, measurable improvements in radiation tolerance over the unmitigated design.

LET (Ion) (MeV-cm^2/mg)	Unmitigated (cm^2/proc.)	Mitigated (cm^2/proc.)	Improv.
3.4 (Ne)	2.40E-5	3.65E-4	15.2
4.7 (Ne)	2.76E-5	2.58E-4	9.4
6 (Ne)	5.07E-5	5.38E-4	10.6
9 (Ne)	5.75E-5	5.59E-4	9.7
11.9 (Ar)	6.94E-5	5.43E-4	7.8
18.5 (Ar)	3.13E-5	2.69E-4	8.6

Table 4: Measured SEU Sensitive Cross Section.

Unlike fault injection where thousands of runs can be performed automatically in a relatively short amount of time (see k in Table 3), it is very labor intensive to perform a run with radiation testing and only a dozen or so runs can be performed at each LET. The low number of runs produces much larger confidence intervals than the fault injection runs.

6.3 Radiation Test Challenges

Although the radiation test successfully demonstrated an order of magnitude improvement in radiation tolerance for the mitigated design, there were a number of significant problems that occurred at the test. First, the scrubbing hardware was incorrectly configured to the wrong FPGA device and was only scrubbing the first third of the configuration memory. With much of the FPGA not being scrubbed, it is likely that the TMR mitigation scheme failed due to the accumulation of upsets (i.e., more than one TMR domain failing). Second, the scrubbing hardware ran relatively slowly in comparison to the configuration upset rate. In a deployed system, the scrub rate is set to operate at several orders of magnitude greater than the individual configuration upset rate. In radiation testing, however, this is not possible since the upset rate is many orders of magnitude higher than the rate in a space environment. For this experiment, the FPGA experienced an average of 5.11 configuration upsets per second in the first third of configuration memory that was scrubbed. The configuration scrubber operated at 2.88 seconds per scrub allowing about 14.72 configuration upsets to accumulate before completing a scrub cycle. Future radiation tests will be conducted to address these issues and hopefully achieve improved reliability results.

7. ORBIT FAILURE RATE ESTIMATES

The radiation test results were used to estimate the failure rate in a near-earth interplanetary/geosynchronous orbit under "solar minimum" solar conditions (maximum cosmic-ray conditions). This estimate is made by first estimating a cross-section curve (or failure probability curve) as a function of LET. It is customary to use the Weibull distribution to model cross-section curves. The parameters of the Weibull distribution are estimated by using the data samples at each LET from the radiation test. Figure 10 shows a Weibull cross-section curve (in green) based on the size LET test samples. The test samples are shown with blue circles and the corresponding error bars.

Once cross-section curve estimates are created, the orbit error rate is estimated using a tool called CREME-96 [31]. This tool begins by estimating the particle flux of the given orbit based on previous orbit measurements – the GEO orbit particle flux is shown in Figure 11. This tool adjusts the radiation environment seen by the spacecraft due to shielding and then convolves the design cross section (Figure 10) with this modified environment (as a function of LET).

The estimated failure rate and the corresponding mean-time to failure (MTTF) of both designs is shown in Table 5. The estimated failure rate of the mitigated LEON3 processor is roughly 76 years. This estimate suggests that the implemented mitigation methods provide sufficient reliability for many space applications. It is important to emphasize that the estimated orbital failure rates are not exact and that much of the uncertainty is based on the parameters chosen for the Weibull cross section curves More radiation testing

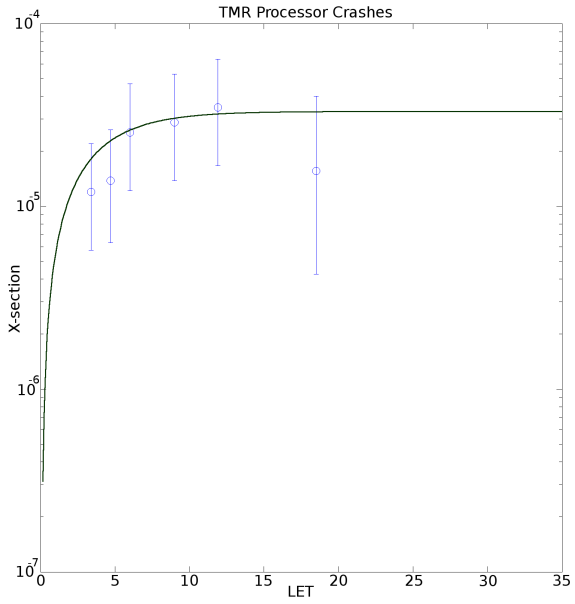

Figure 10: Weibull Cross Section Curve for Mitigated Processor.

is needed at other LET values to increase the accuracy of the Weibull curve fit.

