Low-Level Placement and Routing Changes to Increase SRAM FPGA Reliability

Matthew Cannon, Andrew Keller and Michael Wirthlin
Brigham Young University

***Abstract*—Mitigation techniques, such as TMR, are used to reduce the negative effects of radiation on FPGAs deployed in space environments. While these techniques increase the robustness of the device, there is still room for improvement in the range of 100 to 1,000x. These improvements can be realized through the low-level implementation of the placement and routing on the device. This work has implemented a wide variety of techniques to realize these gains, achieving an overall improvement of 45,653x through fault-injection testing and an improvement of 368x in radiation testing.**

I. Introduction

Field Programmable Gate Arrays (FPGA) are computational devices (much like a CPU or GPU) that are being considered for many space-based applications. An FPGA is a device with many configurable resources coupled with a configurable routing network to allow it to take on many applications as shown in Figure 1. It can implement any logic function, provided that it contains a sufficient amount of resources to do so. Due to the large bank of input/output (I/O) ports available and amount of resources, FPGAs provide speed and power benefits for many applications. Coupled with their relatively inexpensive cost in low quantities, FPGAs provide many benefits for potential space applications.

However, space is full of many radioactive particles that can induce single event effects (SEE) in electronic devices. SRAM FPGAs are particularly vulnerable to SEEs in the form of single event upsets (SEU). An SEU represents a change in the state of a memory structure, such as a bit changing from 0 to 1, or vice-versa, 1 to 0. Such changes in the FPGA state will change the underlying circuitry implemented on the device. This can include introducing new circuitry to the device, modifying the existing circuitry or the removal of some circuitry. Before deployment in space, the circuit needs to be tested to determine its sensitivity in the space environment. The sensitivity may be improved through the application of techniques specifically developed to mitigate against SEUs.

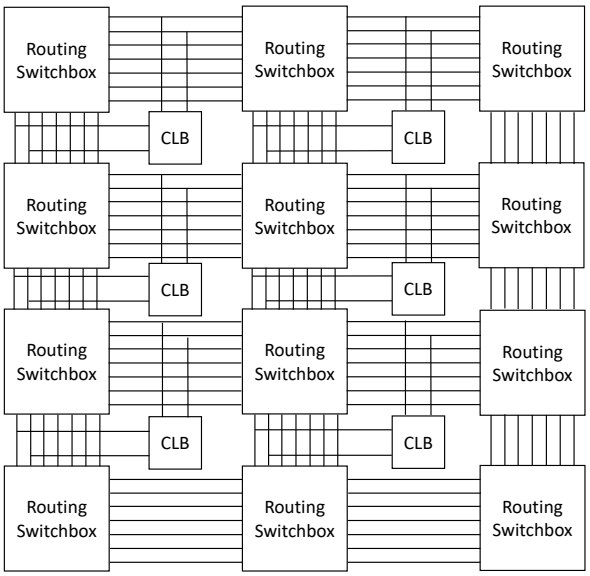

Figure 1: General FPGA Architecture

One of the popular techniques to mitigate against the effects of ionizing radiation is triple modular redundancy (TMR). TMR uses three redundant copies of a module to mask failures. When a module (i.e., the circuit to be protected by TMR) is triplicated, three separate domains are created: TMR_0, TMR_1, and TMR_2, as shown in Figure 2. All three domains are driven by the same input stimulus and under normal operating conditions should yield identical outputs. If one of the domains becomes corrupted, its outputs may not match those of the other two domains. An erroneous output is masked by voting on the outputs from each domain so that only the majority vote is propagated. Voters can be placed throughout a module to synchronize internal signals between domains and increase reliability (often referred to as partitioning). The voting mechanism is often triplicated as well, which prevents the introduction of single-point failures and allows a voter to fail without compromising the integrity of TMR. TMR is able to mask any error that is limited to a single domain between voter insertion points.

