Fault Injection of TMR Open Source RISC-V Processors using Dynamic Partial Reconfiguration on 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*

2021 IEEE Space Computing Conference (SCC) | 978-1-6654-2400-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/SCC49971.2021.00008

***Abstract*—SRAM-based FPGAs are frequently used for critical functions in space applications. Soft processors implemented within these FPGAs are often needed to satisfy the mission requirements. The open ISA, RISC-V, has allowed for the development of a wide range of open source processors. Like all SRAM-based FPGA digital designs, these soft processors are susceptible to SEUs. This paper presents an investigation of the performances and relative SEU sensitivity of a selection of newly available open source RISC-V processors. Utilizing dynamic partial reconfiguration, this novel automatic test equipment rapidly deployed different implementations and evaluated SEU sensitivity through fault injection. Using BYU's new SpyDrNet tools, fine-grain TMR was also applied to each processor with results ranging from a 20× to 500× reduction in sensitivity.**

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

I. Introduction

Static random-access memory (SRAM)-based field programmable gate arrays (FPGAs) often implement soft processors within the digital designs to perform critical functions. These soft processors are implemented using the FPGA's reprogrammable resources such as lookup tables (LUTs), flip-flops (FFs), digital signal processing (DSP) units, and block RAM (BRAM). The configurable soft processors provide a software platform coupled with the hardware description language (HDL)-defined hardware solution. Using an open source soft processor implementing an open instruction set architecture (ISA), such as RISC-V, allows for the integration of established software tools and libraries. Implementing these soft processors in FPGAs can be beneficial for applications in both terrestrial and space environments.

To provide a sufficiently reliable system in a radiation hazardous environment, mitigation may be required to improve the functional reliability of the digital design [1]. Energized particles can cause single event upsets (SEUs) that flip bits in configuration RAM (CRAM) and BRAM [2]. These events can produce unpredictable and unwanted results, potentially leading to a critical failure of the system. Single point failures caused by SEUs can be masked with triple modular redundancy (TMR) [3]. The improvement in reliability provided by TMR comes at a cost of greater power consumption, higher resource utilization, and slower maximum operational frequency.

Random CRAM fault injection can be used to observe the reduction achieved by TMR mitigation in sensitive CRAM bits causing critical failure. Fault injection emulates the effects of an SEU and identifies which single bit upsets result in a functional failure. While previous experiments used JTAG fault injection [4]–[6], this paper utilizes the internal configuration access port (ICAP) and dynamic partial reconfiguration available on Xilinx FPGA devices to rapidly collect fault injection data. Automatic test equipment (ATE) is implemented in the static portion of the FPGA that is capable of testing several designs under test (DUTs) in different dynamic partial regions.

This paper reports the results achieved by this ATE design injecting faults into five different soft processors. Unmitigated and TMR versions of the processors are implemented for a total of ten designs. Each device contained three partial regions that could contain any of the ten DUTs. Utilizing dynamic partial reconfiguration, this novel ATE was capable of rapidly switching out the different DUTs and collecting fault injection data.

This paper compares the utilization and performance of newly available RISC-V processors running a Dhrystone benchmark. The results of this paper report on the CRAM sensitivity of each design and the reduction achieved by TMR mitigation. While previous fault injection experiments of TMR RISC-V soft processors achieved a 10× [6] to 32× [4], [5] reduction in CRAM sensitivity, this paper uses BYU's new SpyDrNet tools to apply TMR [7] and observed a range in improved CRAM sensitivity from 20× to 500×.

The remainder of this paper is organized as follows. Section II presents background information on fault tolerant RISC-V soft processors, TMR, and fault injection. Section III describes the test hardware, including the ATE and DUTs. The fault injection testing and results are detailed in Section IV. Section V concludes the paper.

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

978-1-6654-2400-4/21/$31.00 ©2021 IEEE
DOI 10.1109/SCC49971.2021.00008

II. Fault Tolerant RISC-V Soft Processors

RISC-V is an open ISA that is freely used in academia, research, and industry. There are many open source RISC-V processors available for implementation within FPGAs ranging widely in features, performance, and utilization of FPGA resources. One study has shown the difference of a small selection of RISC-V soft processors in performance and maximum frequency achieved with SRAM-based FPGA implementations [8]. These results showed the Taiga [9] and VexRiscv [10] among the best performing processors for SRAM-based FPGAs. Space applications have tight constraints that require the best performance for the FPGA resources utilized and power consumed.

