Emulating Radiation-Induced Multicell Upset Patterns in SRAM FPGAs With Fault Injection

Andrés Pérez-Celis, Corbin Thurlow, and Michael Wirthlin, *Senior Member, IEEE*

***Abstract*— Radiation-induced multiple-cell upsets (MCUs) are events that account for more than 50% of failures on triple modular redundancy (TMR) designs in SRAM field programmable gate array (FPGA). It is important to understand these events and their impact on FPGA designs to develop improved fault mitigation techniques. This article describes an enhanced fault injection (FI) method for SRAM-based FPGAs that injects MCUs within the configuration memory of an FPGA based on MCU information extracted from previous radiation tests. The improved FI technique uncovers 3× more failures than is observable in conventional single-bit FI approaches. The results from several MCU FI experiments also show that injecting MCUs can replicate the failures observed in the radiation beam test and identify new failure mechanisms.**

***Index Terms*— Fault injection, field programmable gate arrays (FPGA), multiple-cell upsets (MCUs), reliability, single event upsets.**

I. Introduction

ELECTRONIC circuits are susceptible to radiation-induced effects known as single-event effects (SEEs) [1]. These events occur when a high-energy particle strikes and transfers some of its energy to elements of the circuit. This energy is commonly transferred in the form of current. The induced currents from this energy transfer can be sufficient to cause a change in the state of a memory element [2]. This change of state is known as a single-event upset (SEU) and can cause a variety of problems in electronic circuits.

SRAM-based electronic devices are susceptible to SEUs that can change the values stored within SRAM cells leading to data corruption. In an SRAM-based FPGA, the configuration memory (CRAM) is implemented as SRAM memory and contains the device configuration information: routing, interconnections, and logic functions. Since specific CRAM bits implement the logic functions within the FPGA, data corruption in these SRAM cells can lead to FPGA design failure or malfunction.

Manuscript received February 4, 2021; revised March 19, 2021 and March 26, 2021; accepted March 30, 2021. Date of publication April 7, 2021; date of current version August 16, 2021. This work was supported by the Brigham Young University (BYU) site of the National Science Foundation (NSF) Center for Space, High-Performance, and Resilient Computing (SHREC) under Proposal NS-2018-8031-A. The radiation testing was supported by the Los Alamos Neutron Science Center (LANSCE) in December 2018, under Proposal NS-2018-8031-A.

The authors are with the Department of Electrical and Computer Engineering, NSF Center for Space, High-Performance, and Resilient Computing (SHREC), Brigham Young University, Provo, UT 84602 USA (e-mail: pcelis@byu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNS.2021.3071704.

Digital Object Identifier 10.1109/TNS.2021.3071704

Like other SRAM-based devices, ionizing particles can affect more than one memory cell in the CRAM array [3]. An upset in more than one memory cell could happen for two reasons. First, an energized particle may spread the charge over more than one memory cell, causing them to upset. This event is known as a multiple-cell upset (MCU). In a previous neutron test, these MCUs account for 30% of all observed events. Second, an upset may happen in a memory cell that controls the functionality of other elements within the FPGA, causing the other related bits to change their value. This event is known as a micro single-event functional interrupt (micro-SEFI).

Several strategies have been proposed to mitigate SEUs within FPGA configuration memory. A technique widely used on FPGAs is triple modular redundancy (TMR) implemented with configuration scrubbing [4]. While TMR coupled with scrubbing is an effective approach for mitigating against single-event upsets, this approach may not tolerate multiple CRAM upsets caused by a single ionizing particle. The effect of MCUs is important to study because MCUs can overcome the protection from TMR and other techniques such as error detection and correction (EDAC) codes. Researchers have shown that MCUs are a significant contributor to the failures presented in FPGAs protected with TMR. Cannon *et al.* [5] estimated that 50% to 81% of the failures in their TMR designs were attributed to MCUs. With MCUs becoming a major cause of TMR failure, it is important to better understand these events and how they impact FPGA designs using TMR. With this knowledge, improved mitigation techniques can be developed.

Faults can be artificially introduced into an FPGA design to estimate the design sensitivity to SEUs. One commonly used method is fault injection (FI). FI is a technique that emulates SEUs by intentionally changing the content of the FPGA CRAM [6]. FI is usually performed one bit at a time to simulate radiation-induced SEUs.