Configuration memory repair is often coupled with

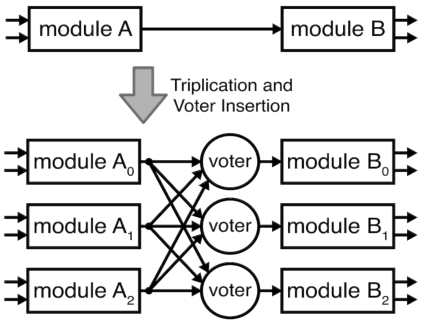

Figure 2: Triple Modular Redundancy

TMR to prevent the accumulation of errors that would break TMR and is often implemented with configuration scrubbing on FPGAs. Configuration scrubbing is usually performed by partially reconfiguring the device with the original bitstream to "scrub" incorrect values. Repair is also needed for the state of the circuit. If the design state becomes corrupted (e.g., counters, state machines, status registers), there needs to be a method to clear the error. Some errors will naturally flush out of the design (i.e., the state is not used in next state logic), or can be manually flushed out of the design on reset. To allow self synchronization, voters need to be placed along feedback paths throughout the design. Scrubbing can even be implemented on BRAM by reading the ECC and correcting any errors, if present.

The theoretical model of a TMR system [1] is shown in Figure 3. In this model there are 3 states: normal operation (S_0), single failed module (S_1), and TMR defeat or system failure (S_2). In states S_0 and S_1, the system functions correctly. In state S_2, the system has failed and outputs of the system should not be trusted. The arc from S_0 to S_1 represents the failure of a single module when all three modules are functioning correctly, which occurs three times faster than the single module failure rate, λ. The arc from S_1 to S_0 represents a repair mechanism (scrubbing) with repair rate μ.

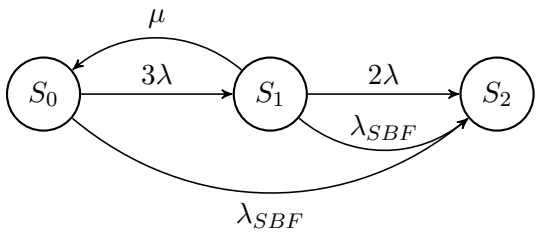

Figure 3: TMR Reliability Model

From the TMR model, the MTTF of a TMR system with repair can be extracted. The MTTF simplifies to:

$$MTTF_{TMR} = \frac{5}{6\lambda} + \frac{\mu}{6\lambda^2}$$

As $\frac{\mu}{\lambda} \to \infty$, $MTTF_{TMR}$ also will go to ∞. This means as $\mu >> \lambda$, the MTTF should be very high . In other words, if a fault affects only one TMR domain and it is repaired before a different domain fails, then there is no limit to the reliability improvement provided by TMR, when considering only single bit upsets. While infinite repair rate is impossible to achieve, high repair to failure rates are possible. It is possible to achieve a repair rate under 1 second for most devices making the repair rate significantly faster than the CRAM upset rate in GEO orbit (typically in the range of .9 upsets per day on the device) [2].

Of particular interest are the two arcs from S_0 and S_1 to S_2. This represents the single bit failure (SBF) rate of the circuit, i.e. the rate at which single SRAM bits cause TMR failure. These bits compromise the functionality of multiple redundant modules simultaneously. Direct system failure can be caused by single point failures (SPF) and common mode failures (CMF). These place a significant ceiling on the overall improvement TMR can provide. Even at an infinite repair rate the maximum improvement is:

$$Improvement_{MAX} = \frac{\lambda}{\lambda_{SBF}}$$

For example, if λ_{SBF} is $\lambda/50$, then TMR can provide, at most, a $50\times$ improvement.

Removing SBF bits can yield significant reliability improvements. TMR combined with configuration scrubbing has already improved the MTTF of a design in space by $50\times$. By removing SBF bits from the design, the MTTF has a potential to improve by $500\times$ or even $5,000\times$.

Testing a circuit's sensitivity is typically done in one of two ways: fault-injection or radiation testing. During fault-injection, errors are randomly introduced into the the devices memory and allowed time to propagate through the system. If the error causes a failure, the injected bit is marked as sensitive, the device is brought back into a known state and the test resumes by selecting a new random bit to inject. This can be easily performed in a lab setting and can be done with the just access to the device.

Radiation testing is done by exposing the device to high energy particles (such as neutrons, protons or heavy ions) while observing the devices behavior. Radiation testing is required to fully understand how the device will behave in its intended environment and to obtain accurate information about the sensitivity of the circuit. However,

this type of testing can be difficult and expensive to perform because it can only be done at a few facilities. Typically, the device will first be tested via fault-injection and only the most promising mitigation strategies will be tested at particle beam. For this work, all mitigation strategies were first tested using fault-injection with the most promising techniques being tested at a particle beam.

