Neutron Radiation Testing of Fault Tolerant RISC-V Soft Processors on Xilinx SRAM-based FPGAs

Andrew E. Wilson and Michael Wirthlin
NSF Center for Space, High-performance, and Resilient Computing (SHREC)
Brigham Young University
Provo, Utah, USA
{*andrew.e.wilson, wirthlin*}*@byu.edu*

***Abstract*—Many space applications are considering the use of commercial SRAM-based FPGAs over radiation hardened devices. When using SRAM-based FPGAs, soft processors may be required to fulfill application requirements, but the FPGA designs must overcome radiation-induced soft errors to provide a reliable system. TMR is one solution in designing a fault tolerant soft processor to mitigate the failures caused by SEUs. This paper compares the neutron soft-error reliability of an unmitigated and TMR version of a Taiga RISC-V soft processor on a Xilinx SRAM-based FPGA. The TMR RISC-V processor showed a 33× reduction in the neutron cross section and a 27% decrease in operational frequency, resulting in a 24× improvement of the mean work to failure with a cost of around 5.6× resource utilization.**

***Keywords*-RISC-V; Fault tolerance; redundancy; Triple Modular Redundancy (TMR); Single Event Upset (SEU); fault injection; radiation testing; FPGA; soft processor**

I. Introduction

SRAM-based FPGAs can provide custom hardware systems using a vast quantity of reprogrammable resources. Such FPGAs contain lookup tables (LUTs), flip-flops (FFs), high-speed I/O, digital signal processing (DSP) units, and block RAM to meet the needs of the designer. As SRAM-based FPGAs can be fully or partially reprogrammed an unlimited amount of times, they have been used for prototyping, as interconnections between different components, and for deploying accelerated hardware solutions.

A RISC-V processor can be implemented as a soft processor within the FPGA configurable logic as defined by a hardware description language (HDL) [1]. Soft processors can be used when a portion of the design proves too complex for a simple state machine and/or there is a need for a software-controlled system. These soft processors can be customized to the given application and provide a hybrid software and hardware system with very little overhead of the FPGA resources. The use of a soft processor allows additional support of already existing software tool-chains and libraries. These established software libraries can be utilized to shorten development time and aid in the operation of the system. FPGAs implementing soft processors can provide great flexibility and performance for applications in both terrestrial and space environments.

Radiation found in both space and terrestrial environments can prove hazardous to SRAM-based FPGAs and the soft processors implemented within. This radiation can cause single event upsets (SEUs) that flip bits in configuration RAM (CRAM) and block RAM (BRAM) [2]. SEUs can cause functional failures in the design within the FPGA by corrupting the state and circuit configuration. As a soft processor's functionality is defined by both the hardware (CRAM) and software (BRAM), SEUs can produce unpredictable and unwanted results that may lead to a critical failure of the system. This hazard must be taken into consideration when developing custom processors on FPGAs for highly-reliable systems. The use of SEU mitigation may be required to improve the functional reliability [3] to meet the required allowable risk of the application.

Triple modular redundancy (TMR) is an effective mitigation technique to improve the reliability of SRAM-based FPGA soft processor designs in harsh radiation environments [4]. TMR uses redundant copies and voters to mask errors that would cause functional errors within the system. Though TMR provides a great improvement in reliability, it comes at a cost of greater power consumption, higher resource utilization, and slower operational frequency. If more than one redundant module fails, TMR cannot provide fault mitigation and the system fails. To prevent an accumulation of SEUs, the CRAM is scrubbed to a known good state without disrupting the operation of the FPGA. To repair any corrupted state within the processor, the state is resynchronized with the output of the other redundant modules.

This paper describes the implementation of a Taiga RISC-V soft processor [5] targeted to a Xilinx SRAM-based FPGA and compares the improvement in reliability of a TMR version to that of the original, unmitigated design. The goal of this experiment is to understand the difference in the neutron cross section of a TMR RISC-V system against a conventional design. Using the BL-TMR tools developed at Brigham Young University [6], the TMR processor showed

This work was supported by the I/UCRC Program of the National Science Foundation under Grant No. 1738550.

a 33× reduction in the neutron cross section and a 27% decrease in operational frequency, resulting in a 24× improvement of the mean work to failure with a cost of around 5.6× resource utilization.