This article introduces a novel approach for understanding the impact of MCUs on SRAM FPGA designs by injecting multiple cell upsets within the CRAM memory. The approach presented in this article relies on fault injection to inject MCU events into the SRAM FPGA that match representative MCU events seen in previous radiation testing. This enhanced FI technique can be used to evaluate the impact of MCUs on mitigation techniques such as TMR and EDAC and provide a more accurate understanding of FPGA design failure.

This article will begin by reviewing previous efforts to identify MCU events within FPGAs in radiation testing. Next, this article describes general FI approaches used for SRAM FPGAs. The novel MCU FI is described, followed by the results of several experiments conducted using this approach.

II. FI FOR SRAM FPGAS

FI is a process where upsets are introduced artificially into the CRAM of the FPGA. The purpose of these upsets is to test the response of specific designs when they occur. This process can be done using hardware, software, or simulation tools. Although the mechanism for inserting faults into the CRAM is different from radiation testing, FI can provide essential information on the sensitivity of FPGA designs to CRAM upsets and provide early estimates on the effectiveness of SEU mitigation techniques [7]. In many cases, FI is often used to prepare for radiation testing.

Before performing a beam test, it is helpful to perform FI to understand how the design will respond to upsets present in the system. FI may expose undesirable responses to failure. For example, the design may experience failures too often to be a suitable candidate for beam testing. Another example is that an automatic recovery method might not detect specific failures, leading the system to fail to recover as intended. Without performing FI before a radiation test, tests can result in missing data, misspent money, and wasted beam time.

Most FPGA FI systems include the following steps: 1) intentionally inject a fault into the CRAM of an FPGA configured with a specific design, 2) operate the FPGA design under the fault condition with test vectors, 3) identify design or system failures, 4) repair the injected fault, and 5) log FPGA design behavior under faulty conditions. This basic approach seeks to understand how faults affect the behavior of FPGA designs when subjected to radiation-induced faults. In most cases, only a single CRAM bit is altered to estimate the sensitivity of a design to single-event upsets.

Although FI can provide insights into the behavior of FPGA designs affected by radiation-induced upsets, it has some important limitations. First, not all internal state elements and CRAM bits in the device can be upset during FI. Many state bits are not user-accessible and thus cannot be artificially upset. As such, FI will not be able to expose a variety of circuit behaviors that may be seen in radiation testing [8].

Second, FI does not insert faults in the same manner as radiation testing. In most approaches, FI involves upsetting a single CRAM bit [9], [10]. This single-bit injection is unlike the behavior seen in radiation testing in which multiple bit upsets are often observed. Because of this limitation, most FI approaches fail to emulate the actual upset behavior seen in radiation testing and radiation environments.

Another challenge with FI is that faults generally do not arrive in the same temporal manner as in a radiation beam test. Faure *et al.* [11] have described an approach to reproduce ground testing results of a microprocessor using FI. In this work, the temporal nature of fault insertion is not a concern as the intent is to allow the fault to propagate throughout the design. By injecting faults before starting the execution of a program, this FI approach allows for maximum propagation of each fault and facilitates reproducibility of the results.

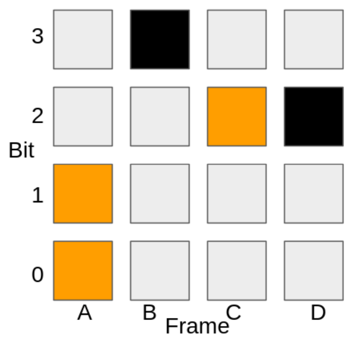

Fig. 1. Example of a 3-bit MCU in the logical view of a CRAM.

Fabero *et al.* [12] extracted MCUs from a neutron beam test. Based on the shapes, they mention that it is possible to perform an FI test that follows the beam distribution to improve the statistics gathered by FI. This work did not use these MCU data for subsequent FI or use these data to demonstrate MCU-specific failure modes.

