Improving SRAM FPGA Radiation Reliability Through Low-Level TMR Implementation

Matthew Cannon, Andrew Keller and Michael Wirthlin
Brigham Young University

Abstract—**Mitigation techniques, such as TMR with repair, 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 57,443x through fault-injection testing and an improvement of 350x 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 that allows it to take on many different 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.

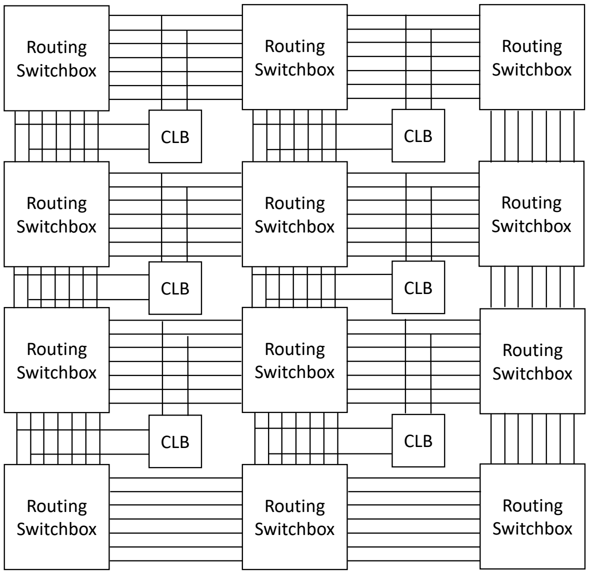

Figure 1: General FPGA Architecture

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

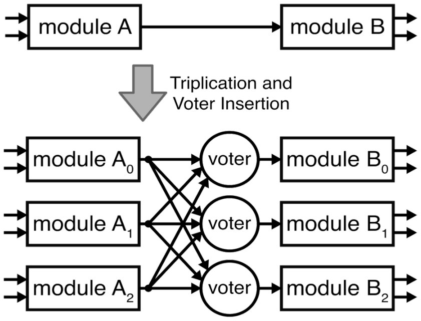

Figure 2: Triple Modular Redundancy

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 [1].

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 and then the most promising techniques were tested at a particle beam.

As the repair rate increases, so should the reliability of the TMR system. During fault injection testing, the repair rate is essentially set to infinity. Mathematical models dictate that the reliability of the TMR system with an infinite repair rate should also be infinite. However, this is not what we observe during testing. There are single bits within the configuration memory, called single bit failures (SBF). The presence of SBF in a circuit limits the effectiveness of TMR, so 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 $5,000\times$.

SBF bits are broken down into two categories: SPF and CMF. SPF bits are caused by components that are not fully triplicated. For example, 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 failures is through complete triplication (i.e. triplicating the component), however, there are other strategies to *reduce* the impact these failures have on the bit sensitivity (but not remove their impact 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 most SBFs occur in the routing network and can be addressed through placement and routing changes [2]. 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 SBF.

II. SPF Mitigation

SPF occurs because of untriplicated components in the circuit. This can be common for global signals such as resets and clocks, or could be 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 3 shows (red dots showing locations of SPF bits that cause failure).

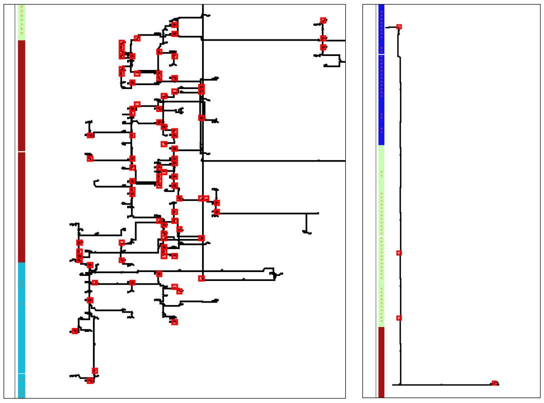

Figure 3: Example of single point failures on a high fanout input pin net (left) and long reduction voter to output pin net (right).

A. Split-Clock

An input I/O pin for a clock is first routed through a buffer, typically a BUFG. In an untriplicated I/O (common-IO) TMR design, there would only be one clock net for each clock in the original design. Each clock can be internally triplicated by being routed to three buffers, one for each domain, and then the output of each buffer is then routed to the cells associated with that particular clock and TMR domain. This mitigation technique is shown in Figure 4.