Related works have investigated the fault tolerance of RISC-V processors. Some studies have used the Rocket Chip implementation, officially supported by RISC-V [11], to study the fault tolerance of the RISC-V architecture. One study characterized this processor against SEUs using fault injection with Xilinx's SEM IP [12] showing that the RocketChip was more sensitive than other soft processors explored in previous works. Another study applied Razer, a timing fault detection and recovery tool, to the Rocket Chip [13]. Using a different RISC-V implementation, other related work showed that implementing Linux OS on the LowRISC processor did not significantly impact the processor's reliability [14]

Previous work has performed fault injection on the Taiga [9] and VexRiscv [10] processors. This paper uses these RISC-V implementations in addition to the PicoRV32 [15] and Kronos [16] implementations. This paper presents an ATE platform to compare the CRAM sensitivity of these different soft processors with identical interfaces and placement constraints.

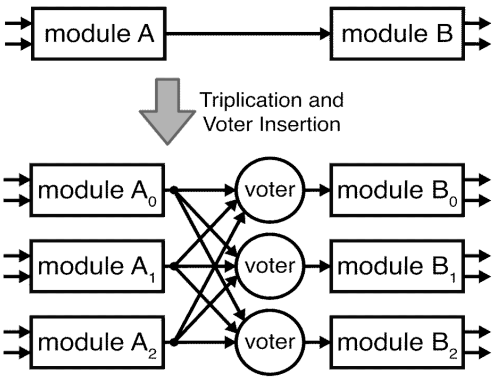

Figure 1: TMR with triplicated voters.

A. Triple Modular Redundancy

FPGA-implemented RISC-V soft-cores can be mitigated against SEUs by using TMR techniques to provide redundancy to the design. The TMR soft processor includes three redundant domains and triplicated voters capable of masking a failure of a single redundant domain (see Figure 1). The triplicated majority voters mask erroneous output and use the majority output of the other two domains [17]. Feedback loops can be introduced to use the voted upon value as next state logic for any flip-flops. TMR mitigation can be applied to the top block level (coarse-grain) or distributed within the design down to the FPGA primitives (fine-grain). Whether implementing TMR mitigation on the full or partial digital design, a repair mechanism (such as CRAM scrubbing) is essential in preventing multiple TMR domain failure and drastically improving the effectiveness of TMR [18].

Many different soft-core processors implementing TMR have been investigated for space application. Several studies have modified the LEON2 and LEON3 processors for improved reliability with different mitigation schemes including TMR [19]–[23]. Other work has applied TMR to the Picoblaze [24], [25], a free 8-bit soft processor provided by Xilinx. Newer works have targeted RISC-V processors such as the Rocket Chip. One work used the Mentor Precision Hi-Rel tool to apply fine-grain TMR and performed fault injection with Xilinx's SEM IP and observed a reduction in sensitive bits of up to 11.5× [26]. Another work used Cadence's EDA [27] flow to apply fine-grain TMR and used custom HDL to inject faults through ICAP and achieved a 3× reduction in the cross section during heavy-ion testing [28]. Other work has also targeted the VexRiscv processor using Synplify's TMR tools and observed a 1.5× improvement in the mean time to failure (MTTF) with JTAG fault injection [29].

This paper uses SpyDrNet, a new Python-based netlist tool developed at BYU, to generate the TMR designs [7]. This tool is discussed in greater detail within subsection III-C. The tool generated netlists for four different RISC-V TMR designs and a TMR MicroBlaze design, the Xilinx provided soft processor [30]. These netlists are then placed and routed within each of the partial regions used for DUT testing. The results in this paper showed this TMR tool achieved a 20× to 500 × reduction in CRAM sensitivity for complex processor designs.