II. RISC-V Background

RISC-V is a promising new instruction set architecture (ISA) that is royalty free, allowing for free use in academia, research, and industry. Its development began in 2010 at the University of California, Berkeley [7] and has been maintained by a non-profit organization with an ever-growing membership of individuals and companies [8]. The RISC-V foundation and its 200+ members have and are currently defining and ratifying specifications for user-level instructions, privileged instructions, debug protocols, and memory models.

RISC-V has specifications for 32-bit, 64-bit, and 128-bit processors with established extensions and opportunity for custom application-specific instructions. These extensions include compressed, integer multiplication and division, atomic, and floating-point instructions. RISC-V is already supported by a rich pool of software libraries, tools, and operating systems. There are many open-source implementations available for use as soft processors on FPGAs, such as the PicoRV32 [9], Orca [10], VexRISCV [11], and SerRV EH1 [12]. This ISA proves to be a strong candidate as a target for a fault tolerant soft processor with its growing support and wide adoption.

Taiga is a 32-bit RISC-V processor designed specifically to be used as a soft processor for Intel and Xilinx SRAM-based FPGAs [13]. This processor was developed for research in heterogeneous processor systems and high performance soft processors. The processor, in addition to the base instruction set, also supports the multiply/divide and atomic operations extensions (RV32IMA). The Taiga processor, implemented in SystemVerilog, supports configurable options such as caches, multiple bus standards, translation lookaside buffers (TLBs), and memory management units (MMUs). The processor is designed to support Linux-based shared-memory systems.

Figure 1 shows a diagram of the pipelined processor with multiple independent execution units. For each execution unit, the upper number is the cycle latency for that type of instruction. The number below is the rate at which the unit can start additional requests (the initiation interval). When multiple numbers are present, there can be multiple latencies, and for units with variable latencies, the numeric value of the minimum latency is given in conjunction with a plus symbol. Additional execution units would have no restrictions on latency. This allows for easy integration of new functional units into the processor.

The Taiga processor design used approximately 33% fewer slices while clocking 39% faster than a LEON3-based system built on a Xilinx Zynq X7CZ020 [5]. This processor

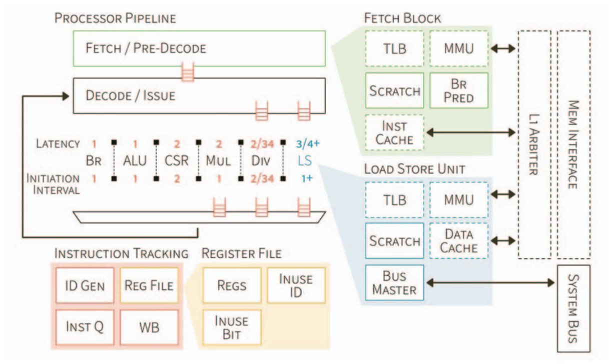

Figure 1. Taiga Overview and Pipeline Structure. [13, Fig. 2].

proves to be a capable soft processor for SRAM-based FPGAs. It is easy to implement in a design and develop software using the RISC-V software libraries.

III. Fault Tolerant RISC-V Soft Processor

Soft processors have been implemented in SRAM-based FPGAs for operation in harsh radiation environments, and many efforts have been made to improve the reliability of these processors. Techniques in detecting, recovering from, and mitigating against these SEU failures have been developed both in hardware and software. Though these techniques improve reliability, they often come at a cost in resource utilization, power consumption, and performance.

