Benefits of Complementary SEU Mitigation for the LEON3 Soft Processor on SRAM-Based FPGAs

Andrew M. Keller, *Student Member, IEEE*, and Michael J. Wirthlin, *Senior Member, IEEE*

***Abstract*— A variety of mitigation techniques have been demonstrated to reduce the sensitivity of FPGA designs to soft errors. Without mitigation, SEUs can cause failure by altering the logic, routing, and state of a design operating on an SRAM-based FPGA. Various combinations of SEU mitigation and repair techniques are applied to the LEON3 soft-core processor to study the effects and complementary nature of each technique. This work focuses on Triple modular redundancy (TMR), configuration memory (CRAM) scrubbing, and internal block memory (BRAM) scrubbing. All mitigation methods demonstrate some improvement in both fault injection and neutron radiation testing. Results in this paper show complementary SEU mitigation techniques working together to improve fault-tolerance. The results also suggest that fault injection can be a good way to estimate the cross section of a design before going to a radiation test. TMR with CRAM scrubbing demonstrates a 27× improvement whereas TMR with both CRAM and BRAM scrubbing demonstrates approximately a 50× improvement.**

***Index Terms*— BRAM scrubbing, configuration scrubbing, Feedback TMR, FPGA, LEON3, neutron beam test, reliability, SEU mitigation, soft processors, TMR.**

I. Introduction

A SOFT-CORE processor is a hardware description of a microprocessor that can be implemented on a field programmable gate array (FPGA) or other device. Several soft core processors are available commercially and as open-source IP cores. These processors are often distributed in a hardware description language and can be customized for specific applications [1]. Implemented on an FPGA, these processors can be modified remotely after they have been deployed.

Soft-core processors on commercial SRAM-based FPGAs could be used in radiation environments, provided that they are sufficiently immune to output errors. An FPGA design, a soft-core processor in this case, operates on the fabric of the FPGA and is configured into the programming data of the FPGA. This data is susceptible to single event upsets (SEUs). Since the processor is configured into the FPGA, an SEU may alter the logic, routing, or state of the processor core and cause failure. Fortunately, SEU mitigation techniques have been shown to improve the reliability of soft-core processors on SRAM-based FPGAs [2].

SEU mitigation techniques are targeted towards specific SEU failure modes of SRAM-based FPGAs [3], [4]. 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 routing resources, lookup tables, control signals, and the contents of smaller memory modules (e.g. shift-registers, LUTRAMs, flip-flops) are stored in CRAM bits. Data associated with the contents of larger block memory modules are stored in BRAM bits. The contents of on-chip memory modules may be altered by the design during operation. Because of the varied accessibility of programming data, complementary SEU mitigation techniques must be combined in order to provide more complete protection from SEUs.

A previous experiment [5] applied triple modular redundancy (TMR), internal block memory (BRAM) scrubbing and configuration memory (CRAM) scrubbing to the LEON3 soft-core processor to improve its fault-tolerance. Only the fully mitigated design was evaluated (i.e. with all three SEU mitigation techniques). These techniques work together to improve the fault-tolerance of the processor. To evaluate the effects and complementary nature of each technique, the work in this paper tests various combinations of these SEU mitigation techniques on the LEON3 soft processor. Five combinations of SEU mitigation techniques are tested: unmitigated, TMR only, TMR with BRAM scrubbing, TMR with CRAM scrubbing, and TMR with both BRAM and CRAM scrubbing.

Both fault injection and neutron radiation testing are used to evaluate each variation of the LEON3 processor. Each variation demonstrates an improvement in fault-tolerance. As more techniques are combined together, the amount of improvement increases. For some variations, more improvement is seen in radiation testing than in fault injection. The results in this paper demonstrate complementary SEU mitigation techniques working together to improve fault-tolerance. The results also suggest that fault injection can be a good way to estimate the cross section of a design before going to a radiation test. Both fault injection and neutron radiation testing demonstrate a 27× improvement for TMR with CRAM scrubbing, and approximately a 50× improvement for TMR with both CRAM and BRAM scrubbing.

II. Fault-Tolerant LEON3 Soft Processor

The LEON3 is an open-source 32-bit soft-core processor that is technology independent and can be implemented on

Manuscript received July 9, 2016; revised September 29, 2016, November 10, 2016, November 25, 2016, and November 28, 2016; accepted November 29, 2016. Date of publication December 1, 2016; date of current version February 28, 2017. This work was supported by the I/UCRC Program of the National Science Foundation under grant 1265957.

