Post-Radiation Fault Analysis of a High Reliability FPGA Linux SoC

Andrew Elbert Wilson
andrew.e.wilson@byu.edu
Brigham Young University
Provo, Utah, USA

Nathan Baker
nathangarybaker@byu.edu
Brigham Young University
Provo, Utah, USA

Ethan Campbell
ecampbell@byu.edu
Brigham Young University
Provo, Utah, USA

Jackson Sahleen
jsahleen@byu.edu
Brigham Young University
Provo, Utah, USA

Michael Wirthlin
wirthlin@byu.edu
Brigham Young University
Provo, Utah, USA

ABSTRACT

FPGAs are increasingly being used in space and other harsh radiation environments. However, SRAM-based FPGAs are susceptible to radiation in these environments and experience upsets within the configuration memory (CRAM), causing design failure. The effects of CRAM upsets can be mitigated using triple-modular redundancy and configuration scrubbing. This work investigates the reliability of a soft RISC-V SoC system executing the Linux operating system mitigated by TMR and configuration scrubbing. In particular, this paper analyzes the failures of this triplicated system observed at a high-energy neutron radiation experiment. Using a bitstream fault analysis tool, the failures of this system caused by CRAM upsets are traced back to the affected FPGA resource and design logic. This fault analysis identifies the interconnect and I/O as the most vulnerable FPGA resources and the DDR controller logic as the design logic most likely to cause a failure. By identifying the FPGA resources and design logic causing failures in this TMR system, additional design enhancements are proposed to create a more reliable design for harsh radiation environments.

CCS CONCEPTS

• **Computer systems organization** → **Redundancy**; *Embedded systems*; • **Hardware** → **Reconfigurable logic and FPGAs**; **Fault tolerance**; *Test-pattern generation and fault simulation.*

KEYWORDS

FPGA, TMR, RISC-V, soft processor, radiation testing, fault injection, fault analysis, reliability

FPGA '23, February 12–14, 2023, Monterey, CA, USA.

ACM ISBN 978-1-4503-9417-8/23/02…$15.00
https://doi.org/10.1145/3543622.3573191

ACM Reference Format:
Andrew Elbert Wilson, Nathan Baker, Ethan Campbell, Jackson Sahleen, and Michael Wirthlin. 2023. Post-Radiation Fault Analysis of a High Reliability FPGA Linux SoC. In *Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '23), February 12–14, 2023, Monterey, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543622.3573191

1 INTRODUCTION

SRAM FPGAs are increasingly being used in space vehicles to provide high-performance computing, high-bandwidth I/O processing, and in-system reconfigurability [1]. Many satellites perform high-fidelity sensor functions such as capturing high-resolution images that require a large amount of raw computation. The sensors on these systems generate a large amount of data requiring the on-board processing of high bandwidth data streams. FPGAs provide an ideal platform for many on-board satellite processing systems and offer the ability of dynamic reconfiguration to support updates and additional features.

SRAM FPGAs, however, are sensitive to the ionizing radiation found in space and other high-radiation environments [2]. Ionizing radiation causes upsets within the configuration memory (CRAM), internal block memory, flip-flops, and other internal FPGA state. Such upsets can often cause the design operating in the FPGA to fail. FPGAs used in such environments must employ some form of single-event upset (SEU) mitigation technique such as triple-modular redundancy (TMR) and/or configuration scrubbing [3].

A fault tolerant processor-based system running Linux was created with these SEU mitigation techniques to demonstrate the improvements in reliability. This system includes a RISC-V processor (VexRiscv), DDR controller, ethernet controller, UART, and other I/O. The improvement in reliability provided by these techniques was measured by operating the system in a high-energy neutron beam. The TMR system was found to have a 14× improvement in mean-neutron fluence to failure (MFTF) than the non-TMR system.

Although a 14× improvement in MFTF demonstrates that the SEU mitigation techniques do indeed improve reliability, the overall improvement is disappointing. A previous study of a "Bare Metal" VexRiscv processor demonstrated a 25× improvement in MFTF using the same TMR and scrubbing techniques [4, 5]. The purpose of this paper is to carefully identify and analyze the failures that occurred in this VexRiscv TMR Linux system and determine why

its improvement in reliability is so much lower than that of the previous "Bare Metal" system.

To perform this analysis, this paper introduces a methodology for identifying the source of CRAM upset induced failures at the FPGA device level and at the design level. This methodology is applied to the VexRiscv TMR Linux system and identifies the DDR controller as the most vulnerable portion of the design. The results from this analysis are used to propose a number of additional SEU mitigation techniques specific for this design to improve its reliability in harsh radiation environments.

This paper is organized as follows. Section 2 will summarize the Xilinx Artix-7 RISC-V Linux system used for this system. Section 3 will describe the neutron radiation test performed on this system at the Los Alamos Neutron Science Center (LANSCE), and Section 4 will discuss the post-radiation fault injection experiments that were performed to reproduce the failures seen at the radiation beam. Section 5 will provide an overview of the Bitstream Fault Analysis Tool used to analyze the CRAM faults from the radiation test. Section 6 analyzes the faults in the RISC-V system and provides a summary of the most vulnerable regions. Section 7 proposes improvements in the design that address the problems identified in the fault analysis.

2 RELIABLE RISC-V LINUX SYSTEM

Highly reliable applications implemented within FPGA systems lack ASIC processors to provide software interfaces and therefore, utilize soft processors [6]. The European Space Agency is exploring the use of FPGA-implemented soft processors to replace ASIC processors [7]. With existing motivation for use of the RISC-V instruction set architecture (ISA) in space applications [8, 9] (i.e. open license and integration with open source tools and libraries), recent works have examined the effectiveness of reliable processors implementing this ISA [7, 10, 11].

Within space processor systems, many spacecraft including microsatellite and CubeSat missions use embedded Linux for system critical functions [12] The Linux operation system is widely adopted, includes a huge catalogue of existing software, and implements industry standards such as POSIX. Software can be developed and debugged on local desktops. Compared to RTOS and standalone applications, the Linux kernel requires more computational power, often utilizing external DDR interfaces and local caches.

These complex soft processor Linux systems are susceptible to failures induced by SEUs. TMR mitigation can be necessary to prevent these critical failures within these systems. This paper investigates the single-point failures found within a TMR soft RISC-V Linux System on Chip (SoC). The SoC is implemented on a Xilinx series 7 SRAM-based FPGA utilizing a high-speed DDR3 memory controller. Fine-grain TMR is applied to the design, and external CRAM scrubbing is used as a repair mechanism.

2.1 Linux on Litex VexRiscv

The SoC system created for this experiment is based on the LiteX SoC framework for FPGAs [13]. This framework generates complex SoC systems using a library of processor cores, common IP I/O building blocks, and software code for control of the hardware. The LiteX framework is device independent and supports a variety of FPGA vendors and families to facilitate rapid development of SoC systems that can be targeted to a large variety of FPGA devices. In addition, LiteX maintains a large set of board support packages for quickly developing SoC systems for over 143 FPGA-based development boards.

The SoC system developed for this project is based on the VexRiscv processor, a 32-bit RISC-V processor supporting the RISC-V ISA and multiple ISA extensions [14]. The VexRiscv processor is implemented with the SpinalHDL language [15]. This processor is reported to have a maximum performance of 1.38 DMIPS/MHz and 2.57 CoreMark/MHz on Xilinx Artix 7 FPGAs [14] implemented with a configuration with a maximum frequency of 151 MHz and utilization of 2021 LUTs and 1541 FFs.