As shown in the figure, there are a few locations where SPF can occur in the original design, denoted by the red “x”s. The first location is from the input pin to the buffer and the second location is from the buffer to the cells of each domain. While the figure only shows one red “x”, it is likely that each of these nets uses multiple wire segments, which would correspond to multiple configuration bits. One “x” is used for simplicity to mark the different locations SPF can occur. After buffer triplication, there is only one location where SPF can occur, on the net from the input pin to each of the buffers. The possibility for SPF has been removed from the design implementation after the buffers.

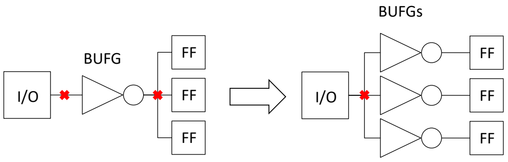

Figure 4: Triplicating Clocks

B. Split-IO

Similar to split-clocks, logic input pins can also be split. This is done by introducing a pseudo-buffer into the netlist. Like a clock buffer, this pseudo-buffer can be inserted for each separate domain to force the net to split, but unlike a clock buffer, this is the pseudo-buffer’s only purpose. This pseudo-buffer is implemented as a pass-through LUT, which is a single input LUT, whose function is to copy the logic value of the input wire onto the output wire. The split-IO technique using pass-through LUTs is shown in Figure 5. These LUTs are constrained to be placed in the CLB tile closet to the input pin.

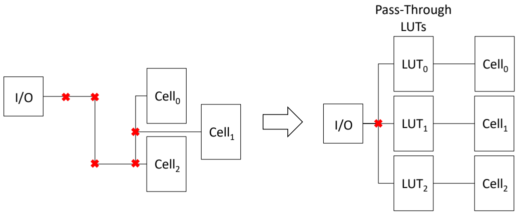

Figure 5: Pass-Through LUTs

C. Early-Split (ES)

Both of these techniques, split-clock and split-IO, can be used together. When used together, they are referred to as early split (ES). Generally, they would be most effective when used together, but certain circumstances may dictate otherwise. These cases could arise in certain environments where parts of the FPGA may be more susceptible to failure than other parts. When only considering single bit upsets (SBU), however, there should be no reason why all of these techniques should not be used.

III. CMF Mitigation

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 [3]).

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

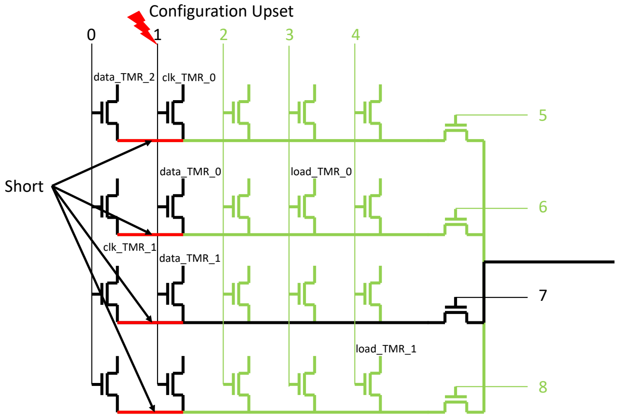

Figure 6: Example of multiple shorts in a routing mux

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. Two techniques have been developed to address this issue: incremental placement and striping.

A. Incremental Placement (PCMF)

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.

B. 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 necessary 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 maximum clock frequency and can increase routing congestion [4].

IV. Results

All of the presented techniques were implemented on the four different circuits, the b13, md5, sha3 and aes128. 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 [5]. This particular design instantiates 256 copies of the b13 to increase resource utilization and statistics collection. The md5, sha3 and aes128 are all hashing algorithms. They are also instantiated multiple times to increase resource utilization.

All of the TMR techniques were implemented on the four circuits to measure their impact on resource utilization. The geometric mean results are presented in Table I. Routing was not able to complete for the striping design on the md5 and sha3 circuits. An important observation from this table is that the number of routing nodes for the striped design is $4.2\times$ that of the unmitigated circuit, $.25$ greater than any other TMR variation. This suggests that striping has a great affect on the circuit's routing.

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 failure the device was reconfigured and the bit was injected again to verify the upset. Fault-injection was preformed via the JTAG interface.