Design	Failure Rate (λ) (failures/processor/s)	MTTF (days/years)
Unmitigated	2.77E-8	501/1.4
Mitigated	4.15E-10	27,889/76

Table 5: Estimated Failure Rate of LEON3 in the GEO Orbit.

8. CONCLUSION

This work investigated the improvements in reliability of the soft core LEON3 processor operating on a commercial Xilinx Kintex 7 FPGA. TMR, internal memory scrubbing, and configuration scrubbing were all used to mitigate against single-event upsets that occur in the configuration memory, user flip-flops, and other FPGA state. Fault injection tests indicated a 51× reduction in SEU cross section, and rough orbit failure rate estimates suggest a MTTF of over 76 years in space. All of these results suggest that the collection of mitigation techniques used to improve the LEON3 reliability were successful and that the mitigated LEON3 processor may be reliable enough to consider for use in space.

Although improvements in reliability were seen in both fault injection and radiation testing, the amount of improvement was not as high as expected. The results suggest that additional SEU mitigation may provide even higher improvements in reliability. A number of single-points of failure still exist in the design (clocking and some I/O) and not all memory structures of the processor were improved with

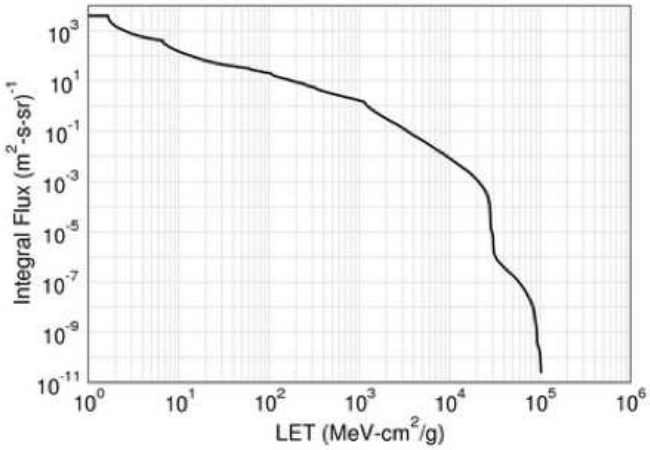

Figure 11: GEO Orbit LET Spectra Expected at Device After Shielding.

scrubbing. In addition, significant improvements in the radiation testing procedure will likely provide additional improvements in the radiation validation. These obvious opportunities for additional improvements in reliability suggest that the reliability estimates in this work can be significantly improved.

To integrate a processor in a complex system, additional features and processor architecture within the LEON3 must be tested. Future work will investigate the reliability of a complex LEON3 system with caches, memory controllers, and other essential I/O peripherals. With this architecture support, complex operating systems like Linux can be used and complex software systems can be deployed. The success of this future work will facilitate the adoption of soft core processors operating in commercial FPGAs in future space systems.

9. ACKNOWLEDGEMENTS

This work was supported by the I/UCRC Program of the National Science Foundation under Grant No. 1265957.

10. REFERENCES

[1] J.G. Tong, ID.L. Anderson, and M.AS. Khalid. Soft-core processors for embedded systems. In *Microelectronics, 2006. ICM '06. International Conference on*, pages 170–173, Dec 2006.

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

[3] F. Lima Kastensmidt, L. Sterpone, L. Carro, and M. Sonza Reorda. On the optimal design of triple modular redundancy logic for SRAM-based FPGAs. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 2*, DATE '05, pages 1290–1295, Washington, DC, USA, 2005. IEEE Computer Society.

[4] Carl Carmichael, Michael Caffrey, and Anthony Salazar. Correcting single-event upsets through Virtex

partial configuration. Technical report, Xilinx Corporation, June 1, 2000. XAPP216 (v1.0).

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

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

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

[8] C.A. Hulme, H.H. Loomis, A.A. Ross, and Rong Yuan. Configurable fault-tolerant processor (CFTP) for spacecraft onboard processing. In *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, volume 4, pages 2269–2276 Vol.4, March 2004.