The SoC system used in this project targets the Digilent Nexys Video development board that includes the Xilinx's XC7A200T-1SBG484C FPGA. This board was chosen because of its low cost, DDR3 memory, Ethernet, and SD-card support. The SoC system is configured with Ethernet, serial communication, DDR3 memory controller, and HDMI video output IP cores (see Figure 1). The VexRiscv configuration for this SoC includes 4 kilobytes of Icache and Dcache and a Wishbone system bus. The implemented design consumes about 7% of the FPGA and operates at 111 MHz (see Table 1).

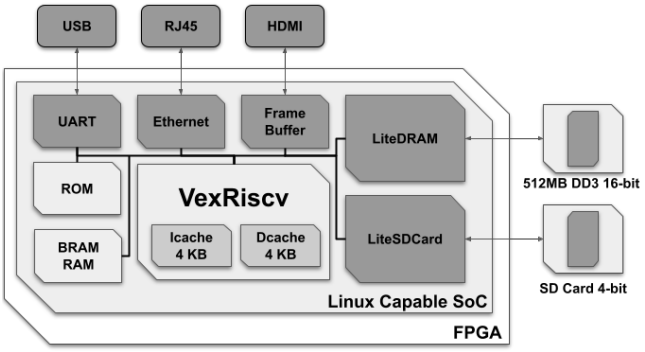

Figure 1: VexRiscv System Architecture.

The SoC system used in this project runs Linux and relies on the "Linux on Litex-Vexriscv" [16] project that facilitates the rapid deployment of a Buildroot-based Linux RISC-V SoC. The Linux system provides a full Ethernet software stack, file system, and drivers for the I/O devices in the system. The system is configured to continuously execute the Dhrystone benchmark after booting and publish the results of the benchmark over the UART.

2.2 TMR Litex VexRiscv System

When placed in a harsh radiation environment such as space, the SoC system described above will experience failures as ionizing radiation upsets the state of the CRAM, block RAM, and other internal state. SEUs can cause functional failures in the design within the FPGA by corrupting the state and circuit configuration, potentially leading to a critical failure of the system. To operate this system reliably in such an environment, single-event upset mitigation techniques must be applied [3]. This section will describe

the approach used to improve the reliability of the SoC system described above.

A variety of approaches for improving the reliability of soft processors within an FPGA have been proposed and demonstrated. Other works have explored TMR mitigation targeting different ISAs such as the LEON processor [17–21] and the Picoblaze [22, 23]. One work applied double modular redundancy to a RISC-V processor and verified the mitigation through fault injection on a Xilinx FPGA [24] Xilinx also offers a Microblaze TMR subsystem for use within their FPGAs [25]. The approach used in this work to mitigate the effects of SEUs is to apply triple modular redundancy and configuration scrubbing.

TMR is an effective mitigation technique capable of masking single point failures within an FPGA soft processor and other FPGA logic [26]. As shown in Figure 2, triple modular redundancy is implemented by triplicating all resources of a design and inserting majority voters (usually triplicated voters). Any single failure in the logic or the voter will be masked by the downstream voters. Although TMR provides additional reliability through redundancy, this mitigation technique results in greater power consumption, higher resource utilization, and slower maximum frequency.

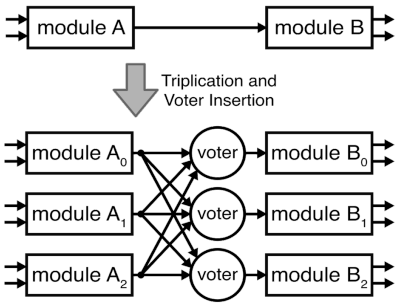

Figure 2: Triple Modular Redundancy and Majority Voting.

There are a variety of approaches for applying TMR to FPGA systems. One work utilized the Mentor Precision Hi-Rel tool to apply fine-grain TMR to the RISC-V Rocket Chip and through fault injection observed a reduction in sensitive bits of up to 11.5× [7]. Another work used Cadence's EDA flow to apply fine-grain TMR to same processor and achieved a 3× reduction in the cross-section during heavy-ion testing [10]. Other work implemented a TMR VexRiscv processor with Synplify's TMR tools and demostrated a 1.5× improvement in the mean time to failure (MTTF) with JTAG fault injection [11].

The approach used in this project performs triplication and voter insertion at the post-synthesis netlist level. In particular, the SpyDrNet TMR tool was used to apply TMR to the post synthesis netlist [27]. This Python-based tool performs fine-grained TMR on the FPGA primitives at the netlist level by triplicating all FFs, LUTs, BRAMs, and DSPs, and inserting triplicated voters between these primitives. The tool's input is a vendor-independent Electronic Design Interchange Format (EDIF) file that can be exported from Xilinx Vivado. The generated TMR EDIF file can be imported back into Xilinx Vivado as a post-synthesis file, and placed and routed in the final design (see Figure 3).

Figure 3: TMR with triplicated voters.

Using the SpyDrNet TMR tools, the original non-mitgated design is triplicated and implemented in the same FPGA device as described above. The FPGA resource utilization of both the non-mitigated and TMR mitigated designs are summarized in Table 1. Although the design is triplicated, the LUT utilization is greater than 3× as additional LUTs are used for the inserted triplicated voters. The maximum frequency (fMax) decreases as the design's critical path increases due to the voter and feedback paths. Figure 4 compares the place and routed floorplan of each design.

Table 1: VexRiscv SoC Design Utilization for Non-TMR and TMR Designs.

Design	LUT	LUTRAM	FF	BRAM	FMAX
Unmitigated	9131(6.8%)	521(1.1%)	7641(2.8%)	36(9.9%)	111.6MHz
TMR	37846(28.1%)	1611(3.5%)	22905(8.5%)	108(29.6%)	91.4MHz
Cost Ratio	4.14×	3.1×	3×	3×	0.81 ×

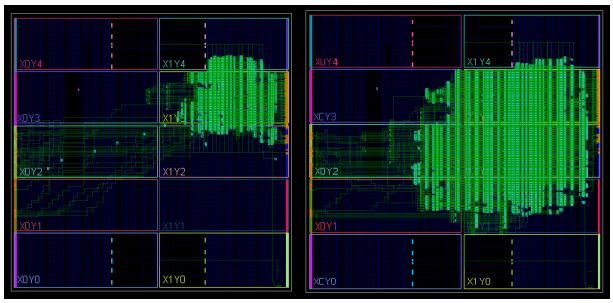

Figure 4: Unmitigated (left) vs TMR (right) FPGA floorplans.

3 HIGH ENERGY NEUTRON RADIATION TEST

High energy radiation testing is often performed to induce single-event effects (SEE) and measure the impact of these effects on complex systems [28]. Such testing is often called "accelerated" testing since the flux of the high energy particles is much higher than the flux of radiation in an actual environment (i.e., ground level radiation or radiation in a specific orbit). The failures observed at a radiation test can be used to measure the benefits of SEE mitigation techniques and to estimate the failure rate of the system in a relevant radiation environment [29].