A. *Fault Tolerant Soft Processors*

In many of the efforts to produce a fault tolerant soft processor, the LEON2 and LEON3 processors were targeted and modified to provide improved reliability [14]–[16]. One study reported a neutron radiation test resulting in a 27× improvement for a TMR LEON3 with CRAM scrubbing and a 50× improvement with both CRAM and BRAM scrubbing [17]. In addition, software techniques have been used to improve the detection of failures and provide adequate recovery without compromising the system. These studies have shown an improvement with much less overhead than TMR requires [18]. A combination of hardware and software techniques can be used to improve the reliability of soft processors.

Other work has looked at implementing TMR with the Picoblaze [19], [20], a free 8-bit soft processor provided by Xilinx. TMR, error correction codes (ECC), and CRAM single error correction (SEC) are used to provide a fault tolerant processor with the ability to detect and recover from errors. Xilinx also offers a Microblaze TMR Subsystem for use within their FPGAs [21]. This subsystem includes the TMR Microblaze with a soft error mitigation (SEM) core to perform CRC checks and SEC on the configuration memory.

B. TMR RISC-V

TMR is a fault tolerance technique that can be used for SEU mitigation to improve the reliability of the RISC-V soft processor. Within the soft processor, three redundant domains and voters are used to mask any failures within one of the redundant domains (see Figure 2). Each domain is provided the same input stimulus and will have the same output during correct operation. If one domain is corrupted, its output may not match the corresponding output of the other domains. A majority voter is used to mask the erroneous output and produce the result that is agreed upon by the other two domains [22]. The voters can be triplicated as well to reduce single points of failure within the design. With this, TMR is able to mask any failure of a single domain. The design can be partitioned down to the FPGA primitives (i.e. LUTs, flip-flops, BRAM cells) and voters inserted between partitions to decrease the size of each partition and improve reliability [23]. Alternately, the processor can be triplicated at the module level to maintain the same performance with a lower cost to utilization, but with only one voter and TMR partition.

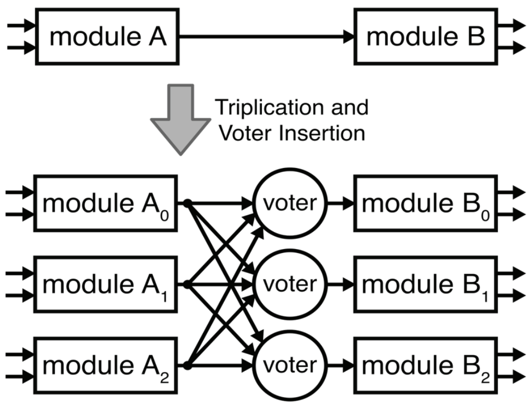

Figure 2. TMR.

The TMR RISC-V processor needs a repair mechanism to continually recover the system from the masked errors in order to prevent multiple TMR domains from failing. A processor using TMR without any repair will have higher reliability early in its operation as it masks the first few errors, but as the TMR design continues to collect errors, its reliability will eventually become worse than the original design. This is due to the increase in circuit size as redundant modules and voters are introduced, thus creating a bigger target.

To repair any SEU upsets in the configuration memory of an SRAM-based FPGA, CRAM scrubbing is employed to repair the configuration bit upset and prevent the accumulation of errors. Configuration scrubbing is performed by rewriting the original FPGA bitstream to scrub any incorrect values [24]. CRAM scrubbing does not affect the operation of the FPGA nor interfere with the BRAMs.

The dynamic memory of the TMR processor, such as registers and caches, may also become corrupt in one of the TMR domains. The voters use feedback loops to resynchronize the state of the correct TMR domains to all the domains on the next cycle. This allows any corrupted state to automatically be repaired each cycle.

Though the soft processor is susceptible to SEUs, with the use of mitigation, the reliability can greatly be improved. If only one domain of the TMR is affected and the system is able to repair itself before another domain fails, the system can maintain operation. The possible improvement to reliability is limited by the scrub rate, imperfections in tools allowing for single point of failure CRAM bits, and multiple bit upsets that affect multiple domains.