II. Automated SBF Removal

SBF bits are broken down into two categories: SPF and CMF. SPF bits are caused by components that are not fully triplicated. Recall in TMR, voters are usually triplicated to avoid single points of failure. If a component (such as an I/O port or a memory) is not triplicated, then a resource failure could also cause a TMR failure. The proper way to address these bits is through complete triplication (i.e. triplicating the component), however, there are other strategies to *reduce* the impact these bits have on the sensitivity (but not remove them completely). CMF bits are different in that they affect multiple domains simultaneously. These can be completely removed through proper mitigation strategies.

Previous work has shown that many of the SBF bits occur in the routing network and can be addressed through placement and routing changes [3]. A circuit goes through several design steps in order to be implemented on an FPGA device. The first is logic synthesis, which converts the hardware description language (HDL) into device specific components, such as lookup tables (LUT) and block memories (BRAM). After synthesis is packing/clustering, which packs the components produced during synthesis into device specific sites or configurable logic blocks (CLB). Once packed, these CLBs are assigned to specific locations on the device. After placement the device can be routed. Finally, after these implementation steps, the bitstream for the circuit is generated which can be used to program the device. This work makes changes to the synthesis, placement and routing of the design to remove the SBF bits.

A. Feedback TMR

There are many ways to apply TMR, however, applying it manually through HDL is error prone and synthesis tools are likely to remove redundancy through optimizations, so automated approaches are preferred. The approach used for this work is to apply TMR to the netlist of a design after logic synthesis. Once the netlist has been modified for TMR, the updated netlist is then used in implementation. Voters are placed along the feedback path to provide auto resynchronization for the flip-flops in the circuit [4]. Doing so also introduces the notion of partitioning in to the circuit. Partitioning refers placing multiple groups of voters in the circuit. Doing so makes TMR more robust as it can withstand multiple errors in multiple domains, as long as those errors occur in different partitions. Figure 4 shows an example of this. Even though errors occurred in all domains, because those errors occurred in multiple partitions, the circuit successfully mitigated against those and did not fail. The tool used in this work is described in [5].

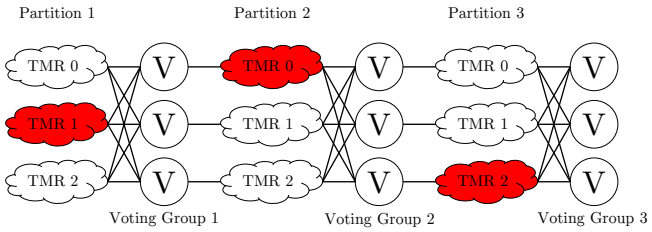

Figure 4: TMR Partitioning

B. SPF Removal

As previously stated, SPF occurs because of untriplicated components in the circuit. This can be common for global signals such as resets and clocks, or could be other untriplicated I/O ports. The effectiveness of TMR will always be limited as long as SPF is present in the design. The best way to mitigate against these is to triplicate them, however, there are still some options to reduce the impact of SPF on the design, the idea being to minimize the footprint of the untriplicated components. Fault-injection testing has shown that the majority of these bits occur along long routes, as the example in Figure 5 shows (red dots showing locations of SPF bits that cause failure). This work has reduced the impact of these bit by inserting 3 identity LUTs into netlist that split the signal and forcing the placement of these LUTs near the source port. Similarly for clocks, 3 BUFGs can be used to "triplicate" the clock for each domain.

C. CMF Removal

Even after triplicating all I/O, there are still single bits that will cause the design to fail, referred to as CMF bits. These bits affect multiple domains and may require more spatial separation to be eliminated. In order to understand the removal techniques a brief overview of the cause for discovered CMF bits (more information can be found in [6]).

The routing configuration bits of the device do not control individual programmable interconnect points

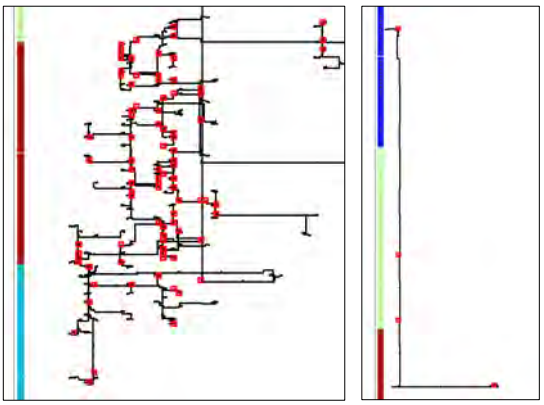

Figure 5: Example of single point failures on a high fanout input pin net (left) and long reduction voter to output pin net (right).