B. Fault Injection with Partial Regions

Fault injection is used to emulate CRAM upsets within an SRAM-based FPGA [31]. By interfacing with the Xilinx FPGA configuration manager, CRAM frames can be read, modified, and written back to the device during operation. After the emulated upset, the DUT can be tested to verify operation. This approach can be used to understand how mitigation techniques such as TMR would behave under the effects of radiation-induced faults before committing to expensive and limited radiation testing. Fault injection is only capable of emulating the CRAM SEU subset of the possible single event effects (SEEs) and does not take

into account any non-CRAM SEUs, single event transients (SETs), or single event functional interrupts (SEFIs).

ICAP-based fault injection into partial regions has been used to perform rapid injections up to an 8× speedup [32] and precise injections into sensitive bits identified by Xilinx Tools [33]. Dynamic partial reconfiguration has been used to inject faults within the FFs [34]. Performing fault injection within partial regions has been also used to validate implemented mitigation methods [35] such as forward temporal redundancy [36].

This paper used dynamic partial reconfiguration to place multiple isolated DUTs on the same FPGA. Utilizing the ICAP, faults are injected into each DUT and testing is performed simultaneously. Taking advantage of partial reconfiguration, different DUTs are loaded into the partial regions to compare different implementations and mitigation techniques. The static ATE acts a constant testing platform able to configure, inject, and test the different DUTs.

III. Test Hardware

The test hardware was implemented on M.2 PCIe capable SQRL Acron CLE 215+ FPGA accelerator cards [37]. This device's schematic is identical to the open-source Nitefury and Litefury devices [38]. The FPGA is a Xilinx Series 7 Artix device with the xc7a200tfbg484-3 part. The experimental design utilizes dynamic partial reconfiguration to define ATE within the static portion and to be able to swap out different DUTs in the dynamic regions. This section will describe these different parts of the experimental design.

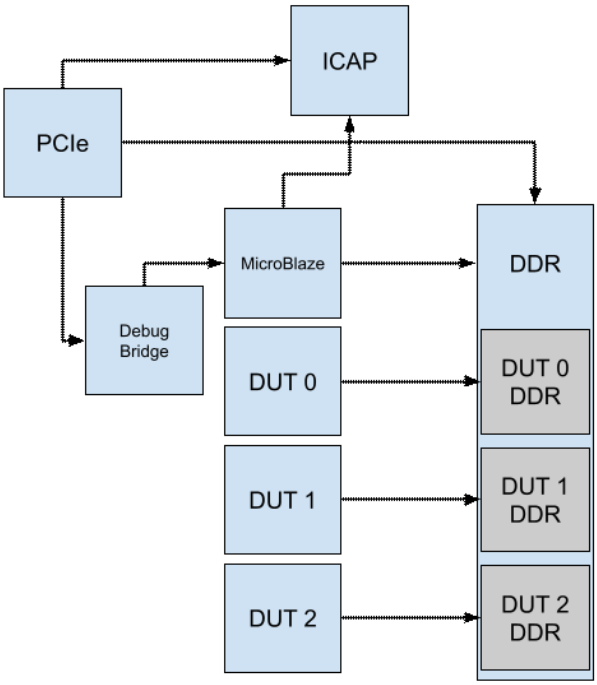

Figure 2: A block diagram for static ATE design.

A. Static Automatic Test Equipment

The ATE takes advantage of a high speed ICAP (over 100× faster than JTAG) and can rapidly test different DUTs without any change to the static design, allowing for rapid and versatile fault injection. The ATE performs these three main tasks: performs partial reconfiguration to load in different DUTs into the partial regions, injects random faults into the partial regions, and verifies the output of each DUT. The main components of the static ATE are a PCIe interface, MicroBlaze, debug bridge, ICAP, and DDR controller.