The BL-TMR tool has shown results of 50-100$\times$ improvement in reliability despite these limitations [17]. This tool automates the process of triplicating the design and adding triplicated voters with necessary feedback to resynchronize any state. It performs fine-grained TMR by triplicating all FFs, LUTs, BRAMS, and DSPs and inserting voters between these primitives. Xilinx-specific primitives such as the Mixed-Mode Clock Manager and clock buffer are ignored in this setup. The tool's input is a vendor-independent electronic design interchange format (EDIF) file such as can be exported from Xilinx Vivado. The tool then triplicates the design and inserts the needed voters. The finished TMR EDIF file can be imported into Xilinx Vivado as a post-synthesis file, whereupon the vendor place and route tools can generate a full bitstream.

IV. Experimental Design

To fully utilize the neutron radiation test, the design was scaled up to contain as many processors as possible. This paper introduces two experimental designs implemented on the Xilinx Kintex Ultrascale KU040 FPGA. These designs used the Taiga RISC-V Processor, one containing 20 unmitigated processors and the other with 20 TMR processors. These designs were irradiated in a neutron beam in order to induce faults. The purpose of this test was to understand the baseline cross section of the unmitigated processor and compare it to the reduction in cross section of the TMR processor. The cross section is the ratio between the number of SEUs that cause failures and the amount of radiation fluence the device was exposed to. The larger the calculated cross section, the more sensitive the design is to SEU upsets.

This experiment implemented Taiga RISC-V processors on a Kintex Ultrascale KCU105 development board with the XCKU040-2FFVA1156E FPGA. The base Taiga design was provided through an open source repository [25]. During this experiment, the processor did not use any caches, TLBs, or MMUs, which were removed from the configuration. A 16KB dual-ported BRAM was used for both the instruction

Table I
COMPARISON OF SINGLE PROCESSOR TEST DESIGNS

Single Processor Test Utilization					
Design	LUT	FF	BRAM	DSP	FMAX
Taiga Processor	1954 (0.80%)	1044 (0.19%)	6 (1.00%)	4 (0.21%)	227.2 MHz
Unmitigated Test Design	5173 (2.13%)	7606 (1.57%)	6 (1.00%)	4 (0.21%)	227.2 MHz
TMR Test Design	29163 (12.03%)	22818 (4.71%)	18 (3.00%)	12 (0.63%)	165.0 MHz
Cost Ratio	5.64×	3.0×	3.0×	3.0×	0.73×

Table II
COMPARISON OF 20 PROCESSOR TEST DESIGNS

20-Processor Test Utilization					
Design	LUT	FF	BRAM	DSP	FMAX
Unmitigated	43350 (17.88%)	36037 (7.43%)	120 (20.00%)	80 (4.17%)	202.4 MHz
TMR	222029 (91.60%)	108021 (22.28%)	360 (60.00%)	240 (12.50%)	149.3 MHz
Cost Ratio	5.12×	3.0×	3.0×	3.0×	0.74×

and data memory. The utilization of a single processor with this configuration is reported in Table I. The CRAM utilization is represented by the LUTs and FFs. The utilization of the unmitigated design in Table I shows the additional resources required for the test hardware.

To verify the functionality of the processor, a Dhrystone benchmark was executed and the results were compared. This RISC-V Dhrystone benchmark was developed by the RISC-V foundation and is available through their online repositories [26]. The benchmark provides a functional test (though not full verification) of the integer instructions on the processor. A checksum was computed using all the Dhrystone calculated values. The processors continually looped through the execution of the Dhrystone benchmark.

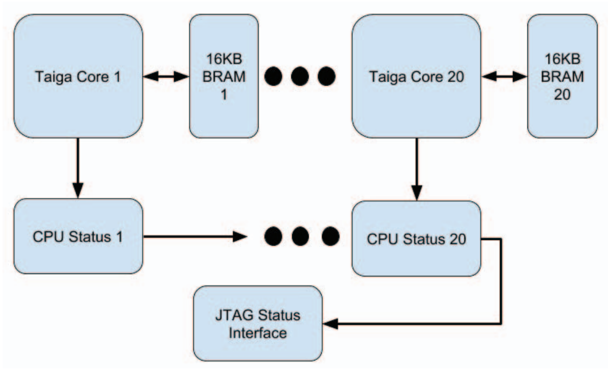

Figure 3. A block diagram of the experimental design.