The non-TMR and TMR versions of the VexRiscv system were tested at the Los Alamos Neutron Science Center (LANSCE) [30] in September of 2021. This facility provides a high energy neutron beam that provides a wide spectrum of neutron energies (up to 800 MeV) that closely matches the energy spectrum of neutrons found on earth (called terrestrial neutrons). This facility is often used to measure the failure rate of electronic circuits due to terrestrial

neutrons. This section will describe the organization of this neutron test and the improvements in mean fluence to failure of the TMR VexRiscv system over the conventional unmitigated version.

3.1 CRAM Scrubbing

To prevent the accumulation of CRAM upsets within the FPGA during the radiation test, a CRAM repair technique called "CRAM scrubbing" is used [31]. Configuration scrubbing involves repeatedly reading the contents of the configuration memory, comparing the memory against a golden copy of the bitstream, and reporting and repairing any errors that are observed. For Xilinx FPGAs, configuration data can be read and written without affecting the operation of the FPGA. All of the CRAM upsets that occur during this radiation test are logged for post-radiation fault analysis (see Section 4).

Configuration scrubbing is especially important when applying TMR to a system. A TMR system will only tolerate a single fault in the system – additional accumulated faults that affect the other TMR copies will cause the system to fail[1]. TMR systems that include a repair mechanism (configuration scrubbing in this case) provide *significantly* higher reliability than non-TMR systems. A fault in one copy of the TMR system due to a configuration upset can be repaired through configuration scrubbing before another fault occurs in the system.

For this work, configuration scrubbing is performed over JTAG utilizing an external JTAG controller called the "JTAG Configuration Manager" or JCM [32]. The JCM is an embedded Linux system that provides high-speed JTAG sequences for FPGA configuration, configuration scrubbing, and fault injection. The JCM is connected to the JTAG port of the Nexys Video board and accessible to a remote host over Ethernet. The JCM can perform a scrub of the entire bitstream for the device in about 3.3 seconds. After the completion of a complete scrub of the device (called a *scrub cycle*), the JCM provides a timestamp of the cycle and reports the specific address(es) of all upset configuration bits.

3.2 Test Organization

The radiation experiment was performed within the ICE house beam line (Target 4, 30 degree right at the Weapons Neutron Research (WNR) facility) within the Los Alamos Neutron Science Center (LANSCE) at the Los Alamos National Laboratory (LANL). The Nexys Video board was placed in the neutron radiation beam path with the Artix 7 FPGA centered with the beam as shown in Figure 5. The FPGA board is placed at a normal angle of incidence and operates at room temperature. A collimator with a diameter of 2 inches was placed within the beam to limit exposure of the high-energy radiation to other board components.

The VexRiscv system operating on the Nexys Video board must operate remotely without any human interaction for safety reasons. This system can boot Linux image over Ethernet or the SD card. The SD card proved to be more convenient and was used throughout testing. After booting, the Linux OS performs a continuous Dhrystone utilizing the processor cache and DDR3 memory. The

[1]An often overlooked property of systems employing TMR is that a TMR system without a repair mechanism actually *reduces* the mean-time to failure of the system. Such TMR systems without repair provide improved reliability of the system early in the mission life but in the long run, fail sooner than a non-TMR system.

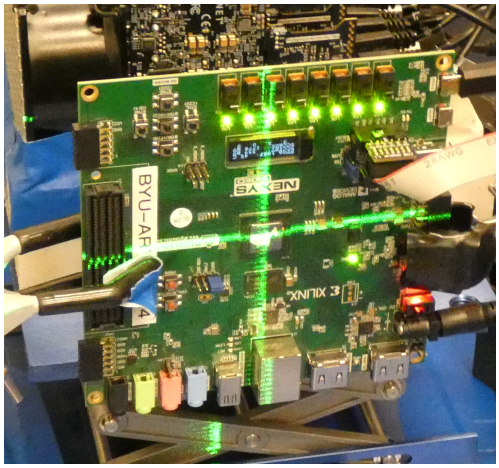

Figure 5: Nexys Video Board in LANSCE Neutron Radiation Beam Line (Laser Crosshairs Mark the Neutron Beam Centered on the Artix 7 FPGA).

Dhrystone results are published over the UART serial communication. The experiment is controlled with a remote host computer (NUC) connected to the system via Ethernet. The NUC logs the UART output of the system to ensure that the system is operating correctly. The NUC also controls the JCM JTAG interface for configuring the device and performing/logging the scrubbing process. The organization of the test setup is shown in Figure 6.

An experiment run begins by configuring the FPGA device with the design under test. The NUC carefully logs the boot process and subsequent execution process of the system. Any failures observed during the boot process or during the system execution are recognized by the NUC. The experiment involves performing as many runs as possible during the allotted test time. The reliability of the system is measured in terms of "fluence to failure" or total amount of neutron fluence (neutrons/cm^2) divided by the number of system failures. A related metric is the sensitive cross section, σ, which is the inverse of fluence to failure (in units cm^2).

Five days of radiation test time were scheduled for the experiment (August 31-September 4, 2021). With the facility operating for 24 hours a day, 120 hours were available to complete the experiment. Only a single board was used for the experiment, so the beam testing time was divided between the non-mitigated system and the TMR mitigated system. One-fourth of the time (30 hours) was scheduled for testing the non-TMR system and three-fourths of the time (90 hours) were scheduled for the TMR design. More time is needed for the mitigated TMR design since much more neutron fluence will be needed for each system failure.

3.3 Test Results

The results from the experiment are summarized in the bottom two rows of Table 2. 76 system failures were observed on the non-TMR system and 27 failures were observed in the TMR system. The mean fluence to failure is 4.79×10^8 n/cm^2 for the non-TMR system and 7.07×10^9 n/cm^2 for the TMR system. In a terrestrial environment, the non-TMR system will have a mean-time to failure (MTTF) of

Table 2: Neutron Radiation Test Data

Design	LUT Utilization	Normalized Utilization	Fluence (n/cm^2)	Observed CRAM Upsets	Failures	Cross Section (cm^2)	+95% Confidence -95% Confidence	Reduction	MWBF
VexRiscv Unmitigated	2261 (0.7%)	1.00×	7.08×10^{9}*	3473*	422*	5.09×10^{-10}	5.59×10^{-10} 4.60×10^{-10}	1×	1×
VexRiscv TMR	10002 (3.0%)	4.42×	5.38×10^{10}*	21783*	30*	1.99×10^{-11}	2.84×10^{-11} 1.34×10^{-11}	25.57×	23.80×
Linux-VexRiscv Unmitigated	9131 (6.78%)	4.04×	3.64×10^{10}	11908	76	1.78×10^{-9}	2.26×10^{-9} 1.30×10^{-9}	1×	1×
Linux-VexRiscv TMR	37846 (28.12%)	16.74×	1.91×10^{11}	59014	27	1.21×10^{-10}	3.26×10^{-10} 2.73×10^{-11}	14.76×	12.07×

* Results from related work with multiple processor implementations on Xilinx Kintex UltraScale+[4].