[9] J. Perez Acle, M.S. Reorda, and M. Violante. Implementing a safe embedded computing system in SRAM-based FPGAs using IP cores: A case study based on the Altera NIOS-II soft processor. In *Circuits and Systems (LASCAS), 2011 IEEE Second Latin American Symposium on*, pages 1–5, Feb 2011.

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

[11] M. Violante, C. Meinhardt, R. Reis, and M.S. Reorda. A low-cost solution for deploying processor cores in harsh environments. *Industrial Electronics, IEEE Transactions on*, 58(7):2617–2626, July 2011.

[12] C. Gauer, B.J. LaMeres, and D. Racek. Spatial avoidance of hardware faults using FPGA partial reconfiguration of tile-based soft processors. In *Aerospace Conference, 2010 IEEE*, pages 1–11, March 2010.

[13] Frederico Ferlini, Felipe A. da Silva, E.A. Bezerra, and Djones V. Lettnin. Non-intrusive fault tolerance in soft processors through circuit duplication. In *Test Workshop (LATW), 2012 13th Latin American*, pages 1–6, April 2012.

[14] Gregory Miller, Carl Carmichael, and Gary Swift. Mitigation, design flow and troubleshooting a soft processor in a complex FPGA. In *Military and Aerospace Programmable Logic Devices (MAPLD) Workshop*, 2008.

[15] Gregory Miller, Carl Carmichael, Gary Swift, Mike Pratt, and Gregory R. Allen. Preliminary analysis of a soft-core processor in a Rad Hard by Design Field Programmable Gate Array. In *Military and Aerospace Programmable Logic Devices (MAPLD) Workshop*, 2009.

[16] IEEE standard for a 32-bit microprocessor architecture. *IEEE Std 1754-1994*, pages 1–, 1995.

[17] Aeroflex gaisler LEON3 processor. http://www.gaisler.com/index.php/products/processors/leon3.

[18] Luo Pei and Zhang Jian. A high reliable SOC on-board computer based on Leon3. In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, volume 1, pages 360–363, May 2012.

[19] Michael R. Gardiner. An evaluation of soft processors as a reliable computing platform. Master's thesis, Brigham Young University, 2015.

[20] GRTools. http://www.gaisler.com/index.php/downloads/grtools.

[21] Jonathan M. Johnson and Michael J. Wirthlin. 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, pages 249–258, New York, NY, USA, 2010. ACM.

[22] M. Niknahad, O. Sander, and J. Becker. A study on fine granular fault tolerance methodologies for FPGAs. In *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on*, pages 1–5, June 2011.

[23] Brendan Bridgford, Carl Carmichael, and Chen Wei Tseng. Single-event upset mitigation selection guide. Technical Report 1, Xilinx Corporation, 2008. Xilinx Application Note XAPP987.

[24] R. Do. The details of triple modular redundancy: An automated mitigation method of I/O signals. In *The prooceedings of the Military and Aerospace Programmable Logic Devices*, 2011.

[25] BL-TMR and BYU Edif Tools. http://sourceforge.net/projects/byuediftools/.

[26] N. Rollins, M. Fuller, and M.J. Wirthlin. A comparison of fault-tolerant memories in SRAM-based FPGAs. In *Aerospace Conference, 2010 IEEE*, pages 1–12, March 2010.

[27] Xilinx Coproration. *7 Series FPGAs Memory Resources: User Guide*. UG473 (v1.11), November 12, 2014.

[28] Daniel P. Siewiorek and Robert S. Swarz. *Reliable Computer Systems*. A. K. Peters, 1998.

[29] F. Lima, C. Carmichael, J. Fabula, R. Padovani, and R. Reis. A fault injection analysis of Virtex FPGA TMR design methodology. In *Proceedings of the 6th European Conference on Radiation and its Effects on Components and Sysemts (RADECS 2001)*, 2001.

[30] M.S. Reorda, L. Sterpone, and M. Violante. Multiple errors produced by single upsets in FPGA configuration memory: a possible solution. In *Test Symposium, 2005. European*, pages 136–141, May 2005.

[31] A.J. Tylka, J.H. Adams, P.R. Boberg, B. Brownstein, W.F. Dietrich, E.O. Flueckiger, E.L. Petersen, M.A. Shea, D.F. Smart, and E.C. Smith. Creme96: A revision of the cosmic ray effects on micro-electronics code. *Nuclear Science, IEEE Transactions on*, 44(6):2150–2160, Dec 1997.