RadPC: A Novel Single-Event Upset Mitigation Strategy for Field Programmable Gate Array–Based Space Computing

Christopher M. Major,* Annie Bachman,† Colter Barney,† Skylar Tamke,‡ and Brock J. LaMeres§
Montana State University, Bozeman, Montana 59715

https://doi.org/10.2514/1.I010859

This paper presents a computer architecture with the ability to respond to radiation-induced faults. The architecture presented advances the state-of-the-art by implementing a redundant multiprocessor system on a commercial-off-the-shelf field programmable gate array. The system, called *RadPC*, can respond to radiation-induced faults and undergo repairs as necessary. A reliability analysis is conducted using Markov chain models to characterize the architecture's ability to continue performance despite faults. Further analysis regarding expected radiation exposure is provided for runtime context of the architecture.

Nomenclature

p	=	probability of failure
R	=	reliability
T	=	transition matrix
α	=	rate of context switch, s^{-1}
λ	=	radiation induced fault rate, s^{-1}
μ	=	repair rate, s^{-1}

I. Introduction

COMPUTING systems operating on the surface of the Earth run without concern of the harsh radiation environments found in space. However, computing systems for space science and space exploration missions do not have the same luxury. Space computers suffer from the unwanted effects of cosmic energy, unlike the billions of computers and embedded systems in use throughout the world, which are effectively shielded from energized particles by the Earth's atmosphere and magnetic field. No such protection is guaranteed for computers outside of the planet's protective radiation shielding [1]. The demand for data processing in space science has become more apparent as the complexity of exploration systems increases, and thus the need for radiation-responsive features has become an area of concern.

There are two classes of space radiation effects on computing platforms: single-event effects (SEE) and total ionizing doses (TIDs) [2]. A single-event upset (SEU) occurs when high-energy particles or heavy ions strike a complementary metal oxide semiconductor (CMOS) device and cause unintended, logic-level transitions—an essentially instantaneous impact on performance. Such transitions can be divided into three subcategories, depending on their effects within the fabric. A single-event transient (SET) occurs when an energized particle changes the voltage of a logic line and therefore changes its logic value. When such a change is stored in a memory device, such as a latching circuit or a D-Flip Flop, an SEU has occurred. Each of these event types can often be corrected with a reset, but if an event causes such malfunction that further counteractive measures such as power cycling the entire system must be taken, the event is classified as a single-event functional interrupt (SEFI).

A TID occurs when lower-energy particles strike a CMOS device and cause physical damage. This is manifested as trapped charges left inside the insulating layers of a CMOS transistor lock the device into a perpetual on or off state, allowing for continuous and uninterrupted current draw. This results in large power draw by the part, interference with the device's tasks, and the eventual material degradation of the device over time. Where SEEs are momentary, TIDs are gradual and demonstrate cumulative effects that are measured as the amount of energy per unit of mass trapped in the material [3].

In modern integrated circuit design (<65 nm), TID is becoming more and more statistically unlikely to occur due to feature sizes with small process nodes [4]. While the smaller feature sizes are reducing the possibility of damage due to trapped charge, the probability of functionality interruption caused by high-energy particles is drastically increased. Thus, the need to mitigate the damaging effects from SEEs becomes of greater concern than TID in modern computing systems. This need increases in importance considering the trend toward shorter mission lifetimes (i.e., small satellites), where the computers are not in space long enough to accumulate TID damage.

In this paper, we propose and detail a radiation tolerant computer system called RadPC, which is designed to respond to the effects of SEUs. We describe an architecture that uses redundant processors configured in N-modular redundancy on a commercial-off-the-shelf (COTS) field programmable gate array (FPGA). Detection and repair of SEU-induced faults are accomplished through a comprehensive strategy including partial reconfiguration (PR) of impacted cores, error correction codes (ECC) for memory that cannot be partially reconfigured, and soft error mitigation monitoring for the configuration memory of the FPGA. We also provide a modeled analysis of the fault-tolerance of RadPC regarding times needed for reconfiguration of the system to flush out SEUs.

II. Motivation

A. Existing TID Mitigation Strategies

As radiation-based faults are a great concern to spaceflight hardware, there currently exist various methods intended to prevent or mitigate their effects. A basic solution for TID mitigation is the concept of shielding, in which some sort of material serves as a boundary between cosmic radiation and the hardware at risk. Within many ground-based nuclear facilities, shielding is required for device and personnel protection and is implemented with little concern as to the geometry needed for protection.

In the case of spaceflight hardware, however, the mass and volume of a component can greatly impact the cost-effectiveness of the mission to detrimental levels as these metrics directly drive the costs for launch service. The thickness of a shield or the density of a material increases as the radiation dosage levels decrease, thus dramatically increasing the costs needed to transport said hardware into space [5]. Thus, shielding may succeed in protecting a device from

Received 20 May 2020; revision received 18 December 2020; accepted for publication 22 February 2021; published online 29 March 2021. Copyright © 2021 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the eISSN 2327-3097 to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

*Graduate Research Assistant, Electrical and Computer Engineering Department, 324 Norm Asbjornson Hall.