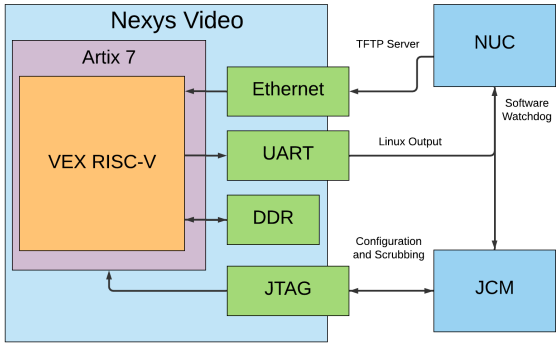

Figure 6: VexRiscv Linux System Test Diagram

4,554 years and the TMR system MTTF is estimated at 67,210 years. The overall improvement in MTTF of the TMR system over the non-TMR system is 14.7×. If the slower clock rate of the TMR system is considered, the TMR system has a "mean work between failure" (MWBF) improvement of 12.1×.

The top two rows of Table 2 summarize the results of another work performed with the same VexRiscv processor in a "standalone" arrangement (i.e., processor only without DDR and other peripherals) [33]. For this previous experiment, the standalone TMR processor achieved an MTTF improvement of 25.6× over its non-TMR baseline design. This 25.6× improvement of the standalone system is much greater than the 14.8× improvement of the Linux VexRiscv system. The reduction in improvement provided by TMR in this system vs. the standalone counterpart is the primary motivation for the fault analysis described in the remainder of this paper.

4 POST RADIATION FAULT INJECTION

Although the single-event mitigation methods employed did improve the reliability of the system, the overall reliability improvement over the non-TMR system was disappointing. Several post-radiation fault injection approaches were employed to understand how the TMR system failed so that future improvements in the system can be made. The post-radiation fault injection approaches performed on this system include "playback" fault injection and "correlated" fault injection. Both of these approaches are described below.

4.1 Playback Fault Injection

The first approach for fault analysis was to "playback" the CRAM faults observed at the radiation test. The goal of this fault injection approach is to see how many of the 27 radiation-induced failures observed at the radiation test can be reproduced in the lab by injecting the same CRAM upsets logged during the radiation test.

The JCM scrubbing hardware was modified to "playback" all the CRAM faults seen at the test. This fault injection is operated by iterating through each sensitive scrub cycle from the radiation test and injecting all CRAM faults of the cycle into the system. After injecting the faults, the fault injection system pauses to allow the faults to propagate through the system. If no failure in the system is observed, the CRAM bits are repaired and the fault injection moves on to the next cycle in the list. If a failure in the system is observed, then the corresponding CRAM bits within the cycle are tagged, the FPGA system is reset and reconfigured, and the fault injection continues with the next scrub cycle in the list.

The JCM logged 32,658 scrub cycles with CRAM upsets during the radiation test of the TMR Linux VexRiscv design. Within these scrub cycles, 59,014 configuration bits had upset for an average of 1.8 configuration upsets per sensitive scrub cycle. Performing a full playback of all 32,658 cycles in the sensitive cycle list requires 115.2 hours to complete or 12.7 seconds per cycle. The playback was performed three times on three different boards (over 350 hours) for a total of nine complete playback events. The playback fault injection identified 22 cycles within the playback list that caused the system to fail. These can be further catagorized as follows:

- 17 failed every time on all three boards,
- 4 failed on some but not all of the boards, and
- 1 failed on all boards but with less than 100% probability.

The 22 sensitive scrub cycles found through playback fault injection are fewer than the 27 failures that were observed during the radiation test. This is not surprising as it is likely that failures in the FPGA system at the radiation test were caused by upsets within the CRAM as well as state elements that are not captured during the scrubbing process. For example, upsets within flip-flops that are not triplicated may cause system failures but will not be observed with configuration scrubbing. Other state registers that may cause failures and not available within the CRAM scrubbing include Dynamic Reconfiguration Port or DRP bits, internal FPGA state not accessible by the user, and BRAM bits.

4.2 Correlated Fault Injection

The next step was to correlate these sensitive scrub cycles found with playback with the actual failure events within the radiation test. This was done by comparing the timestamp of each scrub cycle with the timestamp of the failure log. Those scrub cycles that were found within a 60 second window of a system failure were tagged as causing the given failure. Of the 22 sensitive scrub cycles identified by the playback fault injection, 15 of them were found within a 60 second window of one of the 27 test failures.

The playback fault injection described above identified sets of configuration bits that when upset together as a group, cause the system to fail. Although this is useful, we are more interested in identifying the specific individual bits within the scrub cycle that caused the system to fail. To find these individual bits, each bit within a sensitive cycle was upset by itself within the system to observe the system behavior. From this test, 14 individual bits within the 15 cycles were identified. For one of the cycles, no individual bit upset caused the system to fail. This cycle contained two bits and the system would only fail when both of the bits were upsets. This surprising result suggests that there are cases when a TMR system can fail due to two simultaneous upsets. This is concerning as it is not uncommon for more than one upset to occur with one ionizing particle [34].

5 BITSTREAM FAULT ANALYSIS TOOL

In the radiation experiments described in the previous section, we identified a number of CRAM bits that are known to cause the design to fail when upset. To help us understand why the design failed, we seek to understand which resources within the original design are associated with these bits. A related work used targeted fault injection to perform fault analysis on soft LEON3 processor components such as registers, pipelined logic, and caches [35]. Other works have used simulation-based fault analysis of a RISC-V processor to categorize the different failure modes [36, 37]. This work utilizes sensitive CRAM bits collected from radiation testing and fault injection to perform fault analysis on the routed digital design.

A tool named "Bitstream Fault Analysis Tool" or BFAT was created to identify the design resources associated with specific bits in the bitstream [38]. This tool was used to identify the cause of the radiation-induced system failures for each of the 15 sensitive configuration bits. A brief overview of the steps involved in using the BFAT tool is shown in Figure 7. The inputs to the tool include a list of bit addresses of interest, the original design checkpoint file, and the original bitstream. Each bit in the bit address list is specified by its frame address, word number, and bit number. The design checkpoint file (or .dcp file) contains the details of the implementation of the design including its placement, routing, and the design names associated with all of the resources used by the design.

Each of the steps of this flow will be described using an example with a single configuration bit chosen from the radiation test. The specific configuration bit used for this example occurs at frame address 0x402785, word 100, and bit 15 (defined as the tuple `[0x402785, 100, 15]`). The original value of the bit in the bitstream is '0' and a radiation induced upset changed it to a '1'.

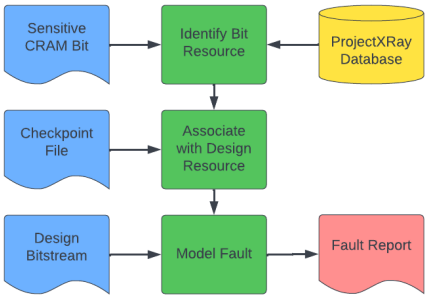

Figure 7: Bitstream Fault Analysis Tool Flow

This upset occurred alone during scrub cycle #2094 on September 3, 2021 at 03:37 am and led to failure of the SoC Linux system. The JCM log file entry associated with this upset is shown below:

```
Cycle 2094 - 1 upsets (2021-09-03 03:37:04)
00402785(10040) 100 00008000
```

The first step of the BFAT flow is to take this bit address and determine the FPGA device resource associated with the bit. This is done by querying the Project XRAY database file for the device. Project XRAY has created a database of bitstream definitions for a variety of devices including the device used in this experiment [39]. The project XRAY database indicates that this particular bit references a tile named `"INT_R_X79Y149"`. This tile is an interconnect tile (specifically an `"INT_R"` style) and is located at tile coordinates [79, 149] (see Figure 8). Within this tile, this bit controls the behavior of the interconnect output port named `"SW6BEG2"` in the lower left corner of the tile. In particular, it enables the input port `"WW2END2"` (located in the upper right corner of the tile) to drive the `"SW6BEG2"` output signal.

