Strategies for Removing Common Mode Failures From TMR Designs Deployed on SRAM FPGAs

Matthew J. Cannon, *Student Member, IEEE*, Andrew M. Keller, *Student Member, IEEE*, Hayden C. Rowberry, *Student Member, IEEE*, Corbin A. Thurlow, Andrés Pérez-Celis, and Michael J. Wirthlin, *Senior Member, IEEE*

***Abstract*— Triple modular redundancy (TMR) with repair has proven to be an effective strategy for mitigating the effects of single-event upsets within the configuration memory of static random access memory field-programmable gate arrays. Applying TMR to the design successfully reduces the design's neutron cross section by 80×. The effectiveness of TMR, however, is limited by the presence of single bits in the configuration memory which cause more than one TMR domain to fail simultaneously. We present three strategies to mitigate against these failures and improve the effectiveness of TMR: incremental routing, incremental placement, and striping. These techniques were tested using both fault injection and a wide spectrum neutron beam with the best technique offering a 400× reduction to the design's sensitive neutron cross section. An analysis from the radiation test shows that no single bits caused failure and that multicell upsets were the main cause of failure for these mitigation strategies.**

***Index Terms*— Configuration scrubbing, field programmable gate arrays (FPGAs), single-event effects (SEEs), single-event upset (SEU), triple modular redundancy (TMR).**

I. Introduction

SRAM field-programmable gate arrays (FPGAs) are integrated circuits that can implement any digital logic function. An FPGA consists of lookup tables (LUTs), flip-flops (FF), block memories (BRAM), other special resources (DSPs, Multi-Gigabit Transceivers, etc.), and a large configurable routing network to programmatically connect all of these components. The operations of the LUTs, FFs, BRAMs, etc., are all controlled by static memory cells called the configuration memory (CRAM) [1].

FPGAs are being increasingly used in many harsh environments, such as in space and high-energy physics experiments. While in these environments, FPGAs are exposed to ionizing radiation and subject to numerous types of single-event effects (SEEs), particularly single-event upsets (SEUs). SEUs on FPGAs typically occur in the large CRAM (approaching 1 Gb on the newest devices) potentially changing the circuits implementation [2], [3]. These SEUs could flip an LUT value and change the circuits functionality or occur in the routing network and connect/disconnect physical wires on the device. It is imperative, then, to consider the effects of SEUs in the CRAM before the circuit is deployed within the intended environment.

Triple modular redundancy (TMR) with repair is a commonly employed strategy to mitigate against the effects of SEUs on FPGAs [4]. With TMR, the circuit module is triplicated and voters are inserted on the outputs. The voters then propagate the majority logic value, masking single-module failures. Applying TMR with repair to a circuit has shown to greatly increase the mean time to failure (MTTF) of the circuit by 50–100× [5].

However, the effectiveness of TMR is limited by the presence of single CRAM bits that cause TMR failure. When upset, these bits cause multiple domain failures in the circuit and are referred to as common-mode failures (CMFs) [6], [7]. The presence of these bits in the mitigated design place a limit on the maximum achievable improvement TMR with repair can provide.

The goal of this paper is to identify the underlying architectural causes for these bits that cause CMF as well as introduce mitigation techniques to address their problems. We do this by developing three techniques to mitigate the existence of CMF in TMR FPGA circuits: incremental routing, incremental placement, and striping. The best of these techniques showed an improvement from 80× without these techniques up to 400× with these techniques, a 5× improvement over TMR.

Section II presents a background on TMR with repair and how it is implemented on FPGAs as well as a review of the previous work in this field. Section III introduces a new Markov chain to model CMF. Section IV reviews the main cause of CMF and proposes three strategies to mitigate for CMF. Section V details the experimental setup we used and Section VI shows and analyzes the results. Finally, this paper concludes with an in-depth analysis of the reliability differences between the techniques in Section VII and then offers a final conclusion and future work.

Manuscript received July 13, 2018; revised September 6, 2018 and October 8, 2018; accepted October 17, 2018. Date of publication October 23, 2018; date of current version January 17, 2019. This work was supported in part by the National Science Foundation through the I/UCRC Program under Grant 1 265 957 and in part by the Utah NASA Space Grant Consortium and LANSCE under Grant NS-2017-7574-A and Grant NS-2017-7574-F.

The authors are with the Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT 84602 USA, and also with the NSF Center for Space, High-Performance, and Resilient Computing, Alexandria, VA 22314 USA (e-mail: matthew.cannon@byu.edu; wirthlin@byu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2018.2877579

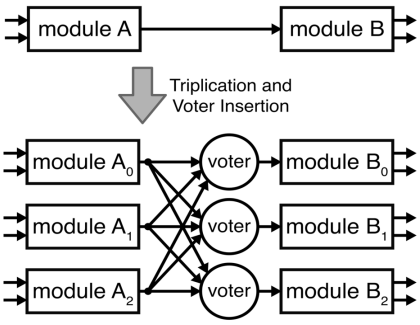

Fig. 1. TMR.

II. Background

TMR can be used to triplicate the circuitry within the FPGA, as shown in Fig. 1, so that if an SEU occurs within the CRAM or FF of one circuit domain, the redundant copies will successfully mask the error. TMR also provides protection for other SEEs such as single-event transients. TMR is usually applied to the circuit using automated tools by directly modifying the design's netlist. The tool we use for this paper is called the BL-TMR tool [8].