III. MCU IN FPGAS

Most of the cells that upset in a radiation environment are *single-bit upsets* or **SBU**s. These upsets involve a single particle causing a single cell to upset. These upsets involve only one cell and have no impact on other cells in the device. Other events in the radiation test will have a single particle upset *multiple* cells. These events are called multicell upsets or **MCU**s, and for a previous neutron testing experiment, they account for 30% of the number of upset events. MCU events are caused by the charge of a single particle spreading to more than one memory cell.

Fig. 1 demonstrates an example of an MCU within a 1-D array CRAM within an FPGA. The x-axis represents the frame number[1] and the y-axis represents the bit number within the frame. This figure includes a 3-bit MCU (represented in orange) with upsets in logical locations (A,0), (A,1), and (C,2).

The work in [3] demonstrates the feasibility of extracting MCU events from radiation test data. This approach analyzes the CRAM upset data within discrete "scrub cycles" that contain multiple upsets that are either unrelated or caused by the same ionizing event. After analyzing these data, MCU events are extracted and specified with both their shape of MCUs and logical location.

Table I shows an example of two MCUs extracted using the method mentioned before for a neutron test at Los Alamos Neutron Science Center. The first column *ScrubID* shows an identification number of the readback of the CRAM. The following three columns specify the exact location in CRAM of the upsets comprising the MCU. With enough radiation test data, the cross section and distribution of various MCU sizes and their shapes can be estimated. These parameters can be used to perform FI of MCUs that provides additional information on the failure modes of the design.

The method introduced in [3] to extract MCUs from a list of unique upset bits involves identifying common patterns on

[1]Frame is the name that Xilinx gives to the smallest addressable unit comprised by a group of words in the logical organization of the CRAM

TABLE I
3-BIT MCU AND A 2-BIT MCU EXTRACTED FROM NEUTRON BEAM TEST DATA

ScrubID	Frame	Word	Bit
541	0x00403084	46	17
541	0x00403085	46	16
541	0x00403086	46	17
640	0x00021122	51	24
640	0x00021122	51	25

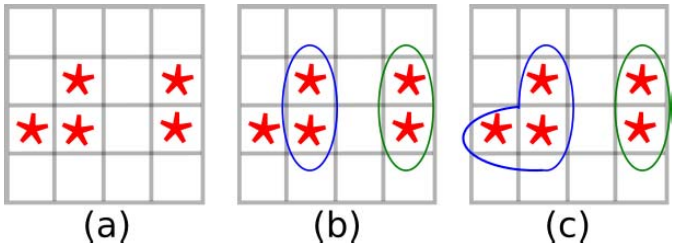

Fig. 2. MCU reconstruction example. (a) Observed upsets in the scrub cycle. (b) Grouping of MCUs after processing the first MCO $(0, -1)$. (c) Updated grouping of MCUs after running through all MCOs.

the positional difference of upsets. These common patterns are then combined and analyzed to identify statistically likely MCUs. This method consists of the following steps:

1) Gather SRAM upset data,
2) Remove contamination from the data,
3) Compute the offsets between the upset locations and generate a histogram of offsets,
4) Select the most common offsets (MCOs) using Poisson statistics, and
5) Reconstruct MCUs based on MCOs.

This MCU extraction technique is based on identifying upset patterns that occur within a scrubbing interval during radiation testing. These patterns are identified by using Poisson statistics to select the pairs of upsets that occur far more frequently than they would if upsets were distributed randomly. The MCOs are selected and used to reconstruct the MCUs.

The reconstruction of MCUs is an iterative process that iterates through each scrub cycle and groups two-bit upset pairs that have the same offset as any of the MCOs. Consider the example in Fig. 2, where a scrub cycle experienced five individual bit upsets. Also, consider that the adjacency model has only two MCOs: $(0, -1)$, $(1, 0)$. Fig. 2(a) shows the data in the scrub cycle. The algorithm takes the first MCO on the adjacency model, $(0, -1)$ for this case, and groups the bits into MCUs. The result is shown in Fig. 2(b). The algorithm goes through all the MCOs of the adjacency model and groups the upsets into MCUs. For this example, the resulting MCUs are shown in Fig. 2(c). Then, the algorithm continues with the next scrub cycle.