Table I: Mean Implementation Metrics

	Metric/Technique	fmax (MHz)	# nodes	# cells	# sites	# tiles
Mean	**Unmitigated**	1×	1×	1×	1×	1×
	Common-IO (1-Voter)	0.77×	3.48×	3.16×	3.75×	3.62×
	Common-IO (3-Voter)	0.73×	3.66×	3.44×	3.73×	3.56×
	Split-IO	0.80×	3.69×	3.44×	3.67×	3.55×
	Split-clock	0.73×	3.70×	3.44×	3.83×	3.68×
	Split-clock-PCMF	0.73×	3.76×	3.44×	3.83×	3.69×
	ES	0.73×	3.68×	3.44×	3.67×	3.55×
	ES-PCMF	0.73×	3.78×	3.44×	3.76×	3.68×
	Trip-IO (1-Voter)	0.79×	3.62×	3.17×	3.86×	3.74×
	Trip-IO (3-Voter)	0.77×	3.89×	3.44×	3.87×	3.75×
	PCMF	0.76×	3.95×	3.44×	3.87×	3.77×
	Striped	0.78×	4.20×	3.51×	3.46×	3.24×

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 7. This setup is used for both radiation testing and fault injection testing. Because 5 copies of the circuit are run concurrently, this allows data to be collected 5× faster, or in radiation testing, provides ≈ 5× effective fluence.

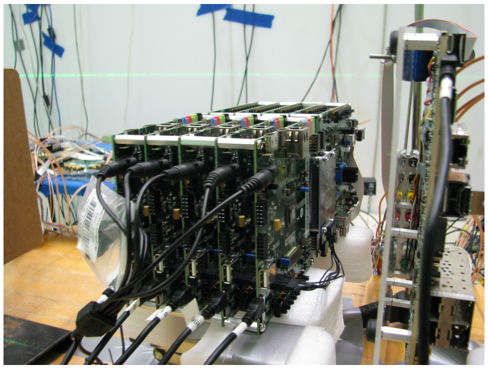

Figure 7: Boards in the neutron beam at LANSCE.

Each circuit was fault injected to measure the bit sensitivity improvement over the unmitigated design. The geometric mean results are shown in Table II. The individual results for each circuit are plotted in Figure 8, with 95% confidence intervals shown. Generally, the split-IO technique performs slightly better than the split-clock technique, although the split-clock-PCMF technique is just as effective as the split-IO technique. The best technique for common-IO TMR was ES-PCMF, which is the split-IO, split-clock and PCMF all applied to the circuit. This yielded a sensitivity improvement of 2,340×. For the B13 and MD5 circuits, ES-PCMF showed a lower bit sensitivity than trip-IO TMR.

Completely triplicating the circuit showed, on average, an order of magnitude improvement over the best common-IO TMR technique. The bit sensitivity for trip-IO TMR is skewed low by the AES128 design which did not show any failed bits during fault injection. This is likely due to the minimal feedback in the circuit which creates few partitions. This allows the tool spatially separate the TMR domains better, leading to fewer CMF bits. Similarly, the SHA3 also has few partitions and only a few sensitive bits were found during fault injection. In contrast, the MD5 design has many partitions and displayed a higher sensitivity for trip-IO TMR.

PCMF and striping showed further improvement upon trip-IO TMR. For striping, no failed bits have been observed during fault injection. Because one failed bit is always assumed when calculating bit sensitivity, the improvement is limited by how amount of injected bits. The PCMF design did not show as much improvement as striping, but it did show a 2× improvement on average. For the B13 and MD5 designs which have more feedback and partitions, PCMF showed more improvement.

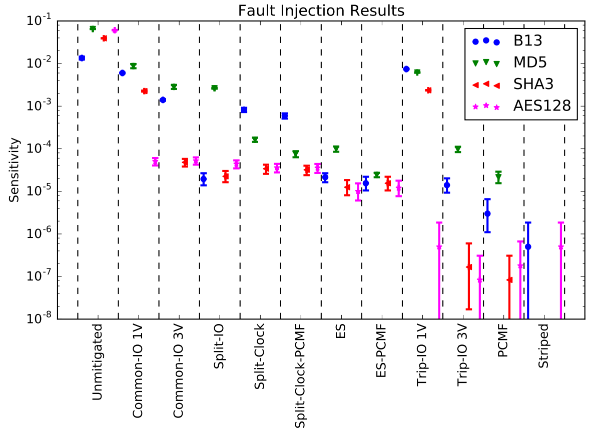

Figure 8: Fault injection sensitivity of designs/circuits

In radiation testing, the striped design only showed marginal improvement over TMR (1.4×) while the