Although not intuitive, the MTTF of a TMR system is actually lower than a single system. The MTTF for the single system is $1/\lambda$, where λ is the failure rate of the single system. The MTTF of the TMR system is

$$\text{MTTF} = \frac{5}{6\lambda}$$

or $5/6$ that of the single system. The MTTF of TMR is lower because TMR increases the circuit footprint by $3\times$, adding additional hardware that can fail.

To realize the full benefits of TMR, it is necessary to provide a repair element. Repairing a system refers to bringing the broken module into a correct operating state after it has failed and then resynchronizing it with the other modules. The MTTF of the TMR with repair system is

$$\text{MTTF} = \frac{5}{6\lambda} + \frac{\mu}{6\lambda^2}$$

where μ is the rate of TMR repair. According to this equation, the MTTF will grow with the repair rate. This means as the repair rate increases, TMR with repair will be significantly better than the TMR with no repair and single systems [9].

There are two components to implementing repair for an FPGA circuit. The first step is to restore upset CRAM bits to their original value to "repair" the original circuit functionality. This is done through configuration scrubbing, i.e., continually correcting the original bitstream to its proper value [10]. There are numerous ways to implement configuration scrubbing including blind scrubbing with a golden copy, readback scrubbing with an external scrubbing circuit, or the use of error-correcting codes (ECC).

The second component is to provide repair for the dynamic memory elements in the circuit, such as the FFs and BRAMs, called resynchronization. For memory that is written every cycle (such as FFs), TMR voters can be added along the feedback paths which will automatically resynchronize the memory as soon as the CRAM is scrubbed, called feedback TMR [11].

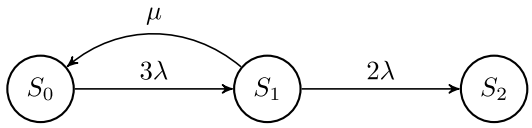

Fig. 2. TMR reliability model.

For memory that is not updated every clock cycle, feedback TMR may not be sufficient. It may be necessary to employ ECC and to refresh the memory every few clock cycles. To refresh the memory, each word is read and ECC is used to correct any errors and the corrected word is stored back in memory [12]. Another option for resynchronization is to assert a global reset when an error is detected.

In spite of adding TMR with repair to static random access memory (SRAM) FPGAs, upsets within some single CRAM bits have been shown to cause design failures [6], [13]. These bits that affect the operation of more than one TMR domain are CMF. These CMF events have been observed in multiple studies [6], [13] and limit the improvement of TMR with repair can provide to 50–100×, but much higher improvements in reliability were expected.

Sterpone and Violante [6] seek to identify both the cause of TMR CMF and to mitigate them using a custom packing, placement, and routing tool. The failure modes that the authors identified for the routing network are shorts between the two wires, double open where two nets are disconnected and a short combined with an open. To resolve these failure modes, the authors propose the reliability-oriented place and route algorithm. This group measured their improvement to be on the order of 350–650× during fault injection. The same group created the VERI-Place tool which also includes an estimator to predict a design sensitivity to SEUs [14]–[16].

One of the main differences between this and other works is that we seek to improve the failure rate of the design when it is subjected to one and only one single-bit upset (SBU). This makes it difficult to compare neutron cross sections with a previous work that used the same circuit, but allowed upsets to accumulate [16]. The authors of [16] presented their results in the form of number of SBUs before failure. In our work, we assume that most deployed systems will have a high repair to failure rate, justifying a mitigation strategy for only one SBU.

III. Reliability Modeling TMR With CMF

The reliability of a TMR system with repair is traditionally modeled using a Markov chain [9], as shown in Fig. 2. Mathematical models are often used to represent the reliability of systems and potential fault-tolerant techniques. Markov chains are useful because reliability metrics such as reliability as a function of time and the MTTF can be derived.

In the Markov chain representing TMR with repair, there are three states: normal operation (S_0), single failed module (S_1), and TMR system failure (S_2), as shown in Fig. 2. In state S_0, the system is functioning correctly and no modules have failed. In state S_1, the system still functions correctly, but there

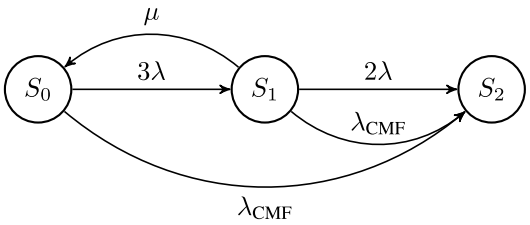

Fig. 3. TMR with CMF reliability model.

is one failed module. The transition from S_0 to S_1 occurs at 3λ or three times the failure rate of a single module. As scrubbing and resynchronization occur, the system can transition back into state S_0, at the repair rate μ, or the CRAM scrub and memory resynchronization rate. If another failure occurs before repair can happen while in state S_1, then the system transitions into the failure state S_2. This occurs two times faster than the single module failure rate. The MTTF equation can be derived from this model as

$$\text{MTTF} = \frac{5\lambda + \mu}{6\lambda^2}.$$

Using the failure rate of a single module ($1/\lambda$), the TMR with repair improvement can be derived as

$$\text{I}_{\text{TMR}} \approx \frac{\mu}{6\lambda}.$$

Assuming a module failure rate of $1/501$ failures/day and a repair rate of 1 repair/second, the TMR improvement should be approximately $120\,000\times$.[1]

However, recent experiments have measured the improvement to be on the order of 10–100$\times$ [6], [17]. Results have shown that there are CRAM bits that when upset will impact the behavior of more than one circuit domain. These bits, called CMF bits, cause CMFs in the design to cancel the mitigation effects of TMR. The discrepancy between the model prediction and measured MTTF can be explained by CMF bits. The model assumes that there must be two independent failures to reach the failure state. However, CMF bits contradict this assumption. They prove that it is possible for one event to cause system failure.