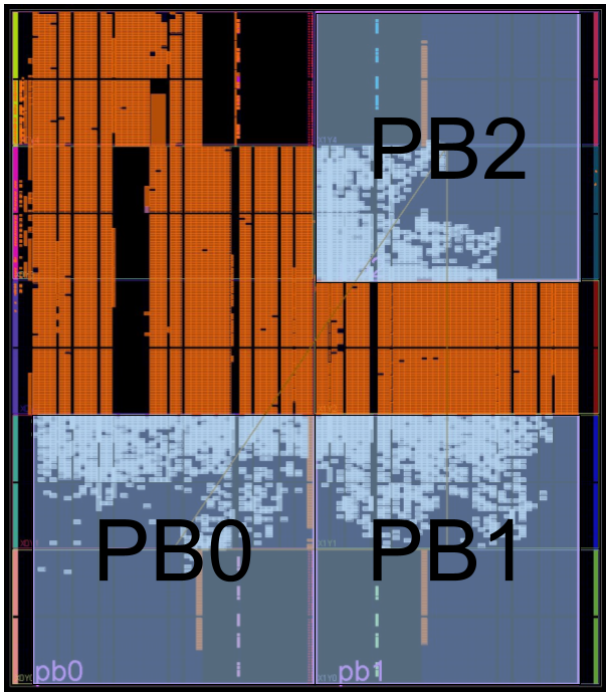

Figure 3: The floorplan for static ATE design.

Figure 2 shows the connections between these various components. The PCIe connection allows for a host to initiate a warm boot over ICAP, load test data into DDR, operate the static MicroBlaze, and retrieve test results. The MicroBlaze runs testing software that resets the DUTs, performs partial reconfiguration and fault injection through the ICAP, and has full access to the DDR address space. The debug bridge loads the testing software into the static MicroBlaze. The DDR is one gigabyte, with 256 megabytes allocated to the MicroBlaze and each DUT. Figure 3 shows the floor plan for the FPGA design with the static ATE region highlighted in orange. The static region's location was dependent on the PCIe, ICAP, and DDR locations.

Multiple ATE platforms can be hosted from the same computer. Additional software is executed on the the host machine to perform the fault injection tests. The setup used for this experiment connected 20 ATE platforms to the same host machine, Figure 4. With this setup, multiple instances

Figure 4: The host machine with 20 FPGAs.

could be executed simultaneously to rapidly collect more data.

Table I: Pblock Sizes

Pblock	LUTs	LUTRAM	FF	BRAM	Total CRAM Frame Bits
Device	133800	46200	267600	365	59145600
PB0	19.7%	20.6%	19.7%	22%	9321088 (15.8%)
PB1	19.1%	20.6%	19.1%	16.4%	8532480 (14.4%)
PB2	19.1%	20.6%	19.1%	16.4%	8532480 (14.4%)

B. Dynamic Designs Under Test

Three dynamic regions were chosen to host the various DUTs used. Each dynamic region used two clock regions in the FPGA to provide clear boundaries during placement and routing. These regions as shown in Figure 3 are named after the Xilinx term pblock, PB0 (bottom left), PB1 (bottom right), and PB2 (top right). These pblocks each use about 20% of the FPGAs resources, see Table I, to allow for the implementation of the unmitigated and TMR verions of each DUT.

1) MicroBlaze: The MicroBlaze is a 64-bit capable RISC, not RISC-V, soft processor optimized by Xilinx for its products [30]. Xilinx reports a maximum frequency (Fmax) of 212 MHz and a Dhrystone million of instructions per second (DMIPs)/MHz of 1.04 for the default configuration implemented on the targeted Artix FPGA. The MicroBlaze processor includes optional features for floating point, cache, and a memory management unit. The default configuration was used for comparison to the following open source solutions used in this experimental design.

2) PicoRV32: The PicoRV32 is a small RISC-V processor that can achieve a high Fmax when implemented in FPGAs [15]. This processor can achieve a Fmax of 454 MHz on Xilinx Series 7 devices with a DMIPs/MHz of 0.516. This processor has been formally verified by Symbiotic EDA's RISC-V formal verification framework [39]. The regular configuration was used for testing with the addition of an AXI4Lite interface.

3) Kronos: Kronos is a 3-stage in-order RISC-V processor optimized for FPGA implementation [16]. This processor's performance is reported at achieving a DMIPS/MHz of 0.7105. The dual Wishbone buses were adapted to perform AXI4Lite transactions.

4) Taiga: Taiga, a 32-bit RISC-V processor, was chosen for its optimized performance for Intel and Xilinx SRAM-based FPGAs [40]. The pipelined processor implements multiple independent execution units, allowing for variable execution latencies. The Taiga has shown have better performance when implemented in FPGAs when compared to other open source solutions [8]. The Taiga examples include an AXI4 bus for cached transactions and an AXI4Lite bus for peripheral devices.