IV. MCU FI

Results from the MCU extraction process can be used to generate a database of MCU shapes and their distribution. With this information, it is possible to perform different types of MCU FI tests. One test can follow the distribution of SBU and MCUs as seen in a radiation environment or during a beam test. Another type of test can exercise the design and monitor the response to uncover failure modes that only occur when exposed to MCUs. This article focuses on the latter.

A straightforward way to exercise the response behavior to MCUs of a given design is to pick an MCU shape from the database generated with the extraction of MCUs using the technique in [3]. From this database, it is possible to pick a random MCU and inject it into the design. Since the MCU injected was previously seen in the beam, we can assume that the MCU is a valid MCU that could be observed in radiation environments. For this work, MCUs injected during FI are those that were previously seen in the neutron beam.

Performing FI of MCUs involves additional challenges, given that MCUs are related to the physical layout of the device. First, it might not be possible to inject the desired MCU shape in the desired location. This obstacle could happen due to an invalid logical bit location that restricts the user from accessing the bit. Consider the example in Fig. 1, where the black squares indicate bits that are not addressable to the user. It is not possible to inject the same MCU shape at bits (B,0), (B,1), and (D,2) because bit (D,2) is an invalid bit.

Second, even though the user can attempt to inject any MCU shape in random bits of the device, this could violate the nature of MCUs. To avoid this violation, the injected MCU shape must be a determined product of a single charged particle. Potentially, the selected bits are not physically close to each other. To overcome this challenge, it is necessary to perform statistical analysis on the locations of the MCUs and restrict the valid location to inject an MCU.

Injecting MCUs presents different challenges than single-bit FI. To inject an MCU successfully, the logical locations of bits to be injected must be constrained to a range of user-accessible locations within the device CRAM. Additionally, MCU injection requires that all upsets are injected and propagate through the system before any repair to the upset CRAM bits is performed. This particular aspect of injecting MCUs requires a change to the normal FI flow.

When injecting an MCU, the smallest amount of delay possible between injections is used to inject the bits that compose the MCU. The FI system keeps track of all the injected bits to repair them after injection. Similar to the normal FI flow, the system status is monitored while the injections occur, and system failures are reported. The FI algorithm then needs to repair all injected faults as part of the MCU to avoid the accumulation of upsets in the design.

V. EXPERIMENTAL SETUP

Several experiments were conducted to show the benefits of emulating MCUs during FI tests. The MCU events injected during these experiments were based on radiation test data from previous neutron radiation testing at LANSCE. The MCUs were identified from the CRAM upset data by following the process described in Section IV.

All of the FI experiments were conducted on the TURTLE FI system [6], as shown in Fig. 3. This system uses two Xilinx XC7A200T devices—one acts as the design under test (DUT)

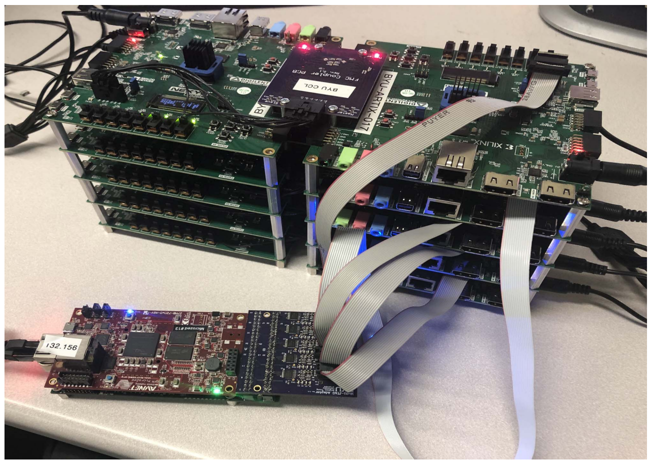

Fig. 3. Multiple TURTLE FI platform featuring two XC7A200T devices per layer.

and the other as the golden device. Faults are injected into the DUT while the golden device carefully monitors the DUT behavior. DUT failures due to CRAM faults are reported to a host system.