The existence of CMF in TMR designs motivates the need for a new reliability model. A new model is presented in Fig. 3 which introduces the possibility of a failure directly from state S_0. A new arc can be added from S_0 directly to S_2 with a new failure rate λ_{CMF}, which represents the failure rate due to CMF. This arc represents a failure due to a single CMF bit in the design. Using the new Markov chain, a more accurate equation for MTTF can be calculated for TMR with repair

$$\text{MTTF} = \frac{5\lambda + \lambda_{\text{CMF}} + \mu}{6\lambda^2 + 5\lambda\lambda_{\text{CMF}} + \lambda_{\text{CMF}}^2 + \mu\lambda_{\text{CMF}}}.$$

[1]The failure rate of a single module λ is different that the failure rate of a single bit λ_{Bit}. Not all bits will cause a module failure, only some bits will cause a failure. Thus, for an FPGA, $\lambda = \sigma\lambda_{Bit}$, where σ is the percent of bits that cause a single module failure. The failure rate was calculated for an unmitigated LEON3 processor in GEO orbit implemented on a Xilinx 7-Series device (see Table V in [17]). The repair rate was chosen from measured speeds using SelectMap and Joint Test Action Group (JTAG) to perform a full device scrub on a XC7K325T device.

TABLE I
CLASSIFICATION OF FAILURES IN TMR

Error Type	Number	Percentage
Clock Routing CMF	67	92%
Site CMF	6	8%
Total	73	100%

Assuming an infinite repair to failure rate ($\mu/\lambda \to \infty$), to measure the best possible MTTF, yields

$$\text{MTTF} = \frac{1}{\lambda_{\text{CMF}}}$$

or the MTTF of this system is limited by its CMF rate. Using this model, the improvement TMR provides is calculated to be

$$\text{I}_{\text{MAX}} = \frac{\lambda}{\lambda_{\text{CMF}}}.$$

This suggests that at sufficiently high repair rates, increasing the repair rate further has little impact. Rather the improvement is limited by the amount of CMF bits in the design. This equation shows that the improvement is bounded by the portion of the design that is protected. As $\lambda_{\text{CMF}} \to 0$, there are no CMF bits and the MTTF becomes bounded by the repair rate.

IV. CMF MITIGATION

The presence of CMF bits in a design can be observed by fault injecting the device, or introducing upsets into a single CRAM bit and then observing the circuit's behavior. Any bit that causes circuit failure must affect multiple TMR domains and is a CMF bit. Multiple studies have shown the existence of CMF bits in several TMR designs [6], [13].

We conducted a fault-injection campaign to determine the cause of CMF. Through this campaign, we identified two different causes for CMF, clock routing CMF (referred to as routing CMF), and site CMF. As shown in Table I, routing CMF accounts for the majority (92%) of failures, with site CMF accounting for the other 8% of failures.

We have developed three techniques to address this issue: incremental routing, incremental placement, and striping. The incremental techniques are implemented as separate steps to the flow using the open-source CAD tools RapidSmith2 with Tincr [18], [19] as well as RapidWright [20].[2] The striping technique operates as a set of tool command language (TCL) commands into the tool flow, as shown in Fig. 4.

Routing CMF occurs in muxes that are used in the routing network. An example of a routing mux is shown in Fig. 5. When an SEU occurs in one of the configuration bits associated with a routing mux, it can cause multiple shorts in the circuit. For example, in Fig. 5, an SEU occurs in the red configuration bit. This causes a short between the nets data_TMR_2 and clk_TMR_0 and another short between the nets clk_TMR_1 and data_TMR_1. Because this shorting involves clocks from multiple domains, it causes CMF. The

[2]This paper has implemented these techniques on both CAD tools, but no distinction will be made between them for the duration of this paper. The exact implementation details vary slightly due to differences in the application programming interface, but they perform the same from a reliability standpoint.

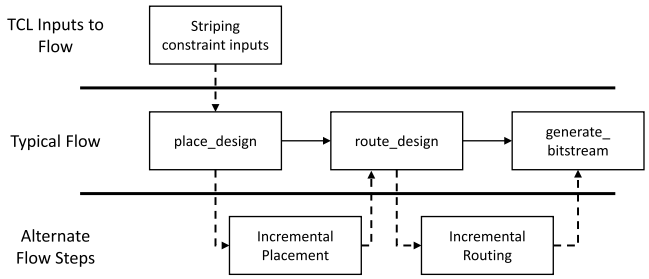

Fig. 4. Flow for implementing CMF mitigation techniques.

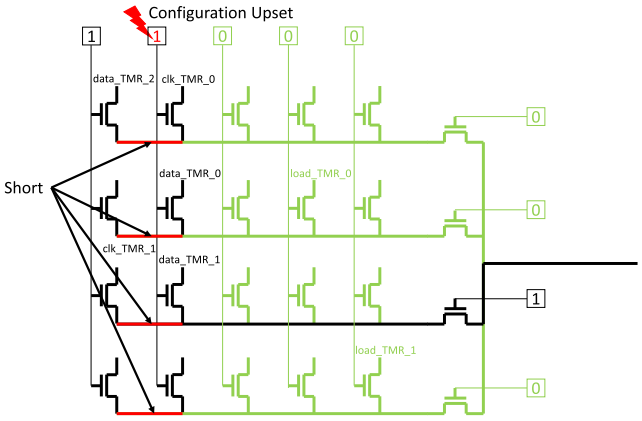

Fig. 5. Example of multiple shorts in a routing mux.

incremental routing and incremental placement techniques focus on addressing the routing CMF issue, as it is the most prevalent and should impact the circuit's reliability the most.