The authors are with the NSF Center for High-Performance Reconfigurable Computing (CHREC), Brigham Young University, Provo, UT 84602 USA (e-mail: andrewmkeller@byu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2016.2635028

SRAM-based FPGAs. The processor is well documented and has a strong user group for support [6]. For this paper, 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. This paper uses the same configuration of the LEON3 system as [5].

Because of its size and complexity, the LEON3 is a good representative design for demonstrating the benefits of complementary SEU mitigation and repair techniques on SRAM-based FPGAs. In harsh radiation environments, it also exhibits many of the SEU failure modes typical to SRAM-based FPGAs. Complementary SEU mitigation and repair techniques are applied to the LEON3 soft processor to improve its fault-tolerance. The benefits demonstrated in this design are applicable to other SRAM-based FPGA designs.

SEUs in an FPGAs' 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 memory module data. SEUs in routing resources may cause nets to disconnect, short to power or ground, or bridge with other nets in the design. SEUs in lookup tables and control signals can corrupt logic and alter the functionality of primitive blocks used by the design. SEUs in memory module data can corrupt the state of a design [3], [4].

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. Separate scrubbing mechanisms must be used to scrub BRAM bits and CRAM bits because of their accessibility. Specific forms of TMR, BRAM scrubbing, and CRAM scrubbing are applied to the LEON3 for this study.

A. Triple Modular Redundancy

TMR protects the designs from errors in the routing resources, lookup tables, control signals, and internal state. It does so by voting between three redundant copies of the design to mask SEUs. So long as two or more of the redundant copies are functioning correctly, up to and including the voter, the output of the module will be correct as well.

The form of TMR used in this experiment is called fine-grain feedback TMR. As illustrated in Figure 1, fine-grain feedback TMR replicates the design at the lowest level possible (e.g., at the primitive level) and partitions the redundant copies by placing majority voters in the feedback structures of the circuit. Feedback TMR places voters in the feedback paths of the circuit. Voters themselves are also triplicated. Each of these aspects decrease the SEU sensitivity of TMR [7], [8].

TMR alone cannot protect against SEU accumulation across more than one redundant copy within the same partition. This includes SEUs in FPGA configuration memory as well as on-chip memory modules, such as block RAMs or shift-registers.

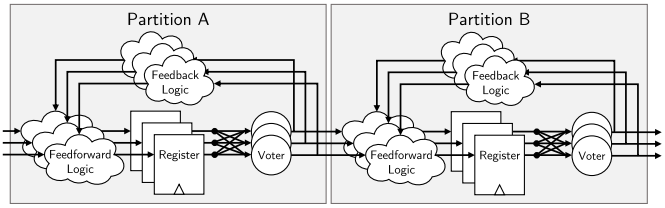

Fig. 1. Fine-grain Feedback TMR

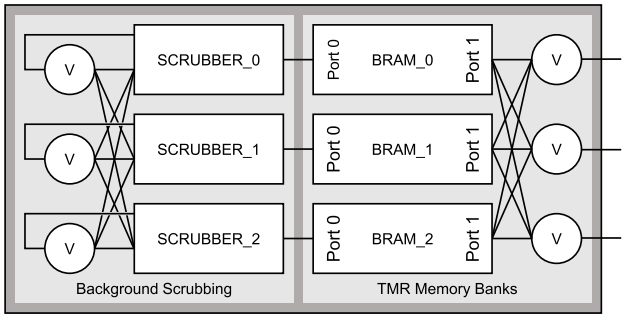

Fig. 2. Internal BRAM Scrubbing

In order to prevent this kind of failure, an SEU repair mechanism, such as scrubbing, must be combined with TMR [9]. Also, without coupling TMR with reliability-oriented place and route tools, a single SEU in routing could affect more than one redundant copy, breaking TMR [10].

B. Internal Block Memory Scrubbing

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. BRAM bits are grouped together into blocks 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.

Only memory that does not change value very often will benefit from scrubbing. Memory that changes often overwrites any previous SEUs, preventing accumulation. No scrubbing is applied to the register file of the LEON3 processor because the values of the registers change frequently. In this implementation of the LEON3 processor, if the caches were enable enabled, they too would not benefit from being scrubbed for similar reasons [11].

In the experiment of this paper, Internal memory scrubbing of the ROM and RAM modules of the LEON3 is performed by continuously writing the correct value of memory to each address. For TMR, the correct value is determined by voting between the redundant copies. The ROM and RAM modules of the LEON3 consume only a single port of the dual-port BRAM modules used by the processor. This leaves the other port available for scrubbing. Figure 2 shows how one port of the BRAM module is used by the LEON3, while the other port is used for scrubbing. The scrubbing logic loops through each address in memory. This logic is also protected by TMR [5], [12].