5) VexRiscv: The VexRiscv is a pipelined 32-bit processor developed with the high-level language SpinalHDL [10]. This processor was chosen for its support of a Buildroot Linux image and performance of 1.21 DMIPS/MHz and 2.27 Coremark/MHz. The processor also takes advantage of a large ecosystem of open source IP for the quick integration of SoCs within FPGA digital designs [41]. The VexRiscv was configured with AXI4 buses in this experimental design.

C. Utilization and Performance

Each processor used local memory within the DUT to execute identical Dhrystone test software and an AXI4 bus to access the DDR memory. The configurations of the processors used within the DUTs did not optimize utilization or performance, but were required to conform to the expected interfaces of the static ATE. The utilizations and performances listed in this paper are specific to this implementation and do not reflect what each processor is ultimately capable of.

Table II: Processor DUT Utilization

Design	LUT	LUTRAM	FF	BRAM
MicroBlaze	2122(1.6%)	747(1.6%)	2019(0.8%)	8(2.2%)
TMR	8619(6.4%)	2241 (4.9%)	6057(2.3%)	24(6.6%)
Cost Ratio	4.05×	3×	3×	3×
PicoRV32	985(0.7%)	49(0.1%)	586(0.2%)	16(4.4%)
TMR	4089(3.0%)	147 (0.3%)	1758(0.7%)	48(13%)
Cost Ratio	4.15×	3×	3×	3×
Kronos	1612(1.2%)	49(0.1%)	913(0.3%)	8(2.2%)
TMR	6663(5.0%)	147 (0.3%)	2739(1.0%)	24(6.6%)
Cost Ratio	4.13×	3×	3×	3×
Taiga	2896(2.2%)	482(1.0%)	1626(0.6%)	16(4.4%)
TMR	11532(8.6%)	1446 (3.1%)	4878(1.8%)	48(13%)
Cost Ratio	3.98×	3×	3×	3×
VexRiscv	3095(2.3%)	54(0.1%)	2789(1.0%)	13(3.6%)
TMR	12363(9.2%)	162 (0.4%)	8367(3.1%)	39(11%)
Cost Ratio	4×	3×	3×	3×

The SpyDrNet tools automated the process of triplicating the design and inserting triplicated voters [7]. It performs fine-grained TMR on the FPGA primitives 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.

Table II lists the utilization of the unmitigated and TMR versions of each DUT. The TMR tools were used to triplicate the netlists, and insert triplicated voters and feedback loops. This resulted in an increase of 3× for the LUTRAM, FF, and BRAM resources with an average of 4× increase of LUT resources, accounting for the additional triplicated voters.

Table III: Processors Results

Processor	Execution (cycles)	Performance per LUT (Normalized)
MicroBlaze	898116	0.99×
PicoRV32	2423212	0.80×
Kronos	1182181	1.0×
Taiga	790199	0.83×
VexRISC	1060932	0.58×

Each processor ran identical test software consisting of 3000 iterations of a Dhrystone benchmark. The execution cycle counts achieved with a 100 MHz clock are listed in Table III. The Fmax is not reported in this paper due to the constraints enforced by the locked routing of the static ATE region. To measure the utilization efficiency of the processor implementations, the performance per LUT was also computed and normalized against the max value as shown in equation 1 where c = number of execution cycles and n = number of LUTs.

$$\textit{Performance per LUT} = \frac{1}{c \times n} \tag{1}$$

The Taiga attained the shortest execution cycle count, while the Kronos and MicroBlaze achieved the best performance per LUT.

IV. Fault Injection Campaign

Random CRAM fault injection was performed on each DUT to collect the necessary data in order to calculate the expected percentage of sensitive CRAM bits that result in a DUT failure. This fault injection process consists of four stages as shown in Figure 5. First, partial reconfiguration is used to bring the DUT to a known working state. The DUT performs the test software, the Dhyrstone benchmark, to verify the known state. Though Dhrystone is used to measure DMIPs (a common processor performance metric), the older benchmark does not fully verify the functionality of the processor. Next, a random fault is injected into the partial region using the region's location to determine the appropriate CRAM frames. The DUT performs the test software again to test if the injected fault resulted in a failure.