The MCU FI was performed using the JTAG Configuration Manager (JCM) [13]. The JCM configures both devices, injects faults into the CRAM of the DUT, compares the output of the golden and DUT designs, reports any failures, and corrects the failure by either scrubbing or reconfiguring the device. For these experiments, a 1 ms delay was allowed between each injected MCU to allow 8 000 test vectors to be evaluated. In this experiment, the delay between each injection in a single MCU was 9.05×10^{-4} s.

For these experiments, the JCM-injected faults based on the MCU patterns extracted from a neutron beam test of the XC7A200T device at LANSCE using the technique on [3]. The collected data from the beam tests are divided into individual scrub cycles. These scrub cycles contain both the logical address location of 1 210 370 upsets and an indication to denote if a failure was detected in the scrub cycle. With these data, two FI experiments were performed on eight different FPGA designs using the TURTLE system.

The FPGA designs used for these experiments were all based on the B13 benchmark circuit (a simple state machine design [14]). The B13 state machine was replicated 256 times within a single FPGA design to utilize a significant number of resources on the device. This base design was then modified with various SEU mitigation techniques to create eight total design variations for testing. These designs are:

Non-TMR	The original circuit with no mitigation techniques applied to it.
Common-IO	The original circuit with TMR applied to it but without any of the input or output pins being triplicated.
Split-clock	The common-io three-voter circuit with a split-clock mitigation technique applied. Unlike the common-io design, the split-clock mitigation technique triplicates internal clock buffers in the DUT design.
Trip-IO	The original circuit with TMR applied to it, and where all the input and output pins are triplicated.
SPF-PCMF	The triplicated-io circuit with single point of failure (SPF) and Place Common Mode Failure (PCMF) applied. The PCMF technique alters the placement of the design to increase the reliability of the design [15].
SPF-TMR	This circuit implements both split-clock and split-io mitigation techniques. Split-io triplicates IO buffers within the design rather than triplicating input pins to the design.
SPF-PCMF	This is the SPF circuit with PCMF added for increased reliability.
Striped-TMR	The trip-io circuit with striping applied. Striping is a technique that constrains each TMR domain to separate columns within the device [5].

The primary goal of all experiments described in this article is to quantify the design failures that are caused MCUs (i.e., not caused by SBUs). When MCU failures are observed during FI, the CRAM cells that make up the MCU are injected one at a time to verify that the fault is caused by the union of CRAM upsets and not due to any one of the CRAM cells by itself.

VI. Experimental Results

Three experiments were performed to better understand MCU-induced design failures. In the first experiment, MCU FI was performed to categorize faults seen in the radiation test as caused by either SBUs or an MCU. In the second experiment, MCUs were injected into the most reliable designs to identify MCU design failures within the designs using the most effective SEU mitigation techniques. In the third experiment, MCU data observed from multiple radiation tests are combined and applied to uncover new failure mechanisms not seen during radiation testing.

A. Experiment 1: Comparison of SBU Failures to MCU Failures

The first experiment was designed to show that some failures will only occur in a design if multiple upsets are present in the system during operation, that is, if an MCU has occurred. For this experiment, each of the eight designs was injected with SBUs. These bits were the upsets recorded during a previous beam test that is part of a scrub cycle with a failure. All upsets were injected as single bits even if the upset was previously determined to be part of an MCU. During FI, all upsets that caused a failure on the design were recorded for further analysis. After injecting all single-bit upsets, each design was injected with the corresponding MCUs of the selected scrub cycles. The first experiment concluded by determining the number of MCUs that caused a failure in each design.

The first experiment shows that there is an additional number of failures that can only occur in the presence of an MCU. The results of this experiment are summarized

TABLE II
SUMMARY OF RESULTS FOR SINGLE-BIT AND MCU FIS FOR EXPERIMENT 1, WITH ONLY SELECTED MCUS INJECTED

	Failures		Additional MCU
	Single-bit	MCU	Failures
non-tmr	893	73	8.17%
common-io	50	11	22.00%
split-clock	29	7	24.14%
trip-io	4	12	300.00%
spf-pcmf	4	5	125.00%
spf-tmr	2	6	300.00%
pcmf-tmr	0	0	NA
striped-tmr	0	0	NA

in Table II. The designs are sorted based on the number of failures shown during the single-bit FI. It is interesting to note that the percentage of additional failures seen during MCU FI passed 100% in three of the eight designs. Not only does this translate into more total failures, but also the additional statistics gathered could tighten the error bounds for the computation of the sensitivity of the design [16] if the injected events were following the distribution of the beam.