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 [12].

BRAM scrubbing only prevents SEU accumulation in the BRAM modules to which it is applied. This protection does not prevent an incorrect value from being written to the memory. If an incorrect value is written to more than one redundant copy of a TMR protected memory, it cannot be corrected by scrubbing.

C. Configuration Memory Scrubbing

CRAM scrubbing repairs the circuit when SEUs occur in configuration bits that do not change value during design operation, (e.g., routing and logic bits). It protects against breaking TMR by preventing the accumulation of SEUs. In order to prevent the accumulation of SEUs, the scrub cycle period must be faster than the upset rate.

For this paper, external readback configuration scrubbing was performed on the LEON3 design. This type of configuration scrubbing compares the current configuration of the FPGA against a golden copy to determine where SEUs have occurred. When an SEU is detected, it is repaired by writing the correct value from the golden copy back to the FPGA via partial reconfiguration. Configuration scrubbing was performed over JTAG using a custom high-speed JTAG controller. External scrubbing has been shown to greatly reduce the sensitive cross section of a design [13].

III. Design SEU Test Infrastructure

Using both fault injection and radiation testing, several variations of the LEON3 processor and associated mitigation techniques are tested for SEU fault tolerance. The same testing infrastructure is used for all design variations and in both fault injection and radiation testing. Using the same testing infrastructure for all design versions minimizes the variation in SEU response due to differences in the testing infrastructure. This section will summarize the SEU testing infrastructure used on the LEON3 for both fault injection and radiation testing.

The primary purpose of the SEU test infrastructure is to *detect* when the design under test, the LEON3 soft processor in this case, operates incorrectly or deviates in any way from its specified behavior. There are a variety of different methods for identifying FPGA design failures including: comparing the design output with a predetermined test vector, comparing the final output of the design processor with a predetermiend output value, and operating the design in lockstep with an identical "golden" design that is not subject to failures. The mechanism for identifying SEU induced faults within the LEON3 processor involves the use of two lockstep LEON3 processor systems running in parallel *on the same FPGA device* as shown in Figure 3.

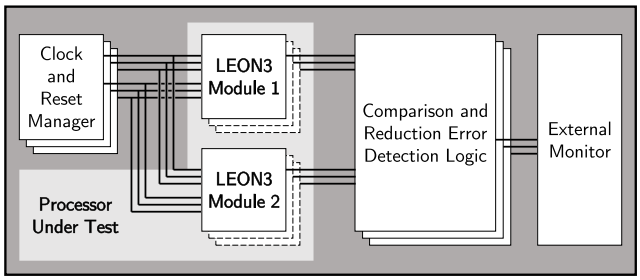

Fig. 3. Single-Chip SEU Testing Infrastructure.

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. A wide, multi-bit comparison circuit simultaneously checks each bus signal and a reduction circuit combines the results of all comparator circuits into a single fault error signal. Because an error may be detected for only a single clock cycle, the signal is latched to ensure that the fault is identified. As suggested in Figure 3, this comparison and reduction logic, along with additional control signals and I/O resources, resides on the FPGA with the two duplicate copies of the LEON3 module. The copies are independent from each other; each has its own memory submodules, peripherals, and system bus.

Placing two copies of the LEON3 in the same FPGA for fault detection has a number of advantages and limitations. The advantages of performing this testing on a single device is that the testing infrastructure is relatively simple and only a single FPGA board is needed to perform the experiment. Because both processors are located on the FPGA, the checking logic can run at a faster rate allowing the processors to run faster than if they are split between two FPGAs. The disadvantages of this approach is that the fault detection logic is inside the single device and subject to SEU induced failures. The on-chip error detection also consumes logic thus limiting the size of the design that can be tested.

To reduce SEU induced failures within the self checking logic, full TMR with frequent voting was applied to the self-checking logic. This TMR checking circuitry was used in all of the LEON3 design variations. The TMR comparison and reduction logic is costly in terms of logic resources because of the large number of signals being compared. Table I shows the number of resources dedicated to various components of the final designs; the percent of total device resources consumed is shown in parenthesis.

A failure is defined as anytime the bus signals of the two LEON3 systems do not agree with each other. Because caching is disabled, bus activity is elevated and better reflects the functional state of the processor and connected peripherals. To keep the design active the LEON3s execute the Dhrystone 2.1 benchmark in a continuous loop. Doing so allows SEU induced errors to propagate throughout most of the system and be detected as a failure. If an SEU is masked

[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).