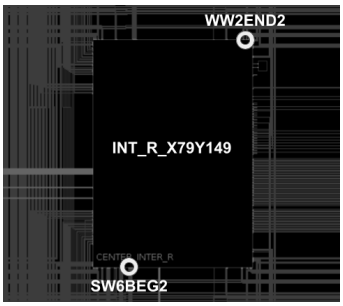

Figure 8: Tile `INT_R_X79Y149` and Affected Ports.

The second step in the BFAT flow is to correlate the affected device resource with the design function as seen from the designer's perspective. The design resource associated with the device resource is determined by querying the design checkpoint file to determine the name(s) given to the affected device resource by the synthesis and implementation tools. These design names provide very useful information about the resource including the location within the hierarchy of the design, the netname, or logic name. For

the current example, the BFAT tool queries the design database to determine which nets, if any, are hooked up to the "WW2END2" and "SW6BEG2" ports of the "INT_R_X79Y149" tile. For this design, there is a unique net hooked up to each of the two ports. The signal named "ISERDESE2_15_n_4" is hooked up to the "SW6BEG2" port and another signal[2] is hooked up to the "WW2END2" port. The first signal is an input coming directly from the ISERDES I/O port and is associated with the DDR interface. The second signal is used for the interconnect switch for the memory devices within the SoC.

The final step in the BFAT flow is to determine how the individual CRAM upset modifies the design and possibly causes a system failure. This is done by modeling the device resources and describing how the device behavior changes. In this example, BFAT determines that this CRAM upset will cause a short between the two nets at the two specified ports of the design. This short is demonstrated in Figure 9. In this example, the "ISERDESE2_15_n_4" signal is represented by the yellow line and the other net is represented by the blue line. The red line indicates the short caused by the upset of the given CRAM bit. The BFAT tool can model a variety of faults including opens and shorts in the interconnect, changes in logic within the CLB tiles, and other device-specific failure modes.

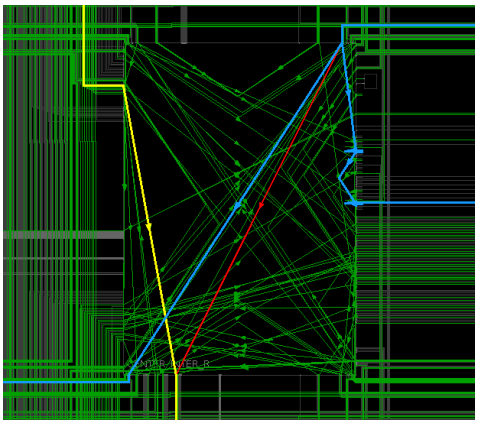

Figure 9: Tile INT_R_X79Y149 with Short Between the Ports "WW2END2" and "SW6BEG2" on the Linux SoC design.

A short between these two nets will cause incorrect values to be propagated on the "ISERDESE2_15_n_4" design net (i.e., "SW6BEG2" output port). This particular net is *not* triplicated as it is a signal coming directly from one of the non-triplicated DDR inputs. Because the signal is not triplicated, faults on this net will not be resolved with voting and the incorrect data will be propagated throughout the system. For this design, a short on this net crashes the Linux system.

[2]The netname associated with this signal is too long to include in the body of the text. The actual netname of this signal within the design is: "VexRiscvLitexSmpCluster_Cc1_Iw32Is4096Iy1_Dw32Ds4096Dy1_ITs4DTs4_Ldw128_Ood/dBusNonCoherent_bmb_cmd_s2mPipe_m2sPipe_rData_fragment_address_TMR_0[14]"

5.1 Radiation-Induced Failure Bitstream Analysis

Each of the 15 sensitive CRAM bits (or set of bits in one case) were analyzed with the BFAT tool to determine how these CRAM upsets caused the design to fail. Of these 15 upsets, 11 occurred within interconnect tiles, 2 within look-up tables within a CLB tile, and 2 upsets could not be characterized. Upsets in the look-up tables changed the logic function within the circuit. Of the 11 interconnect faults, 4 caused an "open" in the design net and 7 caused a "short" between two signals.

The results from the BFAT analysis were also used to investigate why the TMR SEU mitigation approach failed to protect the design in each of these cases. The design resources affected by each CRAM fault were analyzed by hand to understand the failure. In three cases, a single CRAM upset caused two signals from different TMR domains to short. If two signals from two different TMR "domains" are shorted, the TMR voting will fail and produce the wrong result for the given signal, as demonstrated in a related work [40]. Since TMR can only protect faults in one TMR domain at a time, such faults may break the TMR system. Such a fault is similar to a two-bit upset that occurs in a word of data protected by Single-Error Correction, Double Error Detection (SECDED) error correction code.

In the other thirteen cases, upsets occurred in regions of the design that were not triplicated and thus were "single point failures" (SPF). Single-point failures occur when the design cannot triplicate a given resource. Such resources usually include I/O pins, clock buffers, and other special purpose resources such as clock managers or PLLs. In this design, two of the failures occurred in the global clock buffer driving the primary global clock. In the other eleven cases, the failures occurred in the interconnect associated with the I/O pins of the DDR interface.

5.2 Fault Analysis on Larger CRAM Sample Set

The fault analysis of the failures observed at the radiation test experiment was helpful in determining vulnerable areas of the design. However, the small number of faults that we were able to analyze does not provide sufficient statistical confidence into the causes of the system failure. To improve our understanding of the sources of failure in the design, a much larger set of sensitive faults were identified and analyzed using the BFAT tool. The purpose of identifying more faults is to provide more statistical confidence as to the source of the sensitive areas of the design. This information will guide efforts to improve the design through additional SEU mitigation methods.

A more thorough fault injection experiment was performed to identify at least 100 sensitive CRAM bits within the design. Since this particular design requires on average 2,370 fault injections to observe one failure, this fault injection campaign would require over 237,000 injections to obtain the target 100 sensitive CRAM bits. With one injection taking on average 12.7 seconds, this fault injection campaign took over 34 days of continuous fault injection. At the completion of this more thorough fault injection experiment, 106 sensitive CRAM bits were identified.

The BFAT tool was used to analyze each of the 106 failures in order to identify the FPGA resources causing the failure as well as

the design functionality that is failing. The FPGA device resources that caused the design to fail are summarized in Table 3. As shown in this table, the most significant source of failures are faults within the interconnect resources. Over 70% of design failures were caused by upsets within the interconnect resources. Of the 75 sensitive interconnect faults, 57% of them were caused by "opens" in a net and 43% due to "shorts".

Resource	Occurrences	Estimated Bits
Interconnect	75 (70.8%)	17,669
I/O Pin	19 (17.9%)	4,467
Clocking	6 (5.7%)	1,422
CLB	4 (3.8%)	948
Undefined	2 (1.8%)	449
Total	106	24,956

Table 3: FPGA Device Resources Causing Design Failure.