For every iteration of the benchmark, the checksum and iteration count were reported as the CPU status over a JTAG interface. This CPU status was compared to a golden value to confirm correct operation. The JTAG reporting interface was updated to allow messages to be passed through a large shift register. This made it easy to add as many processors as needed. Figure 3 shows a simple block diagram of how the multiple processors were connected to the JTAG interface.

Using the BYU BL-TMR tools, the Taiga processor was triplicated, and voters were inserted into the design. The comparison of utilization of the unmitigated design and the TMR version are reported for both the single processor design (see Table I) and for the 20 processor design (see Table II). The final experiment of 20 Taiga RISC-V processors used about 20% of the resources available on the FPGA (see Figure 4). The TMR designs resulted in about a 5× increase of LUTs; a 3× increase of FFs, BRAMs, and DSPs; and a 27% decrease in operation frequency. The greater

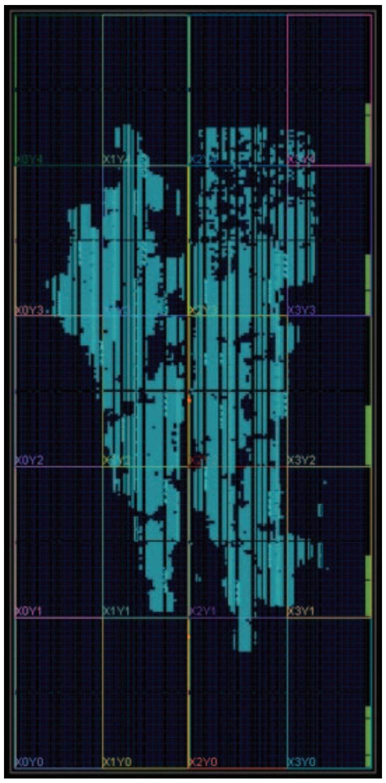

Figure 4. The floorplan for the unmitigated design of 20 Taiga RISC-V processors.

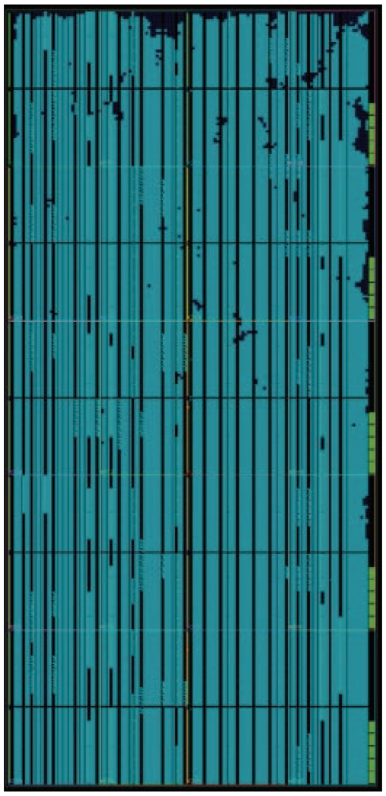

Figure 5. The floorplan for the TMR design of 20 Taiga RISC-V processors.

increase of LUTs over FFs is due to the triplicated voters and resynchronization logic. Though the chosen FPGA could contain 100 unmitigated processors (limited by BRAM), 20 processors was the greatest amount that could be triplicated within the FPGA. The 20 TMR processor design used about 90% of the available LUTs in the FPGA (see Figure 5).

For both of the experimental designs, configuration scrubbing was implemented over a 50 MHz JTAG interface. This active repair prevents multiple upsets from accumulating and allows TMR to operate without multiple failures. The configuration scrubbing was performed by an external device known as the JTAG Configuration Manager (JCM), which has been developed at Brigham Young University [27].