TABLE I
LEON3 DESIGN VARIATION UTILIZATION

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

by TMR or repaired before corrupting the state of the design, it will not cause a failure. Note also that if an SEU causes both LEON3 systems to fail in exactly the same way, the failure will go undetected. It is assumed, since the processors are independent, that this scenario is unlikely.

For external sampling of the failure state, the triplicated failure status registers are attached to a JTAG Boundary Scan. Two or more of the triplicated registers set high indicate that a failure has occurred. Once a failure has occurred a full device reconfiguration must take place in order to reset these registers and begin the test again. Both fault injection and radiation testing use these signals to collect data and conduct the experiment.

IV. LEON3 SEU MITIGATION VARIATIONS

The SEU mitigation techniques described in Section II all work together to provide significant improvements in the overall SEU sensitivity of the LEON3 processor in radiation environments. The impact of each technique on the overall improvement, however, is not clear. This work seeks to understand the impact of each SEU mitigation technique by creating a variety of LEON3 processor systems with different combinations of these SEU mitigation techniques. Each variation of SEU mitigation techniques will be tested using both fault injection and radiation testing to measure the improvement in overall SEU sensitivity. The relative improvement of each variation will help identify the impact of each individual mitigation technique. The following five versions of the LEON3 were created: Unmitigated, TMR Only, TMR with BRAM Scrubbing, TMR with CRAM Scrubbing, and TMR with CRAM and BRAM Scrubbing. This section will summarize each of these five LEON3 variations.

Although the LEON3 processor design varies with each experiment, all of the design variations are organized with the same on-chip testing infrastructure shown in Figure 3. In each design variation, the same control signals and error detection logic is used to identify faults.

A. Unmitigated

The "Unmitigated" design is the baseline reference LEON3 processor that provides no spatial SEU mitigation techniques. Without any additional internal mitigation hardware, this processor is the smallest of the different processor variations (see Table I). It is expected that this design will be the most sensitive to ionizing radiation and that all other design variations will provide greater SEU immunity. All other design variations will be compared to the SEU response of this baseline unmitigated design.

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 testing infrastructure[2]. In some instances, an SEU mitigation approach consisting of only configuration scrubbing may provide sufficient mitigation for designs without any internal feedback [14].

B. TMR Only

A second version of the LEON3 processor was created that applies fine-grain feedback TMR to the processor logic. This triplicated logic and voting will mask logic and routing errors that are induced by single-event upsets. As expected, this version of the processor is significantly larger than the unmitigated baseline version (3.9× the slice count as seen in Table I).

In this "TMR Only" variation, no BRAM or CRAM scrubbing is performed during system testing. Without a scrubbing repair mechanism, an accumulation of SEUs will cause the TMR design to fail. Conventional reliability modeling suggests that applying TMR without repair results in a system with a *lower* mean-time to failure than an unmitigated design [15]. However, a reduction in SEU design sensitivity is expected because the additional voters used in feedback TMR provide significant isolation between TMR domains.

C. TMR and BRAM Scrubbing

TMR and scrubbing work together to mask errors and repair errors. This design variation adds BRAM scrubbing to the TMR LEON3 processor described above. BRAMs are used in the LEON3 design to implement ROM and RAM modules for the instructions and data memory of the processor. Although the BRAM memories are triplicated with TMR, scrubbing will prevent the accumulation of errors within the memory.

BRAM scrubbing requires additional logic resources to implement the scrubbing logic and memory voting (see Figure 2). The TMR and BRAM scrubbing circuit is 4.8× larger than the unmitigated baseline design (see Table I). It is expected that TMR with BRAM scrubbing will provide greater SEU protection than TMR alone because the memories are able to mask single-bit errors *and* do not accumulate multiple errors.

D. TMR w/CRAM Scrubbing

In this design variation, the "TMR only" variation is augmented with configuration scrubbing (CRAM Scrubbing).

[2]In retrospect, it would have been interesting to test the Unmitigated design with *and* without CRAM scrubbing to better understand the failure behavior of the comparison and reduction logic.

TMR masks upsets that cause errors in the routing and logic bits of CRAM; CRAM scrubbing repairs SEUs as they occur to prevent the buildup of errors within the processor logic. If buildup occurs, TMR may be unable to mask future SEUs, which would allow them to cause errors. Integrating TMR with a repair mechanism (CRAM scrubbing in this case) provides a significant decrease in SEU sensitivity over TMR alone.