The next most common failure mode occurred within configuration bits associated with the I/O resources. Faults within the I/O will cause system failures as the I/O resources are not triplicated [41]. Examples of sensitive CRAM bits that cause system failure within the I/O include: the OLOGIC "CLKDIV" bit, OLOGIC "IN USE" bit, OCLK "IOI_LEAF_GCLK3", and OSERDES "DATA_RATE_TQ" bit. Faults in the clocking resources (CLB, PLL, etc.) and CLBs made up the smallest portion of system failures.

6 RISC-V LINUX DESIGN FAULT ANALYSIS

While it is interesting to see which FPGA device resources cause the system to fail, we are more interested in determining which logical circuits in the original design are causing the system to fail. This information is needed to identify design-specific vulnerabilities so reliability improvements can be made. This section will describe how the FPGA device resources identified in the previous section cause design-specific system failures.

Using the BFAT tool, 104 of the 106 FPGA device faults described above were correlated to specific design logic described in the original RTL system and are summarized in Table 4. The most sensitive part of the design is the DDR controller interface, which comprises almost 80% of all system failures. The global clock network is the next most sensitive part of the design, causing 17% of the system failures. The rest of the failures were identified with the constant logic, the reset circuit, and a miscellaneous logic in the design. The cause of these failures will be described in the following sub-sections.

Design Function	Occurrences
DDR Interface	81 (77.9%)
Global Clock	19 (18.3%)
Constant Logic	2 (2.1%)
Reset Circuit	1 (1.0%)
Misc. Circuitry	1 (1.0%)
Total	104

Table 4: Design Resources Causing System Failure.

6.1 DDR Interface

Not surprisingly, the most vulnerable portion of this design is the DDR interface. Since the I/O pins associated with the DDR interfaces are not triplicated, any faults within these I/O will likely cause a system failure. Of the 93 I/O pins used for this design, 48 (52%) of them are associated with the DDR interface. Although some single-point failures in the DDR I/O interface were expected, we were surprised at the extent at which the DDR interface failed. The failures associated with the DDR take on a disproportionate impact when compared to its relative use of I/O pins. Like all other parts of the design, the DDR controller logic was triplicated, and we only expected failures when faults occur within the I/O pins of the device.

The DDR related system failures were analyzed and broken down based on the device resources causing the failure. Of the 81 failures associated with the DDR interface, 64 (79%) of these failures occurred within the FPGA interconnect, 14 occurred within the I/O resources, and 3 occurred within a CLB. This surprising result suggests that it is the *interconnect* associated with the DDR interface that causes the bulk of the DDR system failures rather than the I/O.

To understand why the interconnect is so sensitive, we carefully reviewed each interconnect fault within the design. After manually inspecting each interconnect fault within the implemented design we realized that failures due to these interconnect faults occur in *untriplicated* regions of the net tree. Although the net attached to an untriplicated I/O signal may be triplicated, the actual routing associated with this net may not branch into triplicated nets for some distance from the I/O pin. Figure 10 demonstrates this condition with an untriplicated input pin and a net associated with that pin that does not split into three nets for some distance. The trunk of the signal net remains a single-point failure (highlighted as red in the figure) and any faults within this trunk will cause a failure within the system. If the distance between the source of the net and the split in the net is long, then a large number of interconnect resources are vulnerable to single point failures.

Figure 10: Vulnerable Segments of Signal Trunk Originating from an Untriplicated I/O.

The "ISERDESE2_15_n_4" signal described in the previous section is a good example of this condition. The routing of this signal within the device is shown in Figure 11. This signal begins at the I/O and is driven by an ISERDES I/O resource. The signal proceeds as a single trunk for a relatively long distance before it splits into three distinct triplicated branches. Although this signal is triplicated, the

relatively long distance between the trunk and the branches exposes a relatively large single point failure for the design. The short in the net caused by an upset in bit [0x402785,100,16] physically occurs in this vulnerable region of the signal.

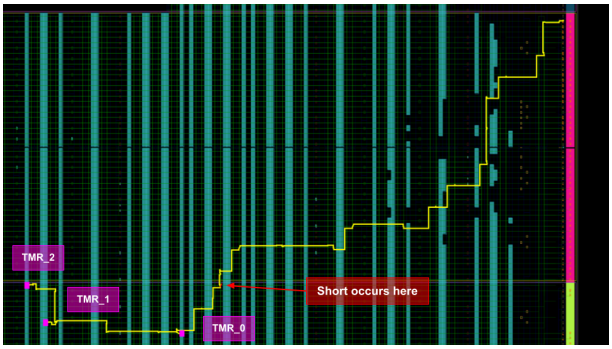

Figure 11: Routing of "ISERDESE2_15_n_4" Signal and Location of Interconnect Short.

This situation of vulnerable net trunks also occurs with output signals. For output signals, a reduction voter is added to the netlist to reduce an internal triplicated signal to a single signal input to an output I/O resource. If the voter associated with this output is placed at a relatively far distance from the I/O pin, the corresponding net will be very sensitive to CRAM upsets. Of the 64 DDR interconnect faults identified within the design, 36 (56%) occurred on untriplicated input I/O signals and 28 (44%) occurred on untriplicated output I/O signals.

The reliability of the DDR interface can be significantly improved by manual placement and routing of the triplicated I/O net resources. For input signals, the reliability can be improved by routing the signal such that the branches for the triplicated signals are located as close to the input I/O pin as possible. For outputs, the reliability can be improved by placing the reduction voter as close to the output I/O pin as possible. It is likely that most of the interconnect faults associated with the DDR interface can be removed by applying these techniques.

6.2 Global Clock Network

The next largest design component contributing to the design failures is the global clock network. Like the DDR interconnect problem, the presence of such a large clock vulnerabiltiy caught us by surprise. The SpyDrNet TMR tools triplicated the global clock buffer, and the implementation tools accepted this triplication approach. We recognized that there would be a small untriplicated trunk of the clock tree but were surprised by the large number of clock related faults. As with the DDR interconnect problem, the problem with the clock network was carefully reviewed using the BFAT tool.

The results of the BFAT fault analysis quickly identified the problem with the clock network – the clock network was not actually triplicated in the design implementation. During the implementation phase, the triplicated global buffers were replaced with a single global buffer and the three independent clock nets were combined into a single net. This optimization went unnoticed during the design process and was only recognized well after the radiation test when the results were analyzed. Not surprisingly, the lack of a triplicated global clock resulted in a relatively large component of the system failures at the radiation test.

To address this problem, a manual triplicated clocking network must be created. One approach is to create three synchronous clock signals using a mixed-mode clock manager (MMCME2) as suggested by Figure 12. In this example, three global buffers are used for three different synchronous global clocks. Although the MMCM and input clock signal can be upset, the overall vulnerability of the clock network will be significantly reduced.

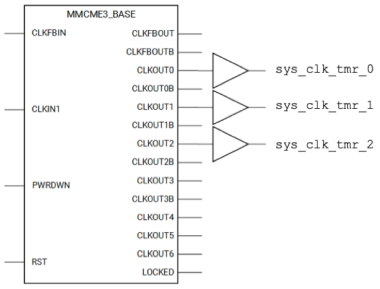

Figure 12: Clock Manager with Triplicated, Synchronous Clocks.

6.3 Other Vulnerable Logic

Four additional faults were found in other areas of the design outside of the DDR interface and global clock network. Although their contribution to the overall failure rate of the system is small, they present interesting failure modes that are worth discussion. These failures occur in constant logic signals, the reset circuit, and in a CLB within the triplicated region of the system.