V. Neutron Radiation Test

These two experimental designs were tested with a neutron radiation beam at the Los Alamos Neutron Science Center (LANSCE). This wide spectrum neutron beam is commonly used for testing of integrated circuits to estimate circuit sensitivity to terrestrial neutrons [28]. The Kintex Ultrascale KCU105 development board was aligned to the beam, as depicted in Figure 6. The board contains an XCKU040-2FFVA1156E FPGA, fabricated using 20-nm technology. Each of the designs were tested at a normal angle of incidence and at room temperature. The neutron

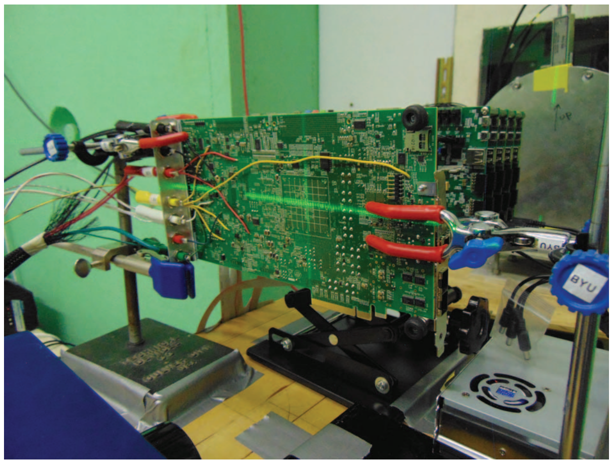

Figure 6. KCU105 Setup at the Neutron Radiation Test.

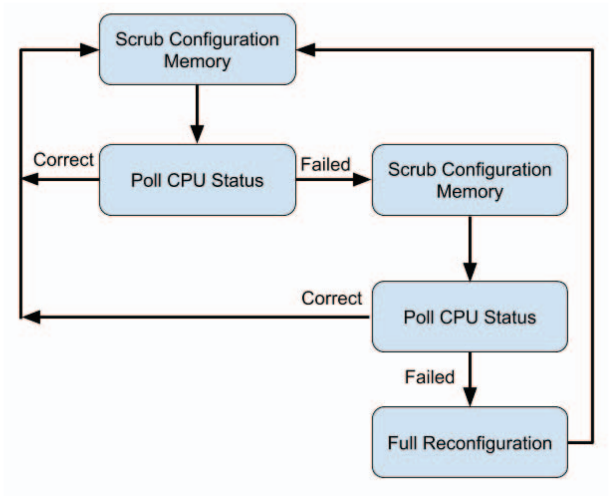

Figure 7. Radiation Testing Flow

beam was collimated to 2 inches.

In previous tests, the board's on-board power regulators proved to be highly susceptible to failure in the radiation test. To overcome this issue, connections were soldered to the board to bypass the power regulators, and the board was powered by an external power source that maintained the correct voltages needed by the FPGA.

Using the JTAG interface on the development board, the JCM was able to interact with the FPGA during the radiation test and perform the tasks shown in Figure 7. The JCM managed the configuration of the FPGA, scrubbed the CRAM, reported CRAM upsets, and polled the status of the processors. The JCM used a 50 MHz JTAG clock to perform a readback of the configuration bits and repair any bits that were upset by using a golden copy of the bitstream. Using the JTAG interface, the JCM would poll the status of each of the 20 processors and determine whether they matched the correct checksums for the Dhrystone and whether the

Table III
NEUTRON RADIATION DATA

Design	Fluence (n/cm^2)	Observed CRAM Upsets	Failures	Cross Section (cm^2)	Cross Section Per Processor	+95% Confidence -95% Confidence	Reduction
20 Processors Unmitigated	1.15×10^{10}	2527	52	4.54×10^{-9}	$2{,}27 \times 10^{-10}$	2.44×10^{-10} 2.10×10^{-10}	1×
20 Processors TMR	2.00×10^{11}	52139	27	1.35×10^{-10}	6.76×10^{-12}	9.81×10^{-12} 4.45×10^{-12}	33×

processors progressed in Dhrystone iteration. If a processor was reported as failing the check, an additional scrub cycle and status poll were performed. If a processor continued to perform incorrectly, the entire device was reconfigured, restarting all 20 processors. The total duration for either a full readback or configuration of the CRAM was about 4 seconds. No other recovery methods were used during this test.