Unlike BRAM scrubbing, CRAM scrubbing does not require additional logic resources in the LEON3 processor. The same design used for the "TMR Only" variation is used for this design variation, but in this case the TMR design is supplemented with an external CRAM scrubbing mechanism (see Section II).

E. TMR, Internal Memory Scrubbing, and Configuration Scrubbing

The final design variation integrates all of the SEU mitigation techniques described in Section II (TMR, internal BRAM scrubbing and CRAM scrubbing). Together these techniques should yield the greatest decrease in SEU sensitivity because they are complementary and cover the greatest number of SEU failure modes.

V. Fault Injection

All five variations of the LEON3 processor were tested using *fault injection* or the artificial injection of upsets within the configuration memory. Injecting faults into the configuration memory through fault injection provides a mechanism for emulating the faults within the configuration memory that would otherwise occur through ionizing radiation. Compared to neutron radiation testing, fault injection is relatively inexpensive and is able to collect data at a relatively high speed. The limitations of fault injection include the inability to inject faults in all FPGA state, including BRAMs, and the inability to inject faults with the same characteristics that may occur within a radiation test beam or in an actual space orbit (i.e., multi-cell upsets and with random inter arrival times) [16].

A. Approach

To emulate the behavior that would occur within a radiation test, a *random* fault injection campaign was conducted for this experiment. 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 LEON3 processor to fail when upset. After a randomly selected CRAM bit is upset, the behavior of the processors are carefully monitored to see if either of the two LEON3 processors deviates from the other. 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.

This random fault injection campaign follows the flow shown in Figure 4. Injected faults and detected failures are recorded and used for analysis. This test procedure follows

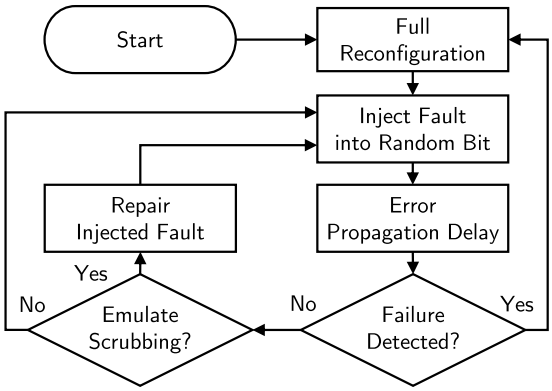

Fig. 4. Fault Injection Flowchart

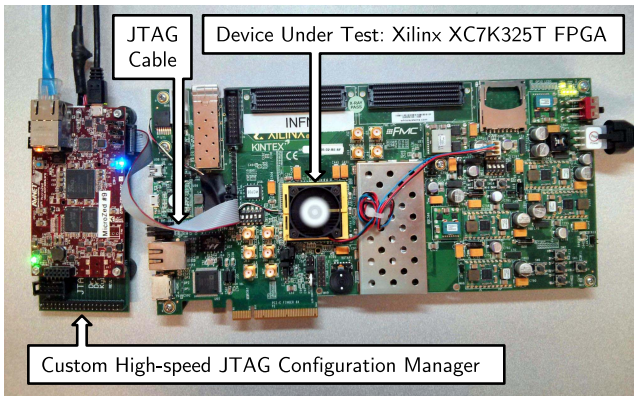

Fig. 5. A custom high-speed JTAG controller (left) is used to perform automated fault injection testing on a Xilinx Kintex 7 FPGA (right).

the methodology of [17]. Since some of the variations of the LEON3 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 are allowed to accumulate, emulating SEU behavior without CRAM scrubbing.

The process of inserting faults into the configuration memory is performed using the JTAG configuration interface of the FPGA. A high-speed and programmable JTAG controller is attached to the interface and emulates random fault injection via partial reconfiguration. It takes 6.9 ms on average to inject a fault. This includes reading an entire frame of configuration data (404 bytes), inverting the selected bit, and writing the data back to the FPGA. To allow errors to propagate, a 1 ms delay is added between fault injection and checking for a failure. At this rate 126.9 faults are injected per second. Figure 5 demonstrates the Kintex 705 evaluation board and the JTAG controller used for the fault injection experiment.

B. 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

TABLE II
FAULT INJECTION RESULTS

Variation	1	2	3	4	5
Description	Unmitigated	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×)

each CRAM bit, the sensitivity of the design is *estimated* by dividing the number of observed failures (k) by the number of faults injected (n). This is equivalent to the maximum likelihood estimator of the binomial distribution. 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:

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

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 sensitive bits) require much more fault injection to get good results. During this experiment, 43 million configuration bits were upset among all five of the LEON3 design variations to obtain useful data (see Table III).