†Undergraduate Research Assistant, Electrical and Computer Engineering Department, 324 Norm Asbjornson Hall.

‡Research Engineer, Space Science and Engineering Laboratory, Cobleigh Hall.

§Professor, Electrical and Computer Engineering Department, 316-C Norm Asbjornson Hall. Member AIAA (Corresponding Author).

TIDs but is likely to be ineffective in protection from SEEs when considering the cost and mass need to prevent secondary radiation.

Radiation-hardened semiconductor manufacturing modifications are another method of mitigating radiation-induced faults in system hardware [6]. One such technique, known as radiation hardened by design (RHBD), mitigates TIDs by using nonstandard layout techniques to reduce the probability of low-energy particles depositing charge within a device's insulating components. This is accomplished through techniques such as triple modular redundancy (TMR) implementation, enclosed-layout transistors (ELTs), and guard rings—elements designed to reroute the flow of low-energy particle charge through safer routes so as not to compromise the device's performance [7]. Additional components for error detection and correction in memory are often used in such designs to maintain data integrity [8].

Another technique to mitigate TID is known as radiation hardened by process (RHBP). RHBP is a practice that uses nonstandard materials during fabrication to reduce the susceptibility probability of trapped charges. In doing so, defects are minimized in the substrate and oxide layers of the semiconductors, thus reducing the probability of radiation-induced electron/hole pairs in the material [9].

Systems designed with RHBD and RHBP techniques, however, require larger circuit surface area and therefore experience poorer power efficiency, speed, and performance than standard, commercially manufactured systems. Their dedicated manufacturing processes, combined with the relatively low market demand in comparison with commercial processes, greatly increase the costs of device production. Device performance is also greatly inhibited by these constraints, as RHBP-developed hardware tends to lag commercially developed equivalents by at least a decade [10].

B. Existing SEE Mitigation Strategies

To mitigate SEEs, one commonly used technique is TMR. TMR is an approach in which a logic circuit is triplicated with identical implementations. Each circuit's output feeds into a voter device that determines the correct value of the original circuit through selection by majority. This strategy seeks to mitigate the effects of SEEs by providing the ability to continue operation in the presence of an SEE-induced fault [11]. TMR requires additional area and components for proper implementation, increasing costs, power, and design complexity. Additional overhead is needed for proper voter performance in the system. A block diagram demonstrating the basic structure of a TMR system is shown in Fig. 1.

Memory in computing systems is uniquely susceptible to SEEs due to the high layout density and large area requirements on the substrate. Memory typically occupies the largest area on the substrate in a computing system. This limits the use of TMR and/or RHBD techniques prohibitive from an area-usage standpoint. The most common techniques to mitigate errors in memory used for data are error correction codes [8].

For memory that is used for program information or configuration data, a technique known as memory scrubbing is often used to mitigate the effects of SEEs. This is accomplished by using a device in hardware or a process in software to periodically iterate through the contents of a memory device and check their values against expected values. Blind scrubbing overwrites the memory cell regardless of whether an error occurred. Readback scrubbing detects if there is any discrepancy, then overwrites the faulty data with good data [12]. Scrubbing has been demonstrated as an effective technique for increasing system reliability and ensuring data integrity. Scrubbing is a common technique for mitigating SEUs in the configuration memory of FPGAs. Faults in the configuration memory of an FPGA represent faults that cannot be repaired using normal reset procedures, and thus they are considered a SEFI.

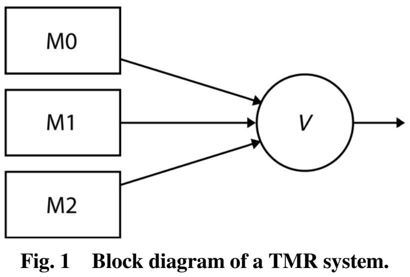

Fig. 1 Block diagram of a TMR system.

C. Our Contribution

Our work contributes to the field of space computing by providing a cost-effective, radiation tolerant design using a commercial-off-the-shelf FPGA to actively respond to radiation-induced faults in the device. Instead of attempting to impede radiation-induced faults, our design seeks to mitigate their effects as a radiation-tolerant computer system. As advances in semiconductor manufacturing technology have led to decreased gate oxide thicknesses, reducing the possibility of trapped charges from TID at processes of 45 nm or smaller [4], our choice of a modern, commercial FPGA has a lower probability of sustaining damage from low-energy particles. Our strategy implements a redundant, multiprocessor system, a background configuration memory scrubber, and data memory error correction codes to mitigate the effects of SEEs. Our approach has the potential to increase the reliability of space computers while simultaneously reducing the cost, compared with existing systems that exploit RHBD/RHBP techniques.