Site CMF can affect the cells in a special logic site called a SLICEM. These sites have other configurable properties that can set the LUTs into different modes. When a site CMF bit is fault injected, we observe many additional upsets upon readback, occurring in the LUTs of the site. Whenever LUT cells from multiple TMR domains are placed on the same SLICEM site, site CMF can occur. The striping technique addresses site CMF as well as routing CMF.

Section IV-A–C details each technique and explains their benefits and limitations.

A. Incremental Routing

As shown in Fig. 4, the incremental routing technique is implemented after the routing step. This technique starts with the routed design and modifies the routing to address the routing CMF issue. The steps for this flow are:

1) identify tiles where routing CMF occurs;
2) unroute all nets associated with these locations (only partially unroute the clock nets);
3) route all these nets with a cost penalty for creating a routing CMF; and
4) iterate using a routing algorithm until all nets are routed (with no resource congestion) and no routing CMF exists [21].

Although this technique does provide fine-grain control over the process, it does have its drawbacks. Particularly, without the vendor's proprietary information associated with the wires on the device, it is difficult to create a router that would have

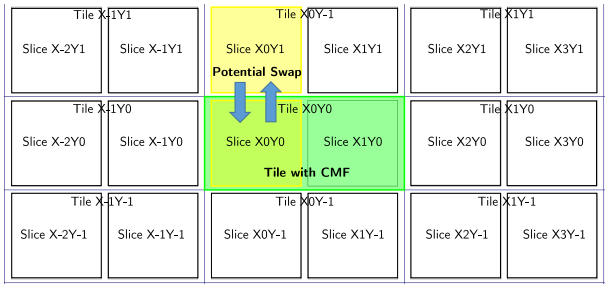

Fig. 6. Incremental placement technique.

the same quality as the one provided by the vendor. This could increase the critical path delay of the design causing it to not meet timing.

B. Incremental Placement

This technique is also implemented as a separate step from the traditional flow, as shown in Fig. 4. It is performed after the placement step, but before the routing step. Because of the design of the FPGA architecture, multiple clock shorting from a single CRAM bit is only possible with clocks in the same tile. Therefore, it is possible to eliminate routing CMF by altering the placement so that a tile cannot contain more than one clock, as shown in Fig. 6. The steps to this flow are:

1) identify tiles with multiple clocks;
2) swap sites within these tiles with neighboring tiles to remove the multiple clocks within the same tile; and
3) ensure that the swap does not introduce multiple clocks to another tile.

One of the benefits for this technique is that most of the optimizations from placement are preserved and that routing is performed using the vendor's tool, thus allowing all of the routing optimizations. Additional details about this technique can be found in [13].

C. Striping

Striping is different from the incremental techniques in that it does not utilize a separate flow or open-source CAD tools, as shown in Fig. 4. Instead, striping introduces a set of constraint commands to the placer and uses TCL commands so that the placer generates a implementation that does not have CMF. The idea behind striping is to restrict every third column in the device to one TMR domain, as shown in Fig. 7. This restriction automatically ensures that every tile has at most one clock. The steps to perform striping are:

1) create a physical block for each column of the device;
2) combine physical blocks such that every third column is included within the same TMR domain; and
3) assign all the cells within a TMR domain to be placed within the associated physical blocks.

The main benefit of this technique is that it can be completely performed within the vendor's tool and no separate flow is needed. However, because of the spatial separation, this could increase routing congestion, potentially yielding an unroutable design, depending on the device utilization.

Striping also differs from other techniques as it mitigates for site CMF. Because site CMF affects multiple cells in the same SLICEM, site CMF can be mitigated by ensuring no two TMR

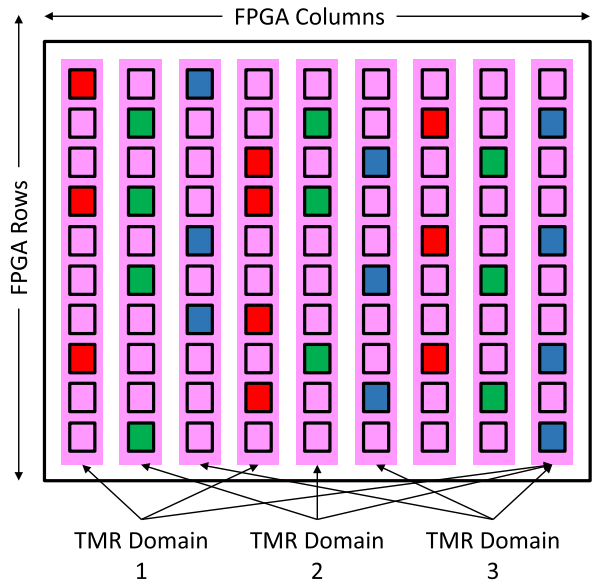

Fig. 7. Striping technique. Each domain is constrained to every third column.

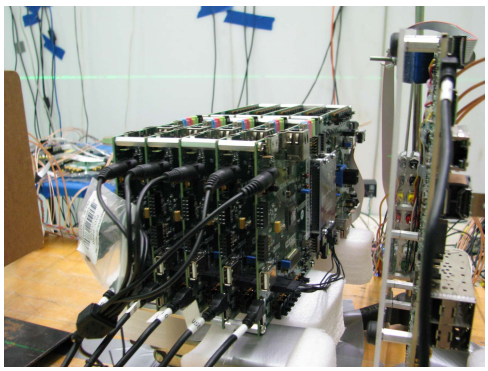

Fig. 8. TURTLE pond in the neutron beam at LANSCE.

domains place cells in the same SLICEM. This condition is met in striping as each column, and by consequence, each slice is restricted to one domain. As shown in Section VI, we have observed no CMF bits when using striping.