VI. TEST RESULTS

During the neutron test, the unmitigated design reported 52 errors with a total fluence of 1.15×10^{10}, while the TMR design reported 27 errors with a total fluence of 2.00×10^{11}. Table III shows the observed upsets of the configuration bits and the number of failures of the processors running the Dhrystone benchmark. Though the BRAM memory was triplicated in this design, there was no way to observe any upsets within the memory. The fluence for the test was calculated using modifiers for the beam degradation in regards to distance and any board degradation of any other experiments in front of this device.

The cross section is the ratio between the failures and the total fluence the device was exposed to. The effectiveness of the TMR is represented by the reduction in the cross section. Table III shows the total neutron cross section for both designs. The estimated cross sections per processor with 95% confidence intervals [29] are also reported within the table. There was a 33× reduction in the neutron cross section between the unmitigated and TMR designs. With this reduction in neutron cross section and the 27% decrease in operational frequency, the TMR processor achieved a 24× improvement of the mean work to failure.

Table IV
TYPES OF UPSETS DURING PROCESSOR FAILURES

	Single Bit	Multi-Bit	Unobserved	Total
Non TMR	18	15	19	52
TMR	10	6	11	27

The processor failures fall into three categories: single bit upsets, multi-bit upsets, and unobserved upsets. Table IV shows the frequency of these different failures during the radiation test. Single bit upsets identify possible single point failures within the TMR design. These single point failures could be related to the FPGA's routing of the data, clocks, and resets. Though the TMR design may be able to mask multi-bit upsets, it is not guaranteed to mitigate all multi-bit upsets. Processor failures that showed no observable upsets could be potential corruption in the BRAM or a single event functional interrupt (SEFI) of the FPGA.

The improvement of the TMR design was limited by single points of failure, multi-bit upsets, and a lack of protection for the BRAM memory. Other research is being done to understand and overcome the limitations of TMR when implemented on SRAM-based FPGAs [30], [31]. Future work will also consider additional techniques to improve the reliability of the RISC-V soft processor and recover from these various types of failures.

VII. CONCLUSION

As soft processors on SRAM-based FPGAs are used in space applications and other radiation-hazardous environments, their reliability and risk need to be understood. This paper has provided a baseline study of the neutron cross section of a Taiga RISC-V processor with a comparison to a TMR version. The successful radiation test of the TMR design showed a 33× reduction in the neutron cross section and a 27% decrease in operational frequency, resulting in a 24× improvement of the mean work to failure with a cost of around 5.6× resource utilization. Though this test represented a terrestrial environment, these results are encouraging and motivate more exploration in the performance and reliability of soft processors and their applications in extra-terrestrial radiation hazardous environments.

Future work will compare hardware and software techniques in detecting, recovering, and mitigating errors for the development of a highly fault tolerant soft processor. Continuing research in neutron and higher energy radiation experiments as well as emulated fault injection tests will verify these techniques and their effectiveness. Additional BRAM protection techniques, such as ECC and active scrubbing, will be considered. This future research will also explore more demanding software and complex extensions of the RISC-V ISA. New versions of the Taiga RISC-V processor will help in developing a highly fault tolerant soft processor capable of supporting operating systems such as Linux for space applications.

REFERENCES

[1] J. G. Tong, I. D. L. Anderson, and M. A. S. Khalid, "Soft-core processors for embedded systems," in *2006 International Conference on Microelectronics*, Dec 2006, pp. 170–173.

[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. M. 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.

[4] 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.