Table II: Fault Injection Mean Results

	Metric/ Technique	Number of Injections	Number of Failures	Sensitivity	Improvement
Mean Results	**Unmitigated**	$6,045,542$	$330,450$	3.80×10^{-2}	$1\times$
	Common-IO (1-Voter)	$6,037,318$	$16,967$	1.56×10^{-3}	$24.3\times$
	Common-IO (3-Voter)	$6,132,724$	$3,367$	3.16×10^{-4}	$121\times$
	Split-IO	$8,000,000$	$5,447$	8.45×10^{-5}	$450\times$
	Split-clock	$6,538,320$	778	1.13×10^{-4}	$336\times$
	Split-clock-PCMF	$6,336,939$	486	8.45×10^{-5}	$450\times$
	ES	$9,255,262$	313	2.27×10^{-5}	$1,675\times$
	ES-PCMF	$18,000,000$	377	1.63×10^{-5}	$2,340\times$
	Trip-IO (1-Voter)	$16,000,000$	$50,836$	4.83×10^{-4}	$78.8\times$
	Trip-IO (3-Voter)	$28,000,000$	225	2.09×10^{-6}	$18,235\times$
	PCMF	$21,541,693$	49	9.92×10^{-7}	$38,341\times$
	Striped	$4,000,000$	0	5.00×10^{-7}	$57,443\times$

Table III: Neutron Radiation Testing Results

TMR Type	Fluence (n/cm^2)	Number of Failures	Cross-Section (cm^2)	+95% Confidence -95% Confidence	FIT Sea-Level	Improvement
Unmitigated	3.19×10^{11}	555	1.74×10^{-9}	1.89×10^{-9} 1.59×10^{-9}	2.26×10^{1}	$1\times$
Common-IO	8.61×10^{10}	51	5.92×10^{-10}	7.58×10^{-10} 4.26×10^{-10}	7.70×10^{0}	$2.9\times$
Split-clock	1.48×10^{11}	39	2.64×10^{-10}	3.60×10^{-10} 1.87×10^{-10}	3.43×10^{0}	$6.6\times$
ES	2.69×10^{11}	19	7.06×10^{-11}	1.10×10^{-10} 4.28×10^{-11}	9.18×10^{-1}	$25\times$
ES-PCMF	4.91×10^{11}	33	6.72×10^{-11}	9.43×10^{-11} 4.62×10^{-11}	8.74×10^{-1}	$26\times$
Trip-IO	1.69×10^{12}	34	2.01×10^{-11}	2.81×10^{-11} 1.39×10^{-11}	2.62×10^{-1}	$86\times$
PCMF	3.98×10^{11}	2	5.03×10^{-12}	1.81×10^{-11} 5.03×10^{-13}	6.53×10^{-2}	$350\times$
Striped	1.90×10^{12}	28	1.47×10^{-11}	2.13×10^{-11} 9.79×10^{-12}	1.92×10^{-1}	$120\times$

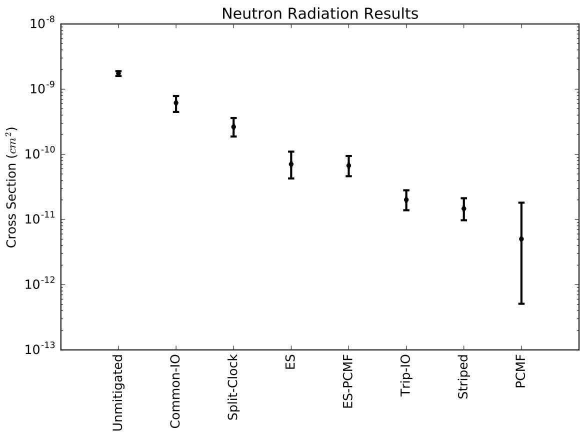

Figure 9: Neutron cross-section of designs

PCMF design showed significant improvement ($4\times$) 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. All of the radiation results are shown in Table III and the cross-sections are plotted in Figure 9.

V. 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 configuration 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 configuration 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 of failures occurring from multiple upsets (instead of single upsets).

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] N. Rollins, M. Fuller, and M. J. Wirthlin, “A comparison of fault-tolerant memories in SRAM-based FPGAs,” in *2010 IEEE Aerospace Conference*, March 2010, pp. 1–12. [Online]. Available: https://doi.org/10.1109/AERO.2010.5446661

[2] 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.

[3] 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.

[4] M. J. Cannon *et al.*, “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.

[5] 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.