V. Experimental Setup

Due to the reduction in cross section and bit sensitivity of these new techniques, it is difficult to observe enough events during radiation testing for statistical significance and to perform significant fault-injection testing. We addressed this problem by developing the testing ultrareliability techniques using low-cost equipment (TURTLE) platform, depicted in Fig. 8. Each TURTLE setup utilizes two Nexys video boards, which contain an Artix-7 XC7A200T FPGA, fabricated at the 28-nm process. One board is designated as the device under test (DUT) and the other as the golden copy. During testing, the DUT is exposed to faults (either through radiation testing or fault injection) and run in lockstep with the golden copy. All of the detection logics is isolated to the golden copy so that it does not contribute to the error rate. Both boards are managed via the JTAG configuration manager (JCM), which has been developed at Brigham Young University [22]. The JCM scrubs and injects faults into the DUT (depending on the type of test) and queries the golden copy to detect errors.

To aid statistics collection, many TURTLEs are stacked on top of each other to form a pond. Typically, in our setup, each pond consists of five TURTLEs. This setup is used for both our radiation testing and fault-injection testing. Because we use five TURTLEs, five instances of the circuit are run concurrently, allowing us to collect data 5× faster in fault injection, or allows us to observe 5× more events during radiation testing.

Our fault-injection approach consists of randomly selecting a bit and flipping it through the JTAG port. After injection, the circuit is tested with numerous input vectors and if an error is observed on the output after a propagation delay, we record that bit as sensitive. The injected bit is then scrubbed and the design is brought back into a known, working state. Fault injection then continues by selecting a new random bit. We randomly choose the bit from the entire device, including sections that might not be used by the circuit. This approach allows us to more accurately measure the bit sensitivity of the circuit and measure improvement margins [16], [23], [24].

Our radiation setup consists of the same setup as fault-injection, five stacked TURTLE boards. During radiation testing, the JCM is used to scrub CRAM upsets, using a readback scrubber. The readback scrubber first reads the bitstream from the DUT and compares that with a stored golden bitstream. Any errors in the DUT bitstream are identified and then corrected. This is continuously done over the JTAG interface and each cycle takes about 2 sec to complete. The board with the golden copy is also scrubbed to correct any SEUs that may occur due to its close physical proximity to the DUT. All SEUs and system outputs are logged for posttest analysis.

VI. Performance and Reliability Results

All the CMF mitigation 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 [16]. Our design instantiates 256 copies of the b13 to increase resource utilization and statistics collection. We choose the b13 design from the benchmark suite to study in depth because its reliability characteristics were well studied in [16]. The unmitigated copy has a neutron cross section of 1.85×10^{-9} cm^2 and bit sensitivity of 1.36×10^{-2} in fault injection. As previously stated, we used the BL-TMR tool to generate feedback TMR designs, and implemented configuration scrubbing using the JCM. Our TMR circuit triplicated everything (including I/O) and had a neutron cross section of 2×10^{-11} cm^2 for an improvement of 80× over no mitigation. We will first analyze each technique's performance and then analyze each technique's reliability characteristics.

A. Timing Analysis

Area effects for TMR are well known (roughly 3-4× more than no mitigation) and these techniques do not affect the area (they only modify the placement and routing, not the netlist). The unmitigated design uses 25 542 cells and 3688 sites, while the TMR design uses 104 066 cells and 19 096 sites. We choose to analyze each technique's maximum achievable

TABLE II
PERFORMANCE ANALYSIS FOR TMR TYPES

Type	fmax (MHz)	Change	Routing Nodes	Change
Unmitigated	82.4	1×	233,780	1×
TMR (trip I/O)	68.2	.83×	1,372,336	5.87×
RCMF	53.0	.64×	1,434,789	6.14×
PCMF	68.6	.83×	1,395,282	5.97×
Striped	69.7	.85×	1,512,096	6.47×

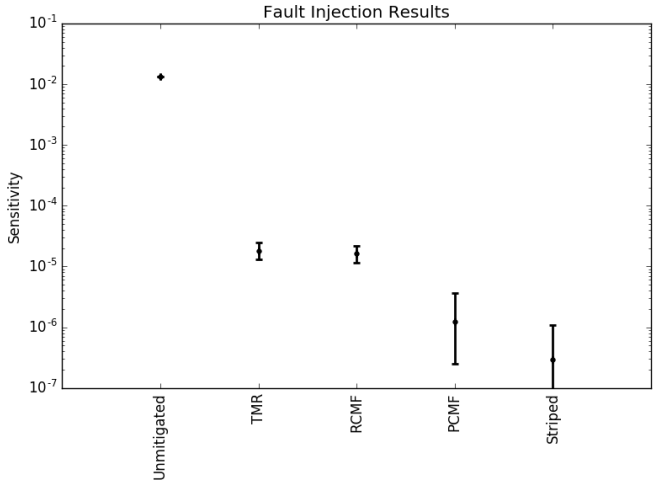

Fig. 9. Fault-injection sensitivity of designs.

clock frequency and the number of routing nodes used as these characteristics will change between each technique. These are reported in Table II.