(PIPs), but instead control the rows and columns of a mux (referred to as a routing mux). The programmed row and column bits act as a grid to select one input to propagate to the output wire. The input of the selected column on each row is allowed to drive the row wire, but only the selected row is able to drive the output wire. When a second column bit becomes programmed (through an SEU), a second column in the mux is allowed to drive the row wires (but not the output), possibly creating *multiple* shorts in the mux (up to one short per row wire), as shown in Figure 6.

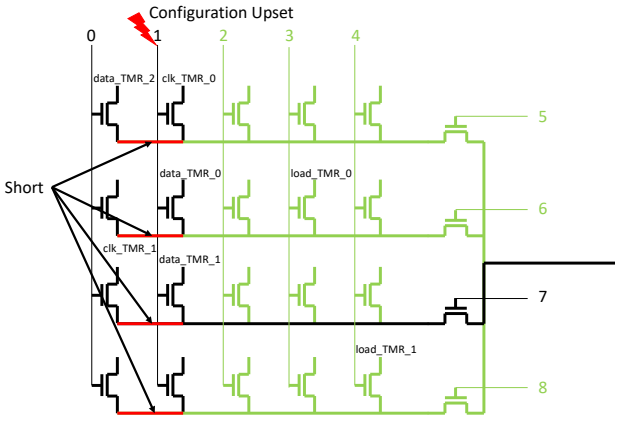

Figure 6: Example of multiple shorts in a routing mux

It is these multiple shorts that cause CMF. Furthermore, through testing TMR failure has only been observed when *multiple* clock nets are shorted in these situations. Three techniques have been developed to address this issue: incremental placement, striping, and incremental routing. Both incremental techniques use the flow shown in Figure 7.

D. Incremental Placement

The goal of this technique is to either limit each tile to a single domain, or ensure that multiple domain tiles do not share flip-flops in the same partition (assuming the TMR design is using advanced partitioning techniques). Because the chosen placement is likely already sub-optimal (heuristics are used for placement), this work assumes that slightly altering the placement should have a negligible impact on its timing. In this technique tiles with CMF are identified and a swap is attempted with a site in one of the tiles neighbors. This will ensure that multiple shorts between clocks will not happen and routing can continue using the vendor's tool. In a sample design, this tool identified many CMF tiles and was able to resolve them as shown in Figure 8.

E. Striping

Another solution is to restrict each tile to only allow cells of one domain to be placed there. While it is difficult in the tools to restrict each individual tile, it is possible to perform this by restricting each column. This can be done by setting a pblock (partial reconfiguration block) for each domain in a column. Thus, the columns of the device are "striped". The tools are forced to comply with these restrictions, thus enforcing the spatial separation to remove CMF from the design. Striping the design is more effective than creating three large pblocks (i.e., one for each domain); however, forcing spacial separation at this level can be detrimental to the achievable clock frequency and can increase routing congestion.

F. Incremental Routing

In this technique the design is placed and routed using the typical design flow, but the design is changed post-routing to remove any CMF that may be present in the design. Because only tiles with multiple blocks can contain CMF, only switchboxes with CMF need to be analyzed. The goal of the incremental routing technique is to prevent shorts between multiple clock signals and other (non-clock) signals.

III. Results

All of the presented techniques were implemented on the b13 design. The b13 design comes from the ITC'99 benchmark suite and is a simple finite state machine that interfaces with a weather station. It has been used by a mitigation working group to test benefits of TMR [7]. This particular design instantiates 256 copies of the b13 to increase resource utilization and statistics collection. The design has been compared against the other

Figure 7: Design flow for removing CMF from a TMR design.