The number of sensitive bits in each design variation is estimated by multiplying the total number of CRAM bits in the device (72,823,424) by the sensitivity of the design. For example, the sensitivity of the unmitigated design is measured at.355%, which suggests that one in every 282 configuration bits will cause the design to fail when upset. Therefore, the total number of sensitive bits in the design is estimated at 258,440. The 95% confidence interval of this estimated is also provided in Table III.

C. Results

The estimated design sensitivity of each of the five LEON3 design variations is shown in Table II. This sensitivity is for the entire design, which includes two processors and testing logic. Because the testing logic is included, only an overestimate of the sensitivity for a single processor can be determined by dividing the sensitivity in half. The same is true for the cross section of a single processor in neutron radiation testing.

The improvement in design sensitivity, over the baseline design, is also shown. 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 (#2) demonstrates a 4× improvement over 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[3]. 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.

VI. NEUTRON RADIATION TEST

All five LEON3 design variations were tested with a neutron radiation beam at the Los Alamos Neutron Science Center (LANSCE) in November of 2015. This wide spectrum neutron beam is commonly used for testing of integrated circuits to estimate circuit sensitivity to terrestrial neutrons [18]. Although more expensive and difficult to conduct, radiation testing provides a more accurate estimate of the sensitivity of the design to single-event upsets since radiation testing upsets *all* of the internal memory of the FPGA as well as single event transients (SET) [19].

A. Approach

The experiments were conducted by configuring one of the LEON3 design variations onto a Xilinx KC705 evaluation board with the FPGA mounted normal to the 2 inch collimated neutron beam, (see Figure 6). The dual LEON3 processor test system operates continuously in the beam and LEON3 processor errors are identified by the internal self-detection logic. When an error is detected, the error event is sent over JTAG to the JTAG controller and the controller reconfigures the FPGA with a clean design to initiate another test. The JTAG controller is also used to perform CRAM scrubbing of the device and to automate the test procedure. Automating these tests is necessary because of the time it takes to collect sufficient data for analysis.

[3]Although errors are not artificially injected into the BRAM, errors can enter the BRAM through errors in the surrounding logic.

TABLE III
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×)

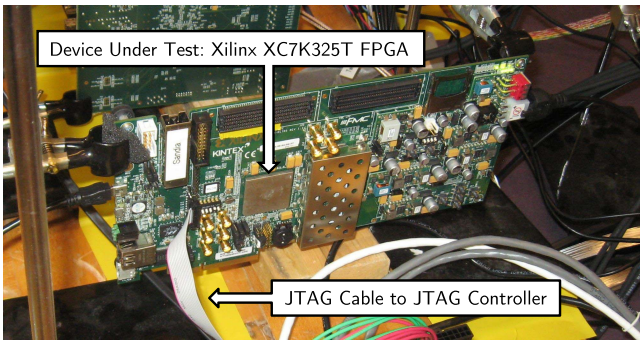

Fig. 6. Neutron Radiation Test Setup.

At the end of a beam run, the number of processor errors, n, is recorded along with the total neutron fluence. The neutron fluence at the FPGA board is calculated by derating the neutron fluence measured by the facility by a constant based on the distance of the target to the beam cap. Multiple beam runs are performed as needed to collect sufficient data to make good cross section estimates.

B. Metrics

The primary metric used to estimate the design sensitivity to ionizing radiation is the design *cross section*. The design 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,

$$\sigma = \frac{N_{errors}}{Fluence} \tag{2}$$

and has units of cm^2. The design cross section measures the cross section of the entire FPGA device running one of the five LEON3 processor variations, which includes the test architecture and two copies of the LEON3. The cross section captures all failure mechanisms of the design including CRAM upsets, BRAM upsets, upsets into hidden FPGA state, and single-event functional interrupts (SEFI). This design 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 VII).

C. Results

The results from the neutron radiation test are shown in Table III. The estimated cross section for each design variation is shown, including the number of failures observed, and the total fluence. 95% confidence intervals for the cross section estimate are computed using the standard deviation. The standard deviation of the cross section was estimated using a normal approximation of the Poisson mean with the equation:

$$\sigma = \frac{\sqrt{N_{errors}}}{Fluence} \tag{3}$$

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.

The estimates 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, and CRAM scrubbing. This combination of techniques reduced the estimated design cross section by 49×.

The results during radiation testing, summarized in Table III, are an improvement over the results obtained during a similar experiment [5]. In this previous experiment, the improvement in SEU sensitivity using all three mitigation techniques was only 10× in spite of fault injection results suggesting 51× 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.