Yet another surprising result from the BFAT analysis was the fact that two failures were due to faults with the "GLOBAL_LOGIC0" and "GLOBAL_LOGIC1" constant signals. Constant signals are sometimes added to the design netlist to provide a constant value within the design at certain device resource inputs. A local constant signal is then added to make sure the input receives the proper value as needed by the implemented design. These signals were not triplicated, so faults in these local constant nets cause failures in the system as the constant input will be incorrect for all three copies of the resource. These failures can be addressed by triplicating these global signals after the implementation phase of the design flow.

One CRAM fault occurred within the I/O pin associated with the global reset signal. Because this pin is not triplicated, nothing can be done to improve the reliability of this part of the circuit.

An interesting failure was found within a CLB resource of a triplicated part of the logic. In this case, a slice within the CLB generated the value for two copies of the same signal (i.e., two of the three triplicated signals). The configuration bit associated with the "NOCLKINV" resource of the slice was changed causing the slice to invert the phase of the clock. Since this fault caused two of the three copies of the signal to change, the TMR voters will choose the incorrect signal and the system will fail. As suggested

in a related work [42], this problem could be avoided by making sure that no two copies of the same signal are computed within the same slice.

7 IMPROVING RISC-V SYSTEM RELIABILITY

The primary purpose of this work is to carefully analyze and understand all failures of the TMR SoC system observed in the radiation experiment. Once the cause of these failures is understood, design changes can be made to improve to overall system reliability. This section will summarize the changes that will be made to this design based on the fault analysis described in this paper. In addition, an estimated improvement in system reliability will be given.

The most important design change that will be made to improve reliability is to reduce the vulnerability of the interconnect resources. The two primary contributors to interconnect-induced failures were the signals associated with top-level ports and the global clock network. As described above, the interconnect signals associated with the top-level ports can be protected from failure by manually controlling the routing and placement of voters. For output I/O signals, this design change will involve placing the reduction voters as close to the output I/O resource as possible. For input I/O signals, this will involve forcing the triplicated branches from the signal trunk to split much closer to the I/O signal. Both of these steps will be done manually, but in the future, tools may be developed to automate this process.

The interconnect associated with the global clock will be protected by creating three clock signals with a clock generator as suggested in Figure 12. This step will require manual manipulation of the netlist to insert global clock buffers and attach the triplicated global clocks. It will not be possible to completely eliminate interconnect failures as there will be at least one programmable interconnect associated with the non-triplicated portion of both the top-level I/O and the global clock network. We estimate that we can reduce the failures associated with the interconnect by 90% (or 75 failures down to 7). Further, we estimate that the failures of the global clocking can be reduced from six to one.

Because there is only a single I/O pin for the top-level ports, we are not able to reduce the system failures associated with I/O failures. It is possible to tolerate temporary failures in the DDR data I/O signals if the system implemented a full 72-bit ECC interface. Using ECC, temporary single-bit errors in the data would be corrected. Unfortunately, the DDR3 interface on the Nexys Video board is only 16 bits wide, preventing any efficient implementation of ECC.

No easy solution is available for preventing the CLB failures identified in this system. Three of the four CLB related failures occured within the "reduction" voters for output I/O signals. Such single voters will remain single-point failures in the design. The fourth CLB failure is related to the corruption of two CLB clocks by a single bit. This failure could be prevented by making sure that the logic associated with two different TMR domains is not mapped within the same CLB slice. We will not be able to address the failures associated with the "Undefined" failures as we do not know their cause.

By implementing a number of design protection measures, we anticipate that we could reduce the 106 CRAM-related failures down to 23 (see Table 5). This would result in a 4.6 × reduction in failure rate of the system. Such enhancements in the reliability of the system would improve the failure rate of the TMR system to 68× the reliability of the non-TMR system. This would increase the MTTF of the TMR system from an estimated 4,554 years for the non-triplicated design, to 67,210 years for the initial TMR design, and to 309,168 years for the estimated improved design.

Device Resource	Observed Failures	Estimated Failures
Interconnect	75	8
I/O Pin	19	19
Clocking	6	1
CLB	4	3
Undefined	2	2
Total	106	23

Table 5: Estimated Reduction in System Failures with Design Improvements.

8 CONCLUSION

This work measured the neutron cross section of a TMR soft RISC-V SoC at LANSCE, and the observed improvement of MTTF was lacking compared to other published results. The BFAT tool performed a fault analysis on observed sensitive CRAM bits and identified several single point failures in the routed digital design. This work proposed several methods to potentially achieve a 4.6 × improvement of the TMR mitigation by targeting single point failures within the DDR interface, global clock network, and other vulnerable logic. Future work will use the results of this fault analysis and implement these improvements within the soft RISC-V SoC. Further radiation testing and fault injection will be used to validate this additional mitigation.

9 ACKNOWLEDGMENTS

This work was supported by the I/UCRC Program of the National Science Foundation under Grant No. 1738550. The authors are associated with the BYU site of the NSF Center for Space, High-performance, and Resilient Computing (SHREC).

REFERENCES

[1] M. Hamblen, "NASA Mars rover Perseverance launches on time Thursday to find evidence of life on Red Planet," Retrieved December 26, 2022 from https://www.fierceelectronics.com/electronics/nasa-mars-rover-perseverance-launches-thursday-to-find-evidence-life-red-planet, Jul 2020.

[2] P. Graham, M. Caffrey, J. Zimmerman, D. Eric Johnson, P. Sundararajan, and C. Patterson, "Consequences and categories of SRAM FPGA configuration SEUs," *Proc. 5th Annu. Int. Conf. Military Aerosp. Program. Logic Devices*, 01 2003.

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

[4] A. E. Wilson, M. Wirthlin, and N. Baker, "Neutron radiation testing of multiple TMR soft processors on SRAM-based FPGAs," *Accepted to IEEE Transactions on Nuclear Science 2023*, August 2022.

[5] A. E. Wilson and M. Wirthlin, "Fault injection of TMR open source RISC-V processors using dynamic partial reconfiguration on SRAM-based FPGAs," in *2021 IEEE Space Computing Conference (SCC)*, 2021, pp. 1–8.

[6] C. Brewer, N. Franconi, R. Ripley, A. Geist, T. Wise, S. Sabogal, G. Crum, S. Heyward, and C. Wilson, "NASA SpaceCube intelligent multi-purpose system for

enabling remote sensing, communication, and navigation in mission architectures," in *Small Satellite Conference 2020*, no. SSC20-VI-07, 2020.

[7] L. A. Aranda, N. J. Wessman, L. Santos, A. Sánchez-Macián, J. Andersson, R. Weigand, and J. A. Maestro, "Analysis of the critical bits of a RISC-V processor implemented in an SRAM-based FPGA for space applications," *Electronics (Switzerland)*, vol. 9, no. 1, 2020.

[8] S. D. Mascio, A. Menicucci, G. Furano, C. Monteleone, and M. Ottavi, "The case for RISC-V in space," in *International Conference on Applications in Electronics Pervading Industry, Environment and Society*. Springer, 2018, pp. 319–325.

[9] S. Di Mascio, A. Menicucci, E. Gill, G. Furano, and C. Monteleone, "Leveraging the openness and modularity of RISC-V in space," *Journal of Aerospace Information Systems*, vol. 16, no. 11, pp. 454–472, 2019.