III. System Design

The architecture of RadPC adapts the strategies of TMR and memory scrubbing by taking advantage of the configuration properties of FPGAs. An FPGA implements digital logic by arranging logic elements according to a design bitstream loaded into its configuration memory. When power is turned on, the device configures all logic, interconnects, inputs, and outputs according to this bitstream. This is known as full configuration. FPGAs provide the flexibility that any design can be implemented and synthesized into an FPGA bitstream. Originally, this capability afforded a unique environment for prototyping. Recently FPGA performance has reached a level that makes implementation in a final system practical. In our work, we exploit the reconfigurability of FPGAs to dynamically repair and recover from faults.

If correctly enabled by the designer, an FPGA can select a predefined subset of the hardware and reconfigure it with a different design. This is known as partial reconfiguration (PR). In PR, a partial bitstream is loaded into a select section of configuration memory to drive this change in logic. This section is often referred to as a "PR region" of the FPGA fabric. Two methods of PR are often used in FPGA design, depending on the state of the device during the process. When the device is shut down or inactive and a partial bitstream is loaded, static PR has occurred. When the device is currently performing its designated task without power cycling or suspending activity outside the PR region, active PR has occurred [13]. PR has a unique role in our FPGA-based SEE mitigation strategy. When a fault occurs in either the foreground circuitry of the FPGA or in the configuration memory region corresponding to the foreground circuitry, PR will restore that region to its original state. This effectively repairs the fault caused by the SEE [14,15]. A design approach can be used that divides the system into various PR regions and incorporates PR as part of the fault-mitigation strategy. This does lead to synchronization issues that must be addressed, which will be covered later.

Our design employs a variety of fault-mitigation strategies based on a redundant multiprocessor system implemented within the FPGA fabric. Figure 2 demonstrates the general layout of this design.

A. Fault Mitigation Strategy

Four parallel processors, denoted as "tiles," are designated as PR regions and run identical software. This system extends the capability of TMR to continue operation in the presence of a fault. TMR can withstand one fault in the triad and still produce the intended output. However, when the fault occurs due to an SEU of SEFI, the system needs to stop foreground operation and restore the faulted system. In FPGA-based computer systems, this typically means either

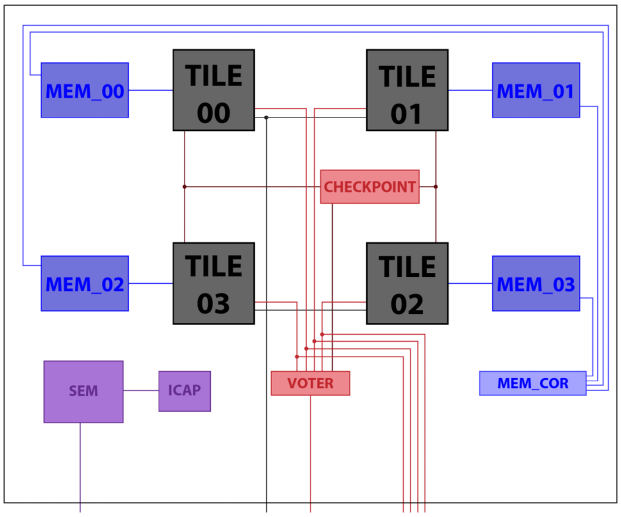

Fig. 2 Block diagram of RadPC architecture.

reconfiguring the entire FPGA or scrubbing the entire configuration memory array. The significant downtime associated with repairing the faulty member of the triad reduces the reliability of the system for two reasons. First, the time associated with full reconfiguration or full scrubbing of the computer system dramatically reduces the availability of the system. Second, the system is still vulnerable to radiation strikes while performing the repair. If a second strike occurs during repair, the system is put into a state that cannot be recovered from. In our prior work, we discovered that increasing the number of redundant members in the system (N) reduces the possibility of a subsequence radiation strike putting the system into an unrecoverable state [16]. Our prior work also showed that the most gain in reliability was achieved with a four-member system, called “4MR.” Adding more members beyond four did not result in any significant reliability gains because the additional circuit area that was added was also susceptible to radiation strikes, so there was diminishing returns. Thus, a four-modular redundant (4MR) system is optimal.

Additionally, the use of PR to repair a faulty tile provides a much faster recovery process than a full FPGA reconfiguration. As tasks are completed by each tile, any output results are fed into a voter circuit and analyzed for discrepancy by majority vote. If a tile shows an incorrect value, it is partially reconfigured to flush out the SEU(s) in its PR region. When this occurs, the remaining tiles continue to proceed with the next software task.