The timing characteristics follow the expected trends. Applying TMR decreases the maximum achievable clock frequency and applying the incremental placement (PCMF) or striped techniques does not significantly alter that (there is a slight increase). However, the clock frequency is significantly worse for incremental routing (RCMF). This is likely due to the complexity in building a router for large designs and could probably be improved with development.

The number of routing nodes shows a similar trend. Applying TMR significantly increases the number of routing nodes over no mitigation. PCMF shows a slight increase in nodes over TMR (about 20 000), RCMF shows a moderate increase (about 70 000), while the striped design shows a sizable increase (about 140 000). The number of routing nodes is a significant metric because they signify how long the router might take to complete, the amount of routing congestion (routability) and correlate to power consumption (extra capacitance to charge/discharge).

B. Fault Injection Results

All of the TMR variations were tested using fault injection, with the results shown in Fig. 9 and numerical results are shown in Table III. TMR shows an improvement of 730× over no mitigation. RCMF did not prove to be as promising as initial tests suggested, with only an 830× improvement over no mitigation, a slight improvement over TMR.

For the striped design, no single bits have been discovered which caused the design to fail. Tests are still ongoing to determine whether any bits exist that would cause failure, but to do so would require exhaustive fault injection (about 60 000 000 bits). Since we assume one fault when we detect no faults, the improvement is capped by the number of fault injections we have performed. The striped design currently shows a 50 000× bit sensitivity improvement over no mitigation (60× over TMR).

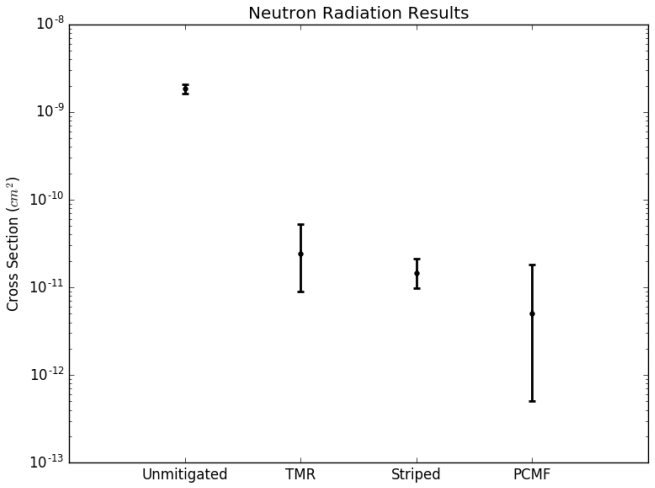

Fig. 10. Neutron cross section of designs.

PCMF also offers a significant improvement over TMR, showing an improvement of 10 000× (10× over TMR) in bit sensitivity over no mitigation. We detected three bits that caused failure for the PCMF design, all site CMF bits. Part of our future work will include creating an additional technique we can use with PCMF to address these bits. Although these results have only been shown on the b13 design, we anticipate that it will be applicable to many other designs.

C. Radiation Testing Results

All of the TMR strategies were tested with a broad-spectrum neutron beam at the Los Alamos Neutron Science Center (LANSCE) [26], excluding the RCMF design due to its poor fault-injection results. The results of the test are presented in Fig. 10 with numerical results provided in Table IV. Each test was performed at normal incident and at room temperature. For these tests, the beam was collimated to 2 in.

The striped design was tested in August 2017 in ICE House II. It showed a 130× improvement over no mitigation and offers a 2× improvement over TMR. Although this is not as much improvement as we anticipated, it still offers two orders of magnitude improvement over no mitigation on a commercial device.

The PCMF was tested in November 2017 in ICE House I. It showed a 400× improvement over no mitigation, which translates to a 5× improvement over TMR. Even with five days of testing across five TURTLE boards, only two errors were observed for the PCMF design during the entire test. Since only two errors were observed for this test, the error range of this estimate is relatively high.

TABLE III
FAULT-INJECTION RESULTS

TMR Type	Number of Injections	Number of Faults	Sensitivity	+95% Confidence -95% Confidence	Improvement
Unmitigated	2,193,073	29,436	1.34×10^{-2}	1.36×10^{-2} 1.33×10^{-2}	1×
TMR	2,351,568	43	1.8×10^{-5}	2.5×10^{-5} 1.3×10^{-5}	730×
RCMF	2,400,791	39	1.6×10^{-5}	2.2×10^{-5} 1.2×10^{-5}	830×
PCMF	2,396,265	3	1×10^{-6}	4×10^{-6} 3×10^{-7}	$10,000\times$
Striped	3,401,285	0	3×10^{-7}	1×10^{-6} 0	$50,000\times$

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

TABLE IV
RADIATION TESTING RESULTS

TMR Type	Fluence (n/cm^2)	Number of Faults	Cross-Section (cm^2)	+95% Confidence -95% Confidence	FIT Sea-Level	Improvement
Unmitigated	1.70×10^{11}	314	1.85×10^{-9}	2.06×10^{-9} 1.64×10^{-9}	2.40×10^{1}	1×
TMR	2.48×10^{11}	6	2×10^{-11}	5×10^{-11} 9×10^{-12}	3×10^{-1}	80×
PCMF	3.98×10^{11}	2	5×10^{-12}	2×10^{-11} 5×10^{-13}	7×10^{-2}	400×
Striped	1.90×10^{12}	28	1.5×10^{-11}	2.1×10^{-11} 9.8×10^{-12}	1.9×10^{-1}	130×

Note: FIT rates calculated using sea-level neutron flux level of 13 n/cm^{2}h according to JESD89A standard [25].