[10] A. B. de Oliveira, L. A. Tambara, F. Benevenuti, L. A. C. Benites, N. Added, V. A. P. Aguiar, N. H. Medina, M. A. G. Silveira, and F. L. Kastensmidt, "Evaluating soft core RISC-V processor in SRAM-based FPGA under radiation effects," *IEEE Transactions on Nuclear Science*, vol. 67, no. 7, pp. 1503–1510, 2020.

[11] F. Minnella, "Protection and characterization of an open source soft core against radiation effects." Ph.D. dissertation, Polytech. Turin, 2018.

[12] H. Leppinen, "Current use of Linux in spacecraft flight software," *IEEE Aerospace and Electronic Systems Magazine*, vol. 32, no. 10, pp. 4–13, 2017.

[13] F. Kermarrec, S. Bourdeauducq, H. Badier, and J.-C. Le Lann, "Litex: an open-source SoC builder and library based on Migen Python DSL," in *OSDA 2019, colocated with DATE 2019 Design Automation and Test in Europe*, 2019.

[14] SpinalHDL, "VexRiscv," Retrieved December 26, 2022 from https://github.com/SpinalHDL/VexRiscv.

[15] SpinalHDL, "SpinalHDL," Retrieved December 26, 2022 from https://github.com/SpinalHDL/SpinalHDL.

[16] LiteX-Hub, "Linux on LiteX VexRiscv," Retrieved February 3, 2022 from https://github.com/litex-hub/linux-on-litex-vexriscv.

[17] M. J. 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 *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2016, pp. 205–214.

[18] A. Lindoso, L. Entrena, M. García-Valderas, and L. Parra, "A hybrid fault-tolerant LEON3 soft core processor implemented in low-end SRAM FPGA," *IEEE Transactions on Nuclear Science*, vol. 64, no. 1, pp. 374–381, Jan 2017.

[19] M. Psarakis, A. Vavousis, C. Bolchini, and A. Miele, "Design and implementation of a self-healing processor on SRAM-based FPGAs," in *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct 2014, pp. 165–170.

[20] A. M. Keller and M. J. Wirthlin, "Benefits of complementary SEU mitigation for the LEON3 soft processor on SRAM-based FPGAs," *IEEE Transactions on Nuclear Science*, vol. 64, no. 1, pp. 519–528, Jan 2017.

[21] N. H. Rollins, "Hardware and software fault-tolerance of softcore processors implemented in SRAM-based FPGAs," Ph.D. dissertation, Brigham Young University, Provo, UT, USA, 2012, aAI3506158.

[22] C. Hong, K. Benkrid, X. Iturbe, and A. Ebrahim, "Design and implementation of fault-tolerant soft processors on FPGAs," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2012, pp. 683–686.

[23] I. M. Safarulla and K. Manilal, "Design of soft error tolerance technique for FPGA based soft core processors," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, May 2014, pp. 1036–1040.

[24] S. Shukla and K. C. Ray, "A low-overhead reconfigurable RISC-V quad-core processor architecture for fault-tolerant applications," *IEEE Access*, vol. 10, pp. 44 136–44 146, 2022.

[25] *Microblaze Triple Modular Redundancy (TMR) Subsystem v1.0*, Retrieved December 26, 2022 from https://www.xilinx.com/support/documentation/ip_documentation/tmr\/v1_0/pg268-tmr.pdf, Xilinx, 2018.

[26] 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 *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, May 2010, pp. 47–54.

[27] D. Skouson, A. Keller, and M. Wirthlin, "Netlist analysis and transformations using SpyDrNet," in *Proceedings of the Python in Science Conference*, 2020.

[28] H. Quinn, "Challenges in testing complex systems," *IEEE Transactions on Nuclear Science*, vol. 61, no. 2, pp. 766–786, April 2014.

[29] D. S. Lee, M. Wirthlin, G. Swift, and A. C. Le, "Single-event characterization of the 28 nm Xilinx Kintex-7 field-programmable gate array under heavy ion irradiation," in *2014 IEEE Radiation Effects Data Workshop (REDW)*, 2014, pp. 1–5.

[30] P. Lisowski, C. Bowman, G. Russell, and S. Wender, "The Los Alamos national laboratory spallation neutron sources," *Nuclear Science and Engineering*, vol. 106, no. 2, pp. 208–218, 1990.

[31] M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K. LaBel, M. Friendlich, H. Kim, and A. Phan, "Effectiveness of internal vs. external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis," in *2007 9th European Conference on Radiation and Its Effects on Components and Systems*, 2007, pp. 459–466.

[32] A. Gruwell, P. Zabriskie, and M. Wirthlin, "High-speed FPGA configuration and testing through JTAG," in *2016 IEEE AUTOTESTCON*, Sep. 2016, pp. 218–225.

[33] A. E. Wilson, S. Larsen, C. Wilson, C. Thurlow, and M. Wirthlin, "Neutron radiation testing of a TMR VexRiscv soft processor on SRAM-based FPGAs," *IEEE Transactions on Nuclear Science*, vol. 68, no. 5, pp. 1054–1060, 2021.

[34] J. A. Perez-Celis, "Statistical method for extracting radiation-induced multi-cell upsets and anomalies in SRAM-Based FPGAs," Ph.D. dissertation, Brigham Young University, Provo, UT, USA, 2021.

[35] M. Rebaudengo, M. Sonza Reorda, and M. Violante, "Analysis of SEU effects in a pipelined processor," in *Proceedings of the Eighth IEEE International On-Line Testing Workshop (IOLTW 2002)*, 2002, pp. 112–116.

[36] M. Barbirotta, A. Mastrandrea, F. Menichelli, F. Vigli, L. Blasi, A. Cheikh, S. Sordillo, F. Di Gennaro, and M. Olivieri, "Fault resilience analysis of a RISC-V microprocessor design through a dedicated UVM environment," in *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2020, pp. 1–6.

[37] M. Barbirotta, A. Cheikh, A. Mastrandrea, F. Menichelli, and M. Olivieri, "Analysis of a fault tolerant edge-computing microarchitecture exploiting vector acceleration," in *2022 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, 2022, pp. 237–240.

[38] BYUCCL, "Bitstream fault analysis tool," Retrieved December 26, 2022 from https://github.com/byuccl/bfat.

[39] F4PGA, "Project X-Ray," Retrieved September 9, 2022 from https://github.com/f4pga/prjxray.

[40] L. Sterpone and L. Boragno, "Analysis of radiation-induced cross domain errors in TMR architectures on SRAM-based FPGAs," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2017, pp. 174–179.

[41] M. J. Cannon, A. M. Keller, C. A. Thurlow, A. Pérez-Celis, and M. J. Wirthlin, "Improving the reliability of TMR with nontriplicated I/O on SRAM FPGAs," *IEEE Transactions on Nuclear Science*, vol. 67, no. 1, pp. 312–320, 2020.

[42] M. J. Cannon, A. M. Keller, H. C. Rowberry, C. A. Thurlow, A. Pérez-Celis, and M. J. Wirthlin, "Strategies for removing common mode failures from TMR designs deployed on SRAM FPGAs," *IEEE Transactions on Nuclear Science*, vol. 66, no. 1, pp. 207–215, 2019.