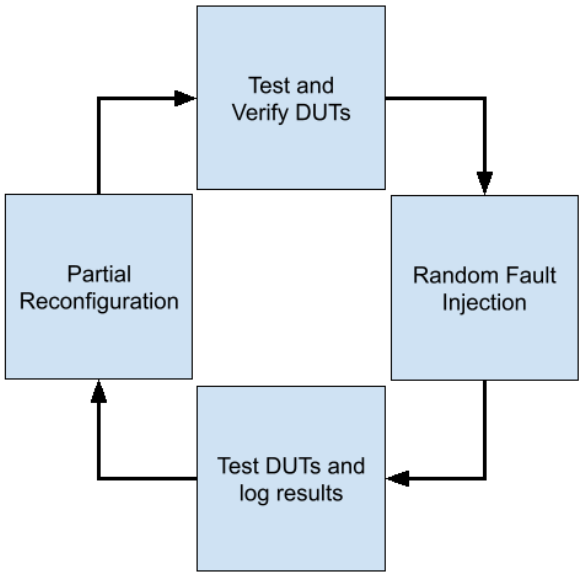

Figure 5: Flow of Fault Injection

This process is repeated until the end of the test or until the ATE is unresponsive. If the ATE is unresponsive, a warm boot is initiated over PCIe or if necessary, the host system is power cycled.

A. Results

Though, theoretically, TMR eliminates all single point failures, due to the FPGA architecture, single CRAM upsets can cause multiple TMR domains to fail and thus a system failure. The results for these single-point failures were collected from the fault injection logs from each FPGA device, see Table IV. Only unique fault locations based on CRAM frame, word, and bit values, were included in the final results. If any injection recorded both passing and failing results, the location was considered a sensitive CRAM bit and counted as a failure. The sensitivity r was computed as the ratio of total injections to the failures observed as shown in equation 2 where k equals the number of failures and n equals the number of faults injected.

$$r = k/n. \tag{2}$$

The standard deviation of the CRAM sensitivity is calculated with equation 3.

$$\sigma = \sqrt{\frac{k}{n^2}\left(1 - \frac{k}{n}\right)} \tag{3}$$

The coefficient of variance of the TMR results demonstrates the variance in the available data in relation to the population mean, as shown in equation 4.

$$\textit{Coefficient of variance} = \frac{\sigma}{r} \tag{4}$$

Table IV: Fault Injection Results

Design	Unmitigated Injections	Unmitigated Failures	Unmitigated CRAM Sensitivity	TMR Injections	TMR Failures	TMR CRAM Sensitivity	Coefficient of Variance	Reduction
MicroBlaze	1088938	8026	0.737%	1926230	187	0.0097%	0.073	76 ×
PB0	363359	2363	0.650%	643409	68	0.011%	0.121	59 ×
PB1	362806	2846	0.784%	641448	103	0.016%	0.152	49 ×
PB2	362773	2817	0.777%	641373	16	0.0025%	0.250	311 ×
PicoRV32	940060	5275	0.561%	2021337	133	0.0066%	0.087	85 ×
PB0	313553	1572	0.501%	675427	39	0.0058%	0.160	86 ×
PB1	313202	1853	0.591%	672836	73	0.011%	0.117	54 ×
PB2	313305	1850	0.590%	673074	21	0.0031%	0.218	190 ×
Kronos	944840	6298	0.667%	1902011	388	0.020%	0.051	33 ×
PB0	315308	1863	0.591%	635224	70	0.011%	0.120	54 ×
PB1	314720	2062	0.655%	633331	76	0.012%	0.114	55 ×
PB2	314812	2373	0.754%	633456	242	0.038%	0.064	20 ×
Taiga	1050770	9576	0.911%	2636047	285	0.011%	0.059	83 ×
PB0	350769	2984	0.851%	881062	95	0.011%	0.103	77 ×
PB1	350080	3270	0.934%	877374	173	0.020%	0.076	47 ×
PB2	349921	3322	0.949%	877611	17	0.0019%	0.243	500 ×
VexRiscv	955561	10064	1.053%	1943079	213	0.011%	0.068	96 ×
PB0	318886	2974	0.933%	648881	66	0.010%	0.123	93 ×
PB1	318348	3591	1.128%	647132	126	0.019%	0.089	59 ×
PB2	318327	3499	1.100%	647066	21	0.0032%	0.218	344 ×

This metric is used to quantify the confidence in a data set [42]. As the CRAM sensitivity decreases, more injections are required to meet the same level of confidence. The reduction in CRAM sensitivity shows the improvement achieved by the TMR mitigation.