Table I: Radiation Test Results - b13

TMR Type	Fluence	Number of Failures	Cross Section (n/cm^2)	95% Confidence	Improvement
Unmitigated	1.70×10^{11}	314	1.85×10^{-9}	2.06×10^{-9}	1×
TMR (trip I/O)	2.48×10^{11}	6	2.42×10^{-11}	4.39×10^{-11}	76×
Striped	1.90×10^{12}	28	1.47×10^{-11}	2.03×10^{-11}	92×
PCMF	3.98×10^{11}	2	5.03×10^{-12}	1.21×10^{-11}	368×

Table II: Fault Injection Results - b13

TMR Type	Number of Injections	Number of Faults	Percent Sensitivity	95% Confidence Intervals	Number of Sensitive Bits	Improvement
Unmitigated	2,193,073	29,436	1.342%	1.327 – 1.357%	784,860 to 802,876	1×
TMR (w/ SPF)	2,573,824	4,064	0.158%	0.153 – 0.163%	90,520 to 96,258	9×
TMR (trip. clock)	2,595,200	2,480	0.096%	0.092 – 0.099%	54,297 to 58,744	14×
TMR-SPF	2,563,200	57	2.2×10^{-3}%	1.6 – 2.8×10^{-3}%	974 to 1,657	604×
PCMF-SPF	2,081,280	34	1.6×10^{-3}%	1.1 – 2.2×10^{-3}%	641 to 1,291	822×
TMR (trip I/O)	2,351,568	43	1.8×10^{-3}%	1.3 – 2.4×10^{-3}%	758 to 1,405	734×
RCMF	2,400,791	39	1.6×10^{-3}%	1.1 – 2.1×10^{-3}%	659 to 1,262	826×
PCMF	2,396,265	3	1.3×10^{-4}%	0 – 2.7×10^{-4}%	0 to 158	10,721×
Striped	3,401,285	0	2.9×10^{-5}%	0 – 8.7×10^{-5}%	0 to 51	45,653×

Note: 1 error is assumed when no faults were detected.

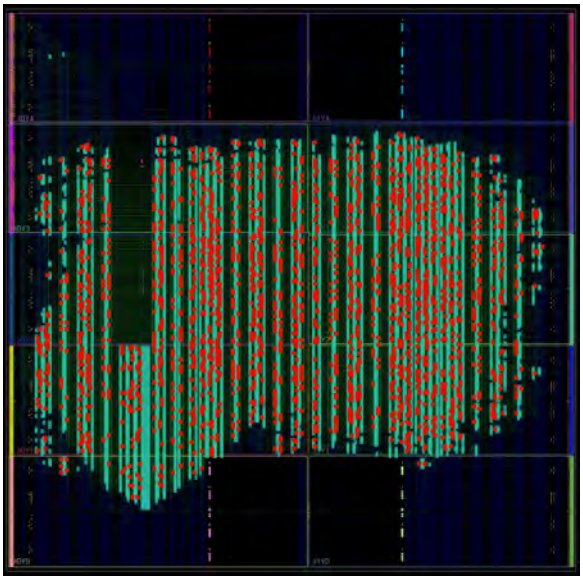

Figure 8: TMR B13 design where identified tiles with CMF are highlighted in red.

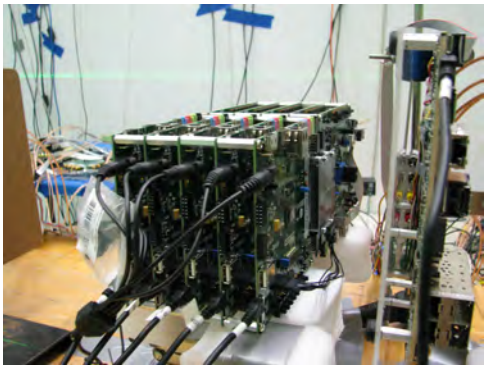

Figure 9: Boards in the neutron beam at LANSCE.

presented techniques for comparison in improvement using both radiation testing and fault-injection, shown in Tables I and II, respectively. Confidence intervals are shown with 95% confidence.

The fault-injection infrastructure for this work consisted of a custom setup using Nexys Video Artix-7 FPGA boards available from Digilent. Each setup consists of 2 boards, one master and one device under test (DUT), connected via the FMC card slot. The master operates with a golden copy of the design (i.e. no CRAM

fault injections) in lockstep with the DUT running the same design, but subject to CRAM upsets. After fault-injection, the design was allowed to run for a period of time to flush out any faults in the system, before being scrubbed and repeating the process. After finding a fault the device was reconfigured and the bit was injected again to verify the upset. Fault-injection was preformed via the JTAG interface.