TABLE IV
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×

D. Estimated Sensitive Bits

To facilitate the comparison of results obtained during fault injection tests against results obtained during radiation testing, the number of design "sensitive bits" is estimated for each LEON3 design variation from radiation test results. This "sensitive bits" estimate was calculated by dividing the estimated neutron cross section of the full design by the cross section of a single CRAM bit of the 7-series FPGA family. The neutron cross section of a CRAM bit for this family was previously measured at LANSCE at 6.99×10^{-15} [20]. The estimated number of sensitive bits for each design variation is summarized in Table III. The number of sensitive configuration bits of the unmitigated design, for example, is estimated at 330,822 or.045% of the Block 0 configuration bitstream.

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.

VII. COMPARISON OF FAULT INJECTION AND RADIATION TESTING 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 IV 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 II and III). 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.

A. SEU Sensitivity Improvement

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 comparison of the *relative* benefit of each mitigation technique for both fault injection and radiation testing. All SEU mitigation techniques provide improvement and the improvement increases as the 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 7 including the confidence interval bounds.

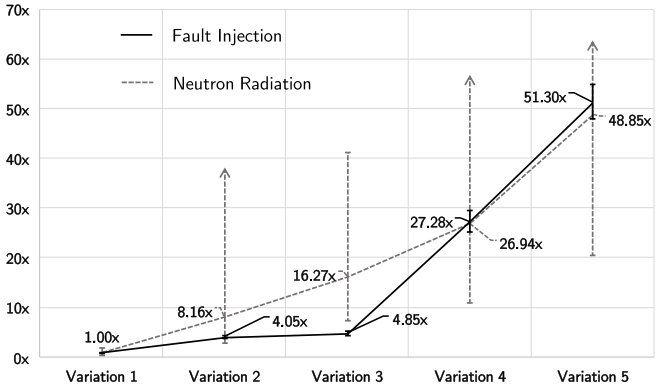

Fig. 7. SEU Sensitivity Improvement for Each Design Variation

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 the 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.

B. Estimated Sensitive Bits

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 IV.

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 during 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 to 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. The estimated number of sensitive bits for designs #2 and #3 is lower for radiation testing than for fault injection. These are the same designs that do not match in SEU sensitivity improvement between radiation testing and fault injection. For reasons that are not fully understood, CRAM scrubbing has a greater effect during radiation testing than fault injection.

VIII. Conclusion

The experiment in this paper tested more variations of SEU mitigation techniques than [5], which only compares the unmitigated design and the fully mitigated design (i.e., all three SEU mitigation techniques applied). Five SEU mitigation variations were tested: without SEU mitigation, TMR alone, TMR with BRAM scrubbing, TMR with CRAM scrubbing, and TMR with both BRAM scrubbing and CRAM scrubbing. Testing these combinations helped isolate the benefits of each SEU mitigation technique and explain the complementary relationships between them. Both fault injection and neutron radiation testing were conducted.

Improvement is measured in terms of sensitivity reduction for fault injection and cross section reduction for neutron radiation testing when compared to the unmitigated design. The results from *both* fault injection and radiation testing demonstrate that each variation of SEU mitigation techniques improve the SEU sensitivity of the LEON3, and that improvement increases as more mitigation techniques are combined. For most design variations (#1, #4, and #5) improvement observed by fault injection and radiation testing is similar.

When TMR or TMR with BRAM scrubbing are the only mitigation techniques applied to the LEON3, more improvement is observed by radiation testing than fault injection. This is most likely due to the fact that radiation testing upsets more FPGA state than fault injection. Combining BRAM scrubbing with TMR boost improvement to 16× for radiation testing. This suggests that BRAM scrubbing plays a significant role in improving the SEU sensitivity of FPGA designs that use a large number of BRAMs. Improvement gained from CRAM scrubbing (27×) is greater than that of BRAM scrubbing, suggesting that the majority of the bits utilized by LEON3 design are part of static configuration memory. When all three SEU mitigation techniques are combined (TMR, BRAM scrubbing, and CRAM scrubbing), approximately 50× improvement is observed by both test methods.

Future areas of study, for SEU mitigation of soft core processors, have been identified. These areas include: the separation of routing paths such that an SEU will not affect multiple TMR domains [21], the insertion of scrubbing for other internal memories, using fault injection and readback data to identify and eliminate single point failures, and adding software mitigation techniques.