In this testing, the reduction in CRAM sensitivity was greater than expected as seen from previous experiments ranging from 10× [6] to 32× [4]. Excluding the Kronos results, TMR processor designs achieved an improvement of 80× or greater. The results for each pblock are included in Table IV. For the unmitigated designs, each pblock achieved similar CRAM sensitivity, but for the TMR designs, the pblocks achieved different results. Excluding the Kronos results, PB1 produced the lowest results while PB2 outperformed the other pblocks. The Kronos TMR results did not fit the pattern of the other processors. While achieving expected TMR results, the TMR Kronos PB2 results greatly under performed compared to the other TMR DUTs. The following subsection will provide an analysis of the results.

B. Analysis

This fault injection experiment differs in three ways from previous experiments. First, dynamic partial regions were used to implement the DUTs and thus constrained the placement and routing (PAR) to fit within the chosen locations and selected interface pins to the static region. Second, a newer Python-based TMR netlist tool was used to generate the TMR designs for testing. Third, partial reconfiguration was used between every fault injection, thus resetting all flip-flop and BRAM values.

The CRAM sensitivity observed across the unmitigated designs scales with their utilization seen in Table II. This expected due to the greater cross-section generated with the use of more FPGA resources. There was no dramatic change in CRAM sensitivity caused by a difference in architecture implementation.

These results show a greater reduction in CRAM sensitivity achieved by the TMR than previous experiments [4], [6]. The new TMR tools and PAR constraints could account for this greater improvement. This could also be caused by the partial reconfiguration fixing any flip-flop and BRAM bits not normally fixed by CRAM scrubbing or the TMR feedback loops.

While the different pblocks using the unmitigated designs attained similar results, the pblocks for the TMR designs achieved very different results from each other. This suggests how greatly PAR can affect the usefulness of TMR mitigation. The PAR for the TMR Kronos PB2 design may have been limited due to some constraint enforced by the static ATE region. Future work will investigate these different possible reasons for the greater improvement of TMR in this experimental design.

V. Conclusion

This novel ATE was successful in utilizing dynamic partial reconfiguration to rapidly deploy and test different DUTs. The FPGA platform was split into one static ATE region and three dynamic regions allocated for DUT implementation. These DUTs consist of unmitigated and TMR versions of five different processors including newly available open source RISC-V processors. While the host system of 20 ATEs was able to perform 15 million fault injections over 10 days across 10 different DUTs, this only covered 5.84% of the possible injections across all the

DUTS. The high-speed ICAP fault injection observed the reduction in CRAM sensitivity achieved by the SpyDrNet TMR tools ranging from 20× to 500× improvements in reliability.

Future work will utilize this powerful platform to further investigate the improvement achieved with mitigation techniques such as TMR. It will explore and compare different PAR strategies, fault injection methods, and recovery methods to build confidence in mitigation techniques chosen for further radiation testing. Additional processors will be added as DUTs with more complex benchmarks for testing and verification to provide sufficient data to aid in the selection of fault tolerant processors for critical applications.

REFERENCES

[1] 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.

[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] 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.

[4] A. E. Wilson and M. Wirthlin, "Neutron radiation testing of fault tolerant RISC-V soft processor on Xilinx SRAM-based FPGAs," in *2019 IEEE Space Computing Conference (SCC)*, July 2019, pp. 25–32.

[5] A. E. Wilson, C. Thurlow, and M. Wirthlin, "Fault injection testing of fault tolerant RISC-V soft processors on Xilinx SRAM-based FPGAs," *JRERE*, March 2021.

[6] 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," *Accepted to IEEE Transactions on Nuclear Science*, 2021.