Due to the reduction in cross-section of these new techniques, many copies of the design are run concurrently in our setup to collect meaningful statistics, as shown in Figure 9. This setup is used for both radiation testing (see Table I) and fault injection testing (see Table II). Because 5 copies of the circuit are run concurrently, this allows data to be collected $5\times$ faster, or in radiation testing, provides $\approx 5\times$ effective fluence.

Many variations of discussed techniques were created to observe their impact on bit sensitivity. Specifically, the following designs were tested: (1) Unmitigated, (2) TMR with SPF (i.e. no triplication on the I/O), (3) TMR with SPF, but clocks triplicated using 3 BUFGs, (4) TMR-SPF (i.e. no triplication on I/O, but SPF mitigation techniques used), (5) PCMF-SPF (i.e. PCMF technique used on TMR-SPF design), (6) TMR with triplicated I/O, (7) Routing CMF technique (8) PCMF technique and (9) Striped technique. The results are shown in Table II. Both the PCMF technique and the striped technique show significant improvement to TMR with complete triplication. This is due to the successful mitigation of single SRAM bits that cause TMR failure.

Due to limited beam time and resources, only a handful of designs were tested using radiation, specifically: unmitigated, TMR with triplicated I/O, striped and PCMF (however other TMR strategies have been tested in [8]). The main purpose of the beam test was to observe the improvement CMF removal provides to a TMR scheme that has implemented all of the other strategies presented. The results of the test are shown in Table I.

The striped design only showed marginal improvement over TMR ($1.6\times$) while the PCMF design showed significant improvement ($4.8\times$) over TMR. As a note, the Striped design was tested at a $2\times$ higher flux rate than the PCMF design which could account for some of the failures. All failures on both designs can be attributed to multi-cell upsets (MCUs) and upset accumulation (multiple single upsets before repair). The differences in improvement between fault-injection and radiation testing are due to radiation testing triggering other SEEs that can not be tested during fault-injection and is expected. However, the techniques that perform well during fault-injection also perform well during radiation testing.

IV. Conclusion

FPGAs are computational devices that can be used in space, but proper mitigation techniques must be applied to assure proper functionality. A TMR tool has been previously developed to help mitigate against SEUs in space and has yielded good results. However, there are many single SRAM bits that still cause failure that could be addressed through more advanced techniques.

An automated tool was developed for this work to identify and remove these single SRAM bits that cause failure. Several techniques were developed to address these bits with varying success. Results from this initial experiment suggest that that the most promising technique can improve the MTTF by $5\times$ over traditional TMR with the vast majority a failures are comprised of multiple upsets (instead of single upsets). Future work will investigate the effectiveness of this technique on a wide variety of designs as well as testing those designs in other radiation sources (such as heavy ions).

This technique and additional techniques that will be developed are improving the reliability of SRAM-based FPGAs in the presence of ionizing radiation. These will allow SRAM-based FPGAs to be increasingly considered for use in spacecraft and other environments with high levels of radiation.

References

[1] S. McConnel and D. P. Siewiorek, "Evaluation criteria," in *Reliable Computer Systems: Design and Evaluation*, 3rd ed., 1998, ch. 5, pp. 334–336.

[2] D. S. Lee *et al.*, "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)*, July 2014, pp. 10–14.

[3] L. Sterpone and M. Violante, "A new reliability-oriented place and route algorithm for SRAM-based FPGAs," *IEEE Transactions on Computers*, vol. 55, no. 6, pp. 732–744, June 2006.

[4] 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.

[5] B. Pratt *et al.*, "Improving FPGA design robustness with partial TMR," in *2006 IEEE International Reliability Physics Symposium Proceedings*, March 2006, pp. 226–232.

[6] M. Cannon *et al.*, "Improving the effectiveness of TMR designs on FPGAs with SEU-aware incremental placement," in *2018 IEEE 26rd Annual International Symposium on Field-Programmable Custom Computing Machines*, April 2018, pp. 1–8.

[7] H. Quinn *et al.*, "Using benchmarks for radiation testing of microprocessors and FPGAs," *IEEE Transactions on Nuclear Science*, vol. 62, no. 6, pp. 2547–2554, Dec 2015.

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