VII. FAILURE ANALYSIS

The difference in neutron cross section between the striped and PCMF designs is surprising, considering that the striped design had a 3× lower bit sensitivity than the PCMF design. Using the log files from the test, it is possible to observe where the SEUs occurred during the scrub cycle that each failure was detected. For every failure for both the striped and PCMF designs, the scrubber logged multiple upset bits.

Using the radiation logs, we can "replay" the radiation test by using fault injection, that is, inject all the upset CRAM bits logged during the scrub cycle of the failure. This can be done to observe whether the logged CRAM bits cause failure outside of the beam. During the radiation replay for the striped design test, we were able to replicate 24 failures.[3] We were able to replicate both errors from the PCMF design test using replay.

The repeatable failures from the radiation replay can be further analyzed for the smallest subset of CRAM upsets that cause failure. For example, the log shows five CRAM upsets for one failure in the striped design. However, only two CRAM upsets are needed to replicate the failure. We determined this by trying all one-bit upsets, then after observing that none of those caused failure, we tried all combinations of two-bit upsets and found a combination that caused a failure.

[3]The other four failures are likely caused by SEEs in other parts of the FPGA that are not logged during the test.

TABLE V
RESOURCES AFFECTED BY MCUS

Resources	Striped # of Bits	Striped Percentage	PCMF # of Bits	PCMF Percentage
Routing	52	96.3%	3	75%
LUT Contents	2	3.7%	1	25%

We performed this technique for all of the repeatable failures for both the striped and PCMF design tests. We then analyzed the specific FPGA resources these bits were associated with and report those in Table V. For the striped design, 96% of these bits were associated with the routing. The PCMF design also shows high sensitivity to the routing, with 75% of these bits being associated with the routing. This leads us to believe that the striped design was more sensitive because it uses more routing resources and has a higher routing congestion.

We noticed another interesting trend during this failure analysis; a high amount of these bits appeared to be multicell upsets (MCUs). For example, the following logical CRAM bits caused failure for the striped design.

- FAR: 0x0040148a, Word: 42, Bit: 31.
- FAR: 0x0040148b, Word: 42, Bit: 30.

These bits only differ by one in the frame address register (FAR) and bit, and are in the same word. Using the technique presented in [27], we confirmed that these were MCUs. This led us to further classify the repeatable failures

TABLE VI
CLASSIFICATION OF FAILURES FROM RADIATION TEST

Failure Type	Striped	PCMF
MCU	18	1
SEU Accumulation	6	1
Not repeatable	4	0
Not Accumulation	22	1

TABLE VII
CROSS SECTION EXCLUDING ACCUMULATION FAILURES

Failure Type	Striped	PCMF
Cross-Section (cm^2)	1.2×10^{-11}	3×10^{-12}
Improvement (over no mitigation)	160×	700×

TABLE VIII
NUMBER OF BIT UPSETS PER SEU

MCU	Striped	PCMF
1-bit (SBU)	71.2%	69.8%
2-bit	23.1%	23.9%
3-bit	2.4%	2.5%
4+-bit	3.3%	3.8%

as MCUs or SEU accumulation (multiple SEUs in the same scrub cycle). As shown in Table VI, out of the 24 repeatable bits for the striped design, 18 were caused by an MCU, with six being caused by SEU accumulation. For the PCMF design, one failure was caused by an MCU and one was caused by SEU accumulation. To ensure that no MCU caused a failure in the classified SEU accumulation failures, each full MCU from the scrub cycle was tested.[4]

When testing in an accelerated environment, the chances of observing multiple events during a scrub cycle are greatly increased. In an unaccelerated environment, it is much less likely to observe multiple events, if the scrub rate is sufficiently high. Using the data from Table VI, we can filter out the failures caused by SEU accumulation to recalculate the neutron cross section for single events. As shown in Table VII, the striped design reduces the cross section over no mitigation by 160× and the PCMF design reduces the cross section by 700×.

Using the MCU data, we can revisit the cross section differences between the striped and PCMF designs. From the data in Table VI, the striped design appears to be more sensitive to MCUs than the PCMF design, or MCUs are more likely to cause failure in the striped design than in the PCMF design. Because the striped and PCMF designs were tested in different locations at LANSCE (ICE House II and ICE House I), the ratio of single-cell upsets (SCU) to MCUs could be different. Using the technique in [27], we computed this ratio for both design tests and present that in Table VIII.

The ratio of SCUs to MCUs for this 28-nm device is comparable to those obtained in heavy-ion testing from [27]. The differences in the SCU to MCU distribution between the striped and PCMF design tests are almost negligible, with the striped design test having 2% less MCUs than the PCMF design test. This is significant as it shows that while the striped design test experienced fewer MCUs per particle, it still demonstrated a higher susceptibility to MCUs.

[4]Each failure classified as SEU accumulation was only caused by bits from separate events, such as bits from multiple MCUs.

VIII. CONCLUSION

Mitigating for single bits that cause multiple domain failures has shown significant improvement for TMR with repair on SRAM FPGAs. Accelerated broad-spectrum neutron testing has also showed that these techniques improve the TMR with repair technique, with the best technique offering a 400× reduction in cross section over no mitigation, a 5× improvement to TMR. We anticipate that these techniques will be beneficial to future FPGA designs in high-upset environments.

We are currently planning future work to develop techniques for other TMR variations. This paper used a TMR variation that included complete triplication of all the I/O. For some designs, this might not be possible to do. TMR variations without I/O triplication might introduce other types of single bit failures into the design that could be addressed by other low-level techniques. Those techniques could then be used in tandem with the CMF mitigation techniques from this paper to provide more options for FPGA circuit designers.

REFERENCES

[1] M. L. Chang, "Device architecture," in *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*, vol. 1, S. Hauck and A. DeHon, Eds. Burlington, MA, USA: Denise E. M. Penrose, 2008, pp. 16–18.
[2] P. Graham *et al.*, "Consequences and categories of SRAM FPGA configuration SEUs," in *Proc. Int. Conf. Mil. Aerosp. Program. Log. Devices*, 2003, pp. 1–10.
[3] M. Ceschia *et al.*, "Identification and classification of single-event upsets in the configuration memory of SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 6, pp. 2088–2094, Dec. 2003.
[4] K. S. Morgan, D. L. McMurtrey, B. H. Pratt, and M. J. Wirthlin, "A comparison of TMR with alternative fault-tolerant design techniques for FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 6, pp. 2065–2072, Dec. 2007.
[5] 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.
[6] L. Sterpone and M. Violante, "A new reliability-oriented place and route algorithm for SRAM-based FPGAs," *IEEE Trans. Comput.*, vol. 55, no. 6, pp. 732–744, Jun. 2006.
[7] H. Quinn, K. Morgan, P. Graham, J. Krone, M. Caffrey, and K. Lundgreen, "Domain crossing errors: Limitations on single device triple-modular redundancy circuits in Xilinx FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 6, pp. 2037–2043, Dec. 2007.
[8] B. Pratt *et al.*, "Improving FPGA design robustness with partial TMR," in *Proc. IEEE Int. Rel. Phys. Symp.*, Mar. 2006, pp. 226–232.
[9] D. Siewiorek and S. McConnel, "Evaluation criteria," in *Reliable Computer Systems: Design and Evaluation*, 3rd ed. New York, NY, USA: CRC Press, 1998, ch. 5, pp. 334–336.
[10] J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb, "FPGA partial reconfiguration via configuration scrubbing," in *Proc. Int. Conf. Field Program. Log. Appl.*, Aug./Sep. 2009, pp. 99–104.
[11] J. M. Johnson and M. Wirthlin, "Voter insertion algorithms for FPGA designs using triple modular redundancy," in *Proc. 18th Annu. ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, New York, NY, USA, 2010, pp. 249–258.
[12] N. Rollins, M. Fuller, and M. J. Wirthlin, "A comparison of fault-tolerant memories in SRAM-based FPGAs," in *Proc. IEEE Aerosp. Conf.*, Mar. 2010, pp. 1972–1983.

[13] M. Cannon, A. Keller, and M. Wirthlin, "Improving the effectiveness of TMR designs on FPGAs with SEU-aware incremental placement," in *Proc. IEEE 26th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2018, pp. 141–148.

[14] L. Sterpone *et al.*, "Experimental validation of a tool for predicting the effects of soft errors in SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 6, pp. 2576–2583, Dec. 2007.

[15] M. Desogus, L. Sterpone, and D. M. Codinachs, "Validation of a tool for estimating the effects of soft-errors on modern SRAM-based FPGAs," in *Proc. IEEE 20th Int. On-Line Test. Symp. (IOLTS)*, Jul. 2014, pp. 111–115.

[16] H. Quinn *et al.*, "Using benchmarks for radiation testing of microprocessors and FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 62, no. 6, pp. 2547–2554, Dec. 2015.

[17] M. J. Wirthlin, A. M. Keller, C. McCloskey, P. Ridd, D. Lee, and J. T. Draper, "SEU mitigation and validation of the LEON3 soft processor using triple modular redundancy for space processing," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, New York, NY, USA, 2016, pp. 205–214.

[18] T. Haroldsen, B. Nelson, and B. Hutchings, "RapidSmith 2: A framework for BEL-level CAD exploration on Xilinx FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, New York, NY, USA, 2015, pp. 66–69.

[19] B. White and B. Nelson, "Tincr—A custom CAD tool framework for Vivado," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs (ReConFig)*, Dec. 2014, pp. 1–6, doi: 10.1109/ReConFig.2014.7032560.

[20] C. Lavin and A. Kaviani, "RapidWright: Enabling custom crafted implementations for FPGAs," in *Proc. IEEE 26th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2018, pp. 133–140.

[21] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," in *Proc. 3rd Int. ACM Symp. Field-Program. Gate Arrays*, Feb. 1995, pp. 111–117.

[22] A. Gruwell, P. Zabriskie, and M. Wirthlin, "High-speed FPGA configuration and testing through JTAG," in *Proc. IEEE AUTOTESTCON*, Sep. 2016, pp. 1–8.

[23] H. Quinn, "Challenges in testing complex systems," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 2, pp. 766–786, Apr. 2014.

[24] J. A. Clark and D. K. Pradhan, "Fault injection: A method for validating computer-system dependability," *Computer*, vol. 28, no. 6, pp. 47–56, Jun. 1995.

[25] *Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Errors in Semiconductor Devices*, JEDEC Standard JESD89A, JEDEC Solid State Technology Association, Oct. 2006.

[26] P. W. Lisowski, C. D. Bowman, G. J. Russell, and S. A. Wender, "The los alamos national laboratory spallation neutron sources," *Nucl. Sci. Eng.*, vol. 106, no. 2, pp. 208–218, 1990.

[27] M. Wirthlin, D. Lee, G. Swift, and H. Quinn, "A method and case study on identifying physically adjacent multiple-cell upsets using 28-nm, interleaved and SECDED-protected arrays," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 6, pp. 3080–3087, Dec. 2014.