After the faulty tile is partially reconfigured, it is ready to be reintroduced into the voting system. To synchronize the repaired tile with the other three, a software checkpoint system is used. A checkpoint bus is implemented between the four tiles that indicates if another tile has been faulted. Code is entered into the software at desired points that will allow the processor to stop execution of the main program and wait for the partially reconfigured tile to synchronize. The locations of the software checkpoints are at the discretion of the developer in order to provide the ability to prioritize critical code execution over synchronizing the repaired PR tile. For mission critical software, the synchronization can wait until there is a break in important execution.

The voter implements additional functionality to serve as the arbitrator of the checkpointing to ensure data integrity and synchronization. The voter observes a register known as the “checkpoint” register, where each tile will set a flag when it has finished executing a given portion of the program. When all four tiles have set their checkpoint flags, the voter receives output data from each of the tiles and checks their values. If these values are all equal, the voter will signal the tiles to continue to the next stage of the program. If these values are not equal, the voter will signal the memory correction component to activate and correct any erroneous memory cells before letting the tiles resume the program.

Each tile accesses an on-FPGA memory block for its data memory. On an FPGA, the user memory is implemented with SRAM technology, which is not impacted by a full or PR. This is advantageous from an operational standpoint because the data are not altered during a PR; however, additional fault mitigation must be included in the system to protect the data memory. In our system, each data memory block employs an ECC system to detect and correct faults that may have occurred in the SRAM cells. Each memory device consists of a data encoder, a memory device, and a data decoder. When a tile writes to this memory device, the data are encoded from an 8-bit format to a 12-bit format. This is accomplished using a Hamming encoder device to add four error correction code bits, which are used in the decoding process to evaluate possible data corruption. When a tile reads from the memory device, the data from the memory device are decoded from a 12-bit format into an 8-bit format. This is accomplished using a Hamming decoder device to check the consistency between the error correction bits and the original data, then attempts to correct any errors present before letting the tile general purpose input/output (GPIO) read the value.

Although the ECCs handle correcting any radiation-induced faults in the data memory, they do not have the ability to synchronize the contents with the other three memory blocks in the event of a tile PR. To handle this, a memory correction system is implemented. The encoded data from each tile is stored in a dual port read/write memory unit. One port is dedicated to tile use; the other is used for the separate memory correction component. The memory correction component accesses a tile’s memory block’s second port, bypassing the Hamming encoder and decoder to directly assess the memory contents. It then evaluates the contents of all four memory units simultaneously, comparing them to see which value is inconsistent with the others. If a value is inconsistent, it will be rewritten with the majority value of its neighboring units at that address.

For SEUs that occur outside the FPGA fabric designating a tile, a soft error mitigation (SEM) controller is used as the configuration readback memory scrubber [17]. This is a Xilinx-provided IP, connected to the ICAP primitive and a UART communication port, that uses error correction codes within the Artix-7 configuration memory to check for single-bit and double-bit adjacent upsets. The controller runs perpetually, independent of the tiles, memory, and voter, to monitor and scrub errors in background operation. When a PR on a tile is performed, the SEM controller is temporarily deactivated to avoid error correction conflicts. Upon completion of the PR, a software reset command is sent to the SEM controller and its monitoring process continues.

To protect the configuration memory itself from faults, a readback memory scrubber can be used. Using the memory’s error correction codes and frame address register (FAR), a scrubber can iterate through the memory frame by frame and check the contents for single-bit or adjacent double-bit errors [18]. In the event of an error, the scrubber replaces the faulty bits in memory and resumes operation. If a scrubber is used alongside a PR strategy, an arbitration system is necessary to prevent interference between the two systems. The Xilinx FPGA provides a built-in configuration memory repair system called an SEM block. This is implemented on our system.

B. Unrecoverable Faults

In any system, there will be faults from which the system cannot be recovered. In our system, multiple faults occurring simultaneously can prevent successful recovery, as majority-based voting strategies can fail to select the correct values based on expected values. Our design is based on the premise that fault recovery is implemented on high probability faults and lower probability faults represent infrequent events that require either a full FPGA configuration or a full system power cycle to mitigate their effects.

C. Fault Mitigation Summary

Table 1 summarizes the range of possible faults, fault-mitigation strategies, and probability of risk.

D. Implementation

This architecture was prototyped on a Basys 3 development board, produced by Digilent. This board features an Artix-7 XC7A35T CGP326-1, a low-power, low-cost device available for purchase from

Table 1 Fault mitigation strategies for upsets in RadPC

Fault condition	Observed by	Action taken	Fault severity	Fault possibility
SEU in tile foreground circuitry	Voter	PR the effected tile	Medium	Medium
SEU in non-tile foreground circuitry	Voter	Full FPGA reconfiguration	High	Low
Single-bit error in on-FPGA data memory	ECC	Automatic data recovery by ECC decoder	Low	Medium
Multibit error in on-FPGA data memory	Voter	Overwrite corrupted data memory with values from healthy data memory	Medium	Low
Single-bit error in FPGA configuration memory	SEM	Automatic correction	Low	High
Double-bit adjacent error in FPGA configuration memory	SEM	Automatic correction	Low	Medium
Multibit, nonadjacent error in FPGA configuration memory	SEM	Full FPGA reconfiguration	High	Low

Xilinx [19]. The hardware design was developed using the Vivado 2019.2 design suite, and the software to be implemented for demonstration was developed using Xilinx's Vitis development suite.