References

[1] J. G. Tong, I. D. L. Anderson, and M. A. S. Khalid, "Soft-core processors for embedded systems," in *Proc. Int. Conf. Microelectron.*, 2006, pp. 170–173.

[2] Y. Ichinomiya, S. Tanoue, M. Amagasaki, M. Iida, M. Kuga, and T. Sueyoshi, "Improving the robustness of a softcore processor against SEUs by using TMR and partial reconfiguration," in *Proc. 18th IEEE Annu. Int. Symp. Field-Program. Custom Comput. Mach.*, May 2010, pp. 47–54.

[3] P. Graham, M. Caffrey, J. Zimmerman, P. Sundararajan, and E. Johnson, "Consequences and categories of SRAM FPGA configuration SEUs," in *Proc. 5th Annu. Int. Conf. Military Aerosp. Program. Logic Devices*, 2003, pp. 1–10. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.484.9371

[4] H. Quinn *et al*, "An introduction to radiation-induced failure modes and related mitigation methods for Xilinx SRAM FPGAs," in *Proc. Int. Conf. Eng. Reconfigurable Syst. Algorithms*, 2008, pp. 139–145.

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

[6] *Cobham Gaisler LEON3 Processor*. [Online]. Avalable: http://www.gaisler.com/index.php/products/processors/leon3

[7] M. Niknahad, O. Sander, and J. Becker, "FGTMR—Fine grain redundancy method for reconfigurable architectures under high failure rates," in *Proc. 15th North-East Asia Symp. Nano, Inf. Technol. Rel.*, 2011, pp. 186–191.

[8] J. M. Johnson and M. J. Wirthlin, "Voter insertion algorithms for FPGA designs using triple modular redundancy," in *Proc. 18th Annu. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2010, pp. 249–258.

[9] A. Manuzzato, S. Gerardin, A. Paccagnella, L. Sterpone, and M. Violante, "Effectiveness of TMR-based techniques to mitigate alpha-induced SEU accumulation in commercial SRAM-based FPGAs," in *Proc. Conf. Radiation Effects Compon. Syst.*, 2007, pp. 98–104.

[10] L. Sterpone, M. Violante, and S. Rezgui, "An analysis based on fault injection of hardening techniques for SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 53, no. 4, pp. 2054–2059, Aug. 2006.

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

[12] N. Rollins, M. Fuller, and M. J. Wirthlin, "A comparison of fault-tolerant memories in SRAM-based FPGAs," in *Proc. IEEE Aerosp. Conf.*, Mar. 2010, pp. 1972–1983.

[13] M. Berg *et al*, "Effectiveness of internal vs. external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis," in *Proc. Conf. Radiat. Effects Compon. Syst.*, 2007, pp. 459–466.

[14] K. Morgan, M. Caffrey, P. Graham, E. Johnson, B. Pratt, and M. Wirthlin, "SEU-induced persistent error propagation in FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2438–2445, Dec. 2005.

[15] M. Straka and Z. Kotasek, "High availability fault tolerant architectures implemented into FPGAs," in *Proc. 12th Euromicro Conf. Digital Syst. Design, Archit., Methods Tools*, 2009, pp. 97–104.

[16] P. S. Ostler *et al.*, "SRAM FPGA reliability analysis for harsh radiation environments," *IEEE Trans. Nucl. Sci.*, vol. 56, no. 6, pp. 3519–3526, Dec. 2009.

[17] H. M. Quinn *et al*, "A test methodology for determining space readiness of Xilinx SRAM-Based FPGA devices and designs," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 10, pp. 3380–3395, Oct. 2009.

[18] K. Schoenberg, "The lansce accelerator: A powerful tool for science and applications," in *Proc. IEEE Particle Accelerator Conf.*, Jun. 2007, p. 120.

[19] A. Spilla, *et al*, "Run-time soft error injection and testing of a microprocessor using FPGAs," in *Proc. Workshop Test Methoden Zuverlässigkeit Schaltungen Syst.*, 2011, pp. 1–6. [Online]. Available: http://ais.informatik.uni-freiburg.de/publications/papers/spilla11tuz.pdf

[20] Xilinx Inc., *Device Reliability Report, First Half 2016*, accessed on Nov. 28, 2016. [Online]. Available: https://www.xilinx.com/support/documentation/userguides/ug116.pdf

[21] F. L. Kastensmidt, C. K. Filho, and L. Carro, "Improving reliability of SRAM-based FPGAs by inserting redundant routing," *IEEE Trans. Nucl. Sci.*, vol. 53, no. 4, pp. 2060–2068, Aug. 2006.