[5] E. Matthews and L. Shannon, "Taiga: A new risc-v soft-processor framework enabling high performance cpu architectural features," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2017, pp. 1–4.

[6] B. Pratt, M. Caffrey, P. Graham, K. Morgan, and M. Wirthlin, "Improving fpga design robustness with partial tmr," in *2006 IEEE International Reliability Physics Symposium Proceedings*, March 2006, pp. 226–232.

[7] A. Waterman, Y. Lee, D. Patterson, and K. Asanovic, *The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA*. EECS Department, University of California, Berkeley: Technical Report UCB/EECS-2011-62, 5 2011.

[8] (2019) Members at a glance. RISC-V Foundation. [Online]. Available: https://riscv.org/members-at-a-glance/

[9] C. Wolf, "Picorv32," https://github.com/cliffordwolf/picorv32, 2019.

[10] VectorBlox, "Orca," https://github.com/VectorBlox/orca, 2019.

[11] SpinalHDL, "Vexriscv," https://github.com/SpinalHDL/VexRiscv, 2019.

[12] W. D. Corporation, "Swerv eh1," https://github.com/westerndigitalcorporation/swerv_eh1, 2019.

[13] E. Matthews, Z. Aguila, and L. Shannon, "Evaluating the performance efficiency of a soft-processor, variable-length, parallel-execution-unit architecture for fpgas using the risc-v isa," in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, April 2018, pp. 1–8.

[14] 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*, ser. FPGA '16. New York, NY, USA: ACM, 2016, pp. 205–214. [Online]. Available: http://doi.acm.org/10.1145/2847263.2847278

[15] A. Lindoso, L. Entrena, M. Garca-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.

[16] 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.

[17] 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.

[18] N. H. Rollins, "Hardware and software fault-tolerance of softcore processors implemented in sram-based fpgas," Ph.D. dissertation, Provo, UT, USA, 2012, aAI3506158.

[19] 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.

[20] 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.

[21] *Microblaze Triple Modular Redundancy (TMR) Subsystem v1.0*, https://www.xilinx.com/support/documentation/ip_documentation/tmr/v1_0/pg268-tmr.pdf, Xilinx, 10 2018.

[22] J. M. Johnson and M. J. Wirthlin, "Voter insertion algorithms for fpga designs using triple modular redundancy," in *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '10. New York, NY, USA: ACM, 2010, pp. 249–258. [Online]. Available: http://doi.acm.org/10.1145/1723112.1723154

[23] M. Niknahad, O. Sander, and J. Becker, "Fgtmr - fine grain redundancy method for reconfigurable architectures under high failure rates," in *The 16th North-East Asia Symposium on Nano, Information Technology and Reliability*, Oct 2011, pp. 186–191.

[24] J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb, "Fpga partial reconfiguration via configuration scrubbing," in *2009 International Conference on Field Programmable Logic and Applications*, Aug 2009, pp. 99–104.

[25] E. Matthews, "Taiga," https://gitlab.com/sfu-rcl/Taigas, 10 2018.

[26] R.-V. Foundation, “riscv-tests,” https://github.com/riscv/riscv-tests, 2019.

[27] A. Gruwell, P. Zabriskie, and M. Wirthlin, “High-speed fpga configuration and testing through jtag,” in *2016 IEEE AUTOTESTCON*, Sep. 2016, pp. 1–8.

[28] P. W. Lisowski, C. D. Bowman, G. J. Russell, and S. A. Wender, “The los alamos national laboratory spallation neutron sources,” *Nuclear Science and Engineering*, vol. 106, no. 2, pp. 208–218, 1990. [Online]. Available: https://doi.org/10.13182/NSE90-A27471

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

[30] M. Cannon, A. Keller, and M. Wirthlin, “Improving the effectiveness of tmr designs on fpgas with seu-aware incremental placement,” in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, April 2018, pp. 141–148.

[31] M. J. Cannon, A. M. Keller, H. C. Rowberry, C. A. Thurlow, A. Prez-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, Jan 2019.