Each tile consists of a Xilinx MicroBlaze softcore processor, 8 KB of program memory, four GPIO devices with two channels each, and a processor reset system. A single program runs on the MicroBlaze, using the GPIO to access an input register, a memory device, the Voter system, and the Checkpoint register. All these components are contained in a Vivado constraints file and wrapped in a hardware wrapper to allow for instantiation in the top-level VHDL file. The tiles are denoted as TILE_00, TILE_01, TILE_02, and TILE_03.

Each tile has a corresponding partial bitstream, which can be used to reset the FPGA fabric in the predefined tile region and bring the processor and its connected components back online. The memory, voter, checkpoint register, and all other external components are not affected by this reconfiguration. Though a reconfigured tile may lag the other tiles in its program's progress, the checkpoint system ensures that the other tiles will not advance in the program until the affected tile can catch up.

IV. Characterization

The reliability of this system depends on the amount of time that the various aspects of recovery take in addition to the predicted rate of incoming faults. This drives the system's availability and mean time to failure (MTTF), but is also dependent on the environment that the computer resides in. The first step is determining the time of the repair strategies in this computer.

Measurements regarding the time needed to perform full configuration and PRs of the device were made by measuring the DONE pin of the Basys 3 board using an oscilloscope. According to Xilinx's 7-Series FPGA configuration guides, the DONE pin asserts when a full configuration is completed or when a PR is completed. Thus, this signal was used to evaluate the speeds of both methods.

A full reconfiguration of the system, using the top-level bitstream, takes an average of 1.24 s to complete. This test was conducted using the joint test action group (JTAG) configuration port to load a 17,536,096 bit configuration bitstream into the FPGA. According to Vivado's bitstream generation reports, 2,793,730 of the 14,663,584 bits of configuration memory, or 19.05%, are classified as "essential bits" that are prioritized in the SEM controller's operation.

The PR of a tile will alter its completion time, depending on the tile to be reconfigured. This is due to the floor planning conducted to allow for the generation of a partial bitstream for each tile, as the XC7A35T has limited fabric space for the positioning of "PBlocks" assigned to various components of the design. Thus, all four tiles differ in position, number of logic elements per PBlock, size, and accessibility to clock and reset lines. PR of a RadPC tile, therefore, ranges between averages of 217 and 434 ms. Specifically, Tile 0 takes an average of 217 ms to be reconfigured, Tile 1 takes an average of 434 ms, Tile 2 takes an average of 404 ms, and Tile 3 takes an average of 217 ms.

Faster configuration times can be obtained by using a different programming interface to the FPGA such as slave serial. For our prototype, the slave serial bus was not connected on the Basys 3 board so it could not be empirically measured. However, an assumption can be made that the ratio of PR to full configuration will remain the same when moving from JTAG to slave serial. This ratio for the largest PR is 35% (i.e., 434 ms/1.24 s). Calculating the amount of time that a slave serial configuration takes, it was determined that a full configuration will take 1.101 s and a worst-case PR will take 418 ms [20,21].

Per Vivado's utilization reports and the available resources on the Artix-7 XC7A35T chip, 12,882 Slice Registers out of 41,600 (30.97%) and 12,068 Slice LUTS out of 20,800 (58.02%) are used in the design, and 23 Block RAM Tiles out of the available 50 (46%) are likewise used. Vivado's timing summary reports that the worst slack for the setup time in the design is 23.790 ns and the worst pulse width slack time is 15.250 ns.

Vivado's power report estimates a total on-chip power of 0.102 W, with dynamic power accounting for 0.029 W and device static accounting for the other 0.073 W. A junction temperature of 25.5°C was specified in the report. This does not account for I/O power on the Basys board itself, merely representing an estimate of the FPGA device itself.

V. Reliability Analysis

A. Markov Chain Models

To estimate the reliability of the proposed architecture, Markov chain models are used [16]. Markov models are used to model a system using state diagrams, linked together by probabilities describing the likelihood of a transition from one state to another. As such, they assume that, in their most basic form, a transition to another state relies only on the current state of the system [22]. This principle makes them a useful tool for analyzing system reliability. The Markov model will transition to a new state upon a fault in the computer. Since not every radiation strike will cause a fault in the computer system, the fault rate fed into the Markov model is scaled down to a value that considers the essential bits on the FPGA (i.e., those that represent implemented circuitry) and a derating for the type of system implement (i.e., 30% for a microprocessor) [23].