Two of the designs, pcmf-tmr and striped-tmr, had no failure for either single-bit FIs and MCU FIs. These two designs also had the least number of failures during beam testing and are the two designs under test for the second experiment.

B. Experiment 2: MCU FI Failure Analysis

The second experiment performed additional MCU FI on the two designs with the lowest number of recorded failures during the beam tests. This additional FI was an exhaustive MCU test where all the scrub cycles, regardless if it contained a failure or not, were considered to extract the MCUs to inject. The goal of injecting MCUs in this experiment was to demonstrate the need to test designs that have added SEU mitigation techniques. This experiment helps understand the behavior and failure modes of these mitigated designs. For this experiment, over 107 000 MCUs corresponding to 284 000 upsets were injected between the striped and PCMF designs. The MCUs injected corresponded to all the MCUs that were detected for each design during beam tests.

Table III shows the results of the second experiment. The first column shows the number of failures that happened on the designs during beam testing. When injecting the extracted MCUs, the number of failures increased for both designs. Interestingly, we were able to replicate and even add one additional failure with the injection of all the MCUs to the striped-tmr design. The results also show that we were exceeded the number of failures for the pcmf-tmr design by 7. Since the MCUs were injected in the same location as they appeared in the beam test data, this experiment shows that even replaying the data can yield tighter bounds on the sensitivity of beam-tested designs.

TABLE III
RESULTS FOR THE SECOND EXPERIMENT

Design	Failures		Increase	Total Upsets
	Beam test	Fault Injection	in Failures	
striped-tmr	37	38	2.7%	450,693
pcmf-tmr	2	9	350%	98,200

TABLE IV
RESULTS FOR THE THIRD EXPERIMENT

Design	Failures		Increase	Total MCU
	Beam test	Additional FI	in Failures	
striped-tmr	37	9	.24×	20,562
pcmf-tmr	2	47	23.5×	87,570

C. Experiment 3: Reusability of MCUs to Test TMR Designs

This experiment injected MCUs from other radiation tests into the pcmf and striped TMR designs. The goal was to demonstrate that MCUs extracted from other radiation tests and MCU FI can be used to gather more statistics on the design failures. As noted before, the two designs chosen for this experiment are the most robust. During all the neutron beam tests, pcmf only had two failures, which yielded a large error bound in the design sensitivity. Using MCU FI can decrease those bounds and provide additional information on the failure modes of the design.

Table IV shows the results of the third experiment. The first column specifies the test design. The second shows the number of failures during the beam test. The third column is the number of additional failures experienced by the design when injecting MCUs from other radiation tests. It is worth noting that this experiment is not trying to replay the data but instead uses other MCU data comprised of valid MCU shapes to make the design fail. The fourth column has the percentage increase of additional failures. Finally, the last column shows the number of injected MCUs.

Results from this experiment show that using data from other beam tests can be beneficial to help uncover failures in robust TMR techniques. This experiment encourages the reusability of data and demonstrates a straightforward, low-cost FI approach to test SEU mitigated designs with valid MCUs. This experiment can be performed as part of the FI routine before going to a beam test or as an alternative when a beam test is not an option.

VII. CONCLUSION AND FUTURE WORK

This article shows an enhanced FI testing method that includes MCUs for their injection. Using MCUs during FI helps uncover possible unwanted behaviors of a design. Additionally, it helps to produce failures on designs protected with robust fault-tolerant techniques. Furthermore, injecting MCUs provides an alternative method to test designs for projects when access to a beam may not be possible.

The results show that MCUs cause additional failures that cannot be replicated with the injection of a single bit. For our designs, this resulted in 300% more failures. For the second experiment, our FI method produced, for one of the designs, more than 3× more failures. This result is interesting because even without moving the location of the extracted MCUs, or without generating MCUs, it is possible to induce more failures in the design than the ones experienced under the beam. This insight shows the importance of at least replaying the beam results. Finally, for both experiments, having additional failures translates into tighter bounds for the computation of the device cross section.

A possible explanation for the difference in results between the neutron and FI tests could be that during beam tests, faults can occur directly after the checking of the device status or during a scrub cycle of the CRAM. The maximum time for a fault to propagate in the beam test is determined by the scrub cycle, which is 0.25 ms. For MCU FI, the time is 1 ms plus 0.1 ms for each bit in the MCU. In contrast, the timing of the injection is controlled in FI. This controlled environment ensures that every injected fault is present in the device for the duration of one entire scrub cycle. This delay propagates the fault throughout the system and allows the failure or other behavior to reach the output before the device status is checked.

For future work, we want to achieve the goal of modeling real-world radiation environments. For this goal, we would adjust the MCU shapes and locations based on the fluence distribution of the radiation environment. Then, we want to explore generating new shapes based on the extracted patterns of the beam test. Finally, we want to combine these data with the constraining of MCU shapes to create an FI technique capable of producing results that more closely mimic ones expected during the exposure to a radiation environment.

REFERENCES

[1] 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.

[2] H. Quinn, P. S. Graham, K. Morgan, J. Krone, M. P. Caffrey, and M. J. Wirthlin, "An introduction to radiation-induced failure modes and related mitigation methods for Xilinx SRAM FPGAs," in *Proc. ERSA*, 2008, pp. 139–145.

[3] A. Perez-Celis and M. J. Wirthlin, "Statistical method to extract radiation-induced multiple-cell upsets in SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 67, no. 1, pp. 50–56, Jan. 2020.

[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] M. J. Cannon, A. M. Keller, H. C. Rowberry, C. A. Thurlow, A. Perez-Celis, and M. J. Wirthlin, "Strategies for removing common mode failures from TMR designs deployed on SRAM FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 66, no. 1, pp. 207–215, Jan. 2019.

[6] C. Thurlow, H. Rowberry, and M. Wirthlin, "TURTLE: A low-cost fault injection platform for SRAM-based FPGAs," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs (ReConFig)*, Dec. 2019, pp. 238–245.

[7] H. M. Quinn, D. A. Black, W. H. Robinson, and S. P. Buchner, "Fault simulation and emulation tools to augment radiation-hardness assurance testing," *IEEE Trans. Nucl. Sci.*, vol. 60, no. 3, pp. 2119–2142, Jun. 2013.

[8] M. Alderighi *et al.*, "Evaluation of single event upset mitigation schemes for SRAM based FPGAs using the FLIPPER fault injection platform," in *Proc. 22nd IEEE Int. Symp. Defect Fault-Tolerance VLSI Syst. (DFT)*, Sep. 2007, pp. 105–113.

[9] M. Alderighi, S. D'Angelo, M. Mancini, and G. R. Sechi, "A fault injection tool for SRAM-based FPGAs," in *Proc. 9th IEEE On-Line Test. Symp. (IOLTS)*, Jul. 2003, pp. 129–133.

[10] F. Lima, C. Carmichael, J. Fabula, R. Padovani, and R. Reis, "A fault injection analysis of virtex FPGA TMR design methodology," in *Proc. 6th Eur. Conf. Radiat. Its Effects Compon. Syst. (RADECS)*, 2001, pp. 275–282.

[11] F. Faure, R. Velazco, and P. Peronnard, "Single-event-upset-like fault injection: A comprehensive framework," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2205–2209, Dec. 2005.

[12] J. C. Fabero *et al.*, "Single event upsets under 14-MeV neutrons in a 28-nm SRAM-based FPGA in static mode," *IEEE Trans. Nucl. Sci.*, vol. 67, no. 7, pp. 1461–1469, Jul. 2020.

[13] A. Gruwell, P. Zabriskie, and M. Wirthlin, "High-speed programmable FPGA configuration through JTAG," in *Proc. 26th Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2016, pp. 257–260.

[14] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Des. Test. IEEE Des. Test. Comput. Comput.*, vol. 17, no. 3, pp. 44–53, Jul. 2000.

[15] 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.

[16] W. E. Ricker, "The concept of confidence or fiducial limits applied to the Poisson frequency distribution," *J. Amer. Stat. Assoc.*, vol. 32, no. 198, pp. 349–356, Jun. 1937.