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

[8] C. Heinz, Y. Lavan, J. Hofmann, and A. Koch, "A catalog and in-hardware evaluation of open-source drop-in compatible RISC-V softcore processors," in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 2019, pp. 1–8.

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

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

[11] ChipsAlliance, "rocket-chip," https://github.com/chipsalliance/rocket-chip, 2020.

[12] A. Ramos, J. A. Maestro, and P. Reviriego, "Characterizing a RISC-V SRAM-based FPGA implementation against Single Event Upsets using fault injection," *Microelectronics Reliability*, vol. 78, pp. 205–211, 2017. [Online]. Available: https://doi.org/10.1016/j.microrel.2017.09.007

[13] U. Jimbo, R. Shioya, and M. Goshima, "Application of timing fault detection to rocket core on FPGA," in *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*, 2018, pp. 178–181.

[14] I. Wali, A. Sánchez-Macián, A. Ramos, and J. A. Maestro, "Analyzing the impact of the operating system on the reliability of a RISC-V FPGA implementation," in *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2020, pp. 1–4.

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

[16] SonalPinto, "Kronos." [Online]. Available: https://github.com/SonalPinto/kronos

[17] 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

[18] 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.

[19] 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

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] L. A. C. Benites and F. L. Kastensmidt, "Automated design flow for applying triple modular redundancy (TMR) in complex digital circuits," in *2018 IEEE 19th Latin-American Test Symposium (LATS)*, 2018, pp. 230–233.

[28] 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.

[29] P. D. I. Torino, P. M. Parvis, and P. S. Bufalino, "Protection and characterization of an open source soft core against radiation effects," no. April, 2018. [Online]. Available: http://cds.cern.ch/record/2313417

[30] Xilinx, "Microblaze soft processor core." [Online]. Available: https://www.xilinx.com/products/design-tools/microblaze.html

[31] C. Thurlow, H. Rowberry, and M. Wirthlin, "Turtle: A low-cost fault injection platform for SRAM-based FPGAs," in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec 2019, pp. 238–245.

[32] F. Ghaffari, F. Sahraoui, M. El Amine Benkhelifa, B. Granado, M. A. Kacou, and O. Romain, "Fast SRAM-FPGA fault injection platform based on dynamic partial reconfiguration," in *2014 26th International Conference on Microelectronics (ICM)*. Doha, Qatar: IEEE, Dec. 2014, pp. 144–147. [Online]. Available: http://ieeexplore.ieee.org/document/7071827/

[33] L. A. Aranda, A. Sánchez-Macián, and J. A. Maestro, "ACME: A Tool to Improve Configuration Memory Fault Injection in SRAM-Based FPGAs," *IEEE Access*, vol. 7, pp. 128 153–128 161, 2019, conference Name: IEEE Access.

[34] F. Serrano, J. A. Clemente, and H. Mecha, "A Methodology to Emulate Single Event Upsets in Flip-Flops Using FPGAs through Partial Reconfiguration and Instrumentation," *IEEE TRANSACTIONS ON NUCLEAR SCIENCE*, vol. 62, no. 4, p. 8, 2015.

[35] B. F. Ruag and M. E. Ruag, "SEU Mitigation Techniques for Advanced Reprogrammable FPGA in Space," p. 128, 2014.

[36] C. Frenkel, J.-D. Legat, and D. Bol, "A Partial Reconfiguration-Based Scheme to Mitigate Multiple-Bit Upsets for FPGAs in Low-Cost Space Applications," p. 7.

[37] SQRL. [Online]. Available: http://www.squirrelsresearch.com/acorn-cle-215-plus/

[38] RHSResearchLLC, "Nitefury-and-litefury." [Online]. Available: https://github.com/RHSResearchLLC/NiteFury-and-LiteFury

[39] SymbioticEDA, "riscv-formal." [Online]. Available: https://github.com/SymbioticEDA/riscv-formal

[40] 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.

[41] "Linux on LiteX VexRiscv," https://github.com/litex-hub/linux-on-litex-vexriscv, 10 2019.

[42] H. Quinn and M. Wirthlin, "Validation techniques for fault emulation of sram-based fpgas," *IEEE Transactions on Nuclear Science*, vol. 62, no. 4, pp. 1487–1500, 2015.