In a Markov chain model, a system is represented by a transition matrix of size (m, n), where each entry represents the probability of transitioning from state m to state n. In the example of a simple two-state system with states S_0 and S_1, the transitions can be represented with a 2×2 matrix filled with a probability of each transition in each entry, where λ represents the fault rate of the system. Figure 3 demonstrates this model as a state diagram. This model represents a simplex computer system where state S_0 represents an operational system and state S_1 represents a faulted system.

For a general state diagram with s states, a transition matrix T of size$S \times S$ can be created to represent the probability of all transitions t from one state m to another state n, as shown in Eq. (1):

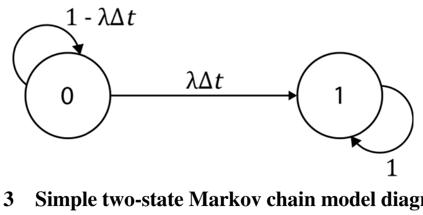

Fig. 3 Simple two-state Markov chain model diagram.

$$T(m,n) = \begin{bmatrix} t_{1,1} & \dots & t_{1,n} \\ \dots & \dots & \dots \\ t_{m,1} & \dots & t_{m,n} \end{bmatrix} \tag{1}$$

To calculate the probability p of being in a state S given a time step k, the following equation must be used:

$$p_s(t = k\Delta t) = \begin{bmatrix} p_0(0)\dots p_n(0) \end{bmatrix} \cdot \begin{bmatrix} t_{1,1} & \dots & t_{1,n} \\ \dots & \dots & \dots \\ t_{m,1} & \dots & t_{m,n} \end{bmatrix}^k \tag{2}$$

In the case of the two-state diagram in Fig. 3, assume that S_0 represents a state of healthy device operation and S_1 represents a state of faulty or failed device operation. The transition matrix T of this model is

$$T(m,n) = \begin{bmatrix} 1-\lambda\Delta t & \lambda\Delta t \\ 0 & 1 \end{bmatrix} \tag{3}$$

The probability of residing in state S after a time step k, p_s, is given by

$$p_s(t = k\Delta t) = \begin{bmatrix} p_0(0), p_1(0) \end{bmatrix} \cdot \begin{bmatrix} 1-\lambda\Delta t & \lambda\Delta t \\ 0 & 1 \end{bmatrix}^k \tag{4}$$

The reliability is calculated by the following:

$$R(t) = 1 - p_s(t = k\Delta t) \tag{5}$$

As the performance analysis of RadPC hinges on its reliability, the MTTF must be considered. This is defined as the point in time$t$ when the reliability of the system $R(t) = 0.5$.

In the case of RadPC, there are more states to reflect the various capabilities and safeguards in the system. For example, a TMR system can be modeled with three states [16]. State S_0 represents an operational system where none of the three tiles have radiation-induced faults. State S_1 represents the situation where one tile of the triad has been faulted. Note that the system is still operational because two of the three tiles are operational. State S_2 represents a faulted system because two of the three tiles have been faulted and the system can no longer determine the correct output. Figure 4 demonstrates the state diagram used to represent a TMR system as a Markov chain model.

The transition matrix of this system is as follows:

$$T(m,n) = \begin{bmatrix} 1–3\lambda\Delta t & 3\lambda\Delta t & 0 \\ 0 & 1–2\lambda\Delta t & 2\lambda\Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

The probability of an operation TMR system is given as the product of their individual reliabilities, as follows:

$$R(t) = R^3(t) + 3R^2(t)\cdot(1 - R(t)) \tag{7}$$

$$R(t) = 3R^2(t) - R^3(t) \tag{8}$$

ATMR-based system alone, however, has no option of returning to the initial, fully operational state and therefore reduces the reliability of the system. Thus, the option to repair a tile through PR or SEM must be represented in the state diagram. This can be referred to as a TMR + Spare system. The repair rate of the tile is represented as a parameter μ, which can be determined as the worst-case number of repairs per second. Based on the performance of the Basys 3 board, the repair rate was found to be $\mu = 437$ ms. Figure 5 demonstrates the new state diagram, including the PR/SEM recovery methods.

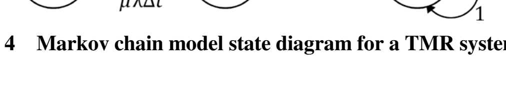

Fig. 4 Markov chain model state diagram for a TMR system.

To further increase the reliability of the system, a fourth tile is used in RadPC [16]. This results in a 4MR system, where the additional tile allows two faults to occur in the system without faulting the system. An assumption is made that in the event of two faults, the two damaged tiles will not produce the same, yet inaccurate, output. As before, S_0 indicates a healthy and fully operational state. If an operating tile is damaged, the system transitions to S_1 and must undergo repairs. The faulty tile is disregarded in the vote process until it is brought back into operation. This is represented in S_2. If the faulty tile cannot be repaired, the system can proceed in S_3 until repairs are made. Only when a system has experienced two unrecoverable faults, the transition to S_4 is made, in which system failure is experienced entirely. Figure 6 demonstrates this behavior as a new state diagram.

The difference between a 4MR system and a TMR + Spare system is in the treatment of the "spare tile." In a 4MR system, the spare actively runs the same program as its neighbors and is considered a valid vote in the overall architecture. When an error occurs in any of the tiles, PR is conducted immediately without swapping in new components, as in the previous TMR + Spare system. The voter will then track any faults in the running tiles and perform the repairs as necessary, including memory scrubbing if necessary.

A comparison between architectures was performed in order to verify that the RadPC architecture using four tiles with repair is significantly more reliable than simplex, TMR, and TMR with repair approaches. The graph in Fig. 7, plotted by running the Markov model transition matrices through a MATLAB script, demonstrates the exponential reliability curve for the different system architectures. For this comparison, a value of $\mu = 437$ ms was used and represents the slowest PR time (i.e., Tile 1). A fault rate of $\lambda = 0.001$ [SEU/ μs] was used in all systems and was chosen arbitrarily.

The plots indicate the increase in reliability when additional features are implemented alongside a basic TMR design. With a singular, or "simplex" system, or an unprotected TMR system, the reliability decays at an exponential rate and provides little to no assurance of system longevity. With a TMR system protected by a scrubbing mechanism, be that the SEM or PR, the reliability drops at a far slower linear rate. Including a spare tile to be included in the TMR + Spare system dramatically decreases the rate of reliability decreases and therefore provides the best system longevity of all possible combinations. Thus, it can be seen that the 4MR system provides the best reliability of all aforementioned systems. It is left to the discretion of a mission planner to consider the range of radiation environments in which RadPC would be exposed, and to assess

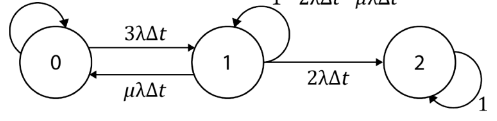

Fig. 5 Markov chain model state diagram for a TMR + PR + SEM system.

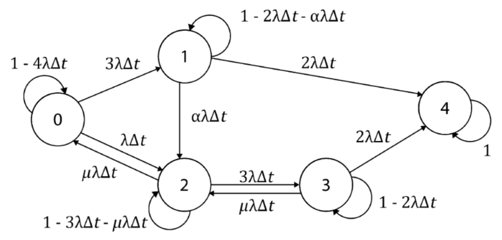

Fig. 6 Markov chain model state diagram for a TMR + PR + SEM + spare tile System.

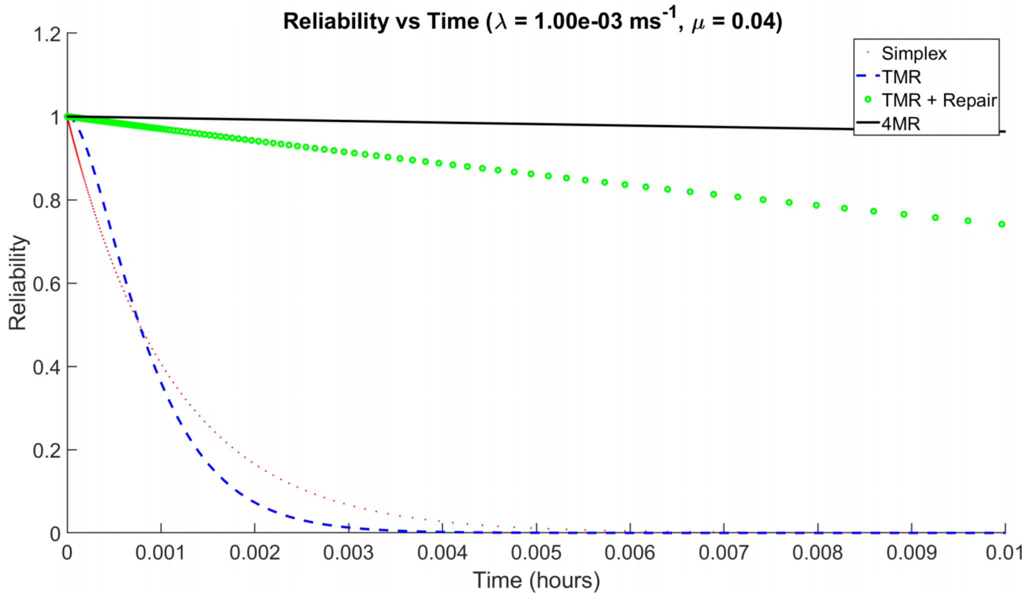

Fig. 7 Exponential reliability curve for four-tile RadPC system with repair capability.

whether an environment's expected fault rate would warrant full reconfiguration of the device.

B. CREME96 Analysis

To characterize the radiation environments in which RadPC would be exposed, a numerical modeling tool known as the Cosmic Ray Effects on Micro-Electronics code (CREME96) was used [24]. Given the dimensions of the Artix-7 XC7A35T chip and the essential bit count provided by the Vivado software suite, a series of estimates were given to predict the levels of direct ionization-induced SEEs the system would experience. Table 2 displays the results of the ISS-environment and lunar-environment analyses. Accounting for the essential bits in the system and the derating for a microprocessor, these fault rates are scaled down by a factor of 0.3. The resulting figures are shown in Table 3.

Generally, the average and peak fault rates are relatively small and do not present a significant risk to the system. A worst-case scenario analysis, however, proves more useful in evaluating the limits of the system. Thus, a script was written in MATLAB to plot the mean-time-to-failure as opposed to fault rate, shown in Fig. 8.

The percentage increase in improvement of each of these architectures, in comparison to the simplex architecture, is shown in Table 4.

The plot and table demonstrate that the architecture that best retains its reliability as the fault rate increases is the 4MR system, given the repair rate $\mu = 437$ ms. Even in the extreme cases of the ISS-orbit and lunar-orbit environments, it performs significantly better than the simplex and TMR architectures in maintaining reliability. As previous iterations of RadPC have been experimented with varying numbers of tiles, it has been found that the gains of additional redundant tiles diminish shortly after adding one. Thus, it can be concluded that the 4MR system is the most optimal architecture of those discussed in this paper.

VI. Discussion

Several considerations are to be taken into account regarding the performance of the RadPC system. As the software synchronization

Table 2 CREME96 modeling of ISS-orbit and lunar-orbit radiation environments

Essential bits analysis (2,793,730)		Total bits analysis (14,663,584)	
Lunar environment, average conditions	Device/day	Lunar environment, average conditions	Device/day
Average	1.969	Average	10.336
Peak	5.463	Peak	28.676
Lunar environment, inside Earth's magnetosphere, worst-case scenarios	Device/day	Lunar environment, inside Earth's magnetosphere, worst-case scenarios	Device/day
Worst week	166,696.000	Worst week	874,944.000
Worst day	910,118.000	Worst day	4,776,980.000
Worst 5 min	312,909,000.000	Worst 5 min	1,642,380,000.000
Lunar environment, outside Earth's magnetosphere, worst-case scenarios	Device/day	Lunar environment, outside Earth's magnetosphere, worst-case scenarios	Device/day
Worst week	166,712.000	Worst week	875,200.000
Worst day	910,207.000	Worst day	4,778,370.000
Worst 5 min	3,473,810.000	Worst 5 min	18,236,700.000
ISS environment, stormy solar conditions	Device/day	ISS environment, stormy solar conditions	Device/day
Average	1,179.700	Average	6,192.300
Peak	2,232.800	Peak	11,721.700
ISS environment, quiet solar conditions	Device/day	ISS environment, quiet solar conditions	Device/day
Average	29.046	Average	152.470
Peak	82.695	Peak	434.109
ISS environment, worst-case scenarios	Device/day	ISS environment, worst-case scenarios	Device/day
Worst week	9,939.990	Worst week	52,182.500
Worst day	54,377.500	Worst day	285,468.000
Worst 5 min	207,490.000	Worst 5 min	1,089,270.000

Table 3 CREME96 modeling of radiation environments, 30% microprocessor derating

Essential bits analysis (2,793,730)		Total bits analysis (14,663,584)	
Lunar environment, average conditions	Device/day	Lunar environment, average conditions	Device/day
Average	0.591	Average	3.101
Peak	1.639	Peak	8.603
Lunar environment, inside Earth's magnetosphere, worst-case scenarios	Device/day	Lunar environment, inside Earth's magnetosphere, worst-case scenarios	Device/day
Worst week	50,008.800	Worst week	262,483.200
Worst day	273,035.400	Worst day	1,433,094.000
Worst 5 min	93,872,700.000	Worst 5 min	492,714,000.000
Lunar environment, outside Earth's magnetosphere, worst-case scenarios	Device/day	Lunar environment, outside Earth's magnetosphere, worst-case scenarios	Device/day
Worst week	50,013.600	Worst week	262,560.000
Worst day	273,062.100	Worst day	1,433,511.000
Worst 5 min	1,042,143.000	Worst 5 min	5,471,010.000
ISS environment, stormy solar conditions	Device/day	ISS environment, stormy solar conditions	Device/day
Average	353.910	Average	1,857.690
Peak	669.849	Peak	3,516.510
ISS environment, quiet solar conditions	Device/day	ISS environment, quiet solar conditions	Device/day
Average	8.714	Average	45.741
Peak	24.809	Peak	130.233
ISS environment, worst-case scenarios	Device/day	ISS environment, worst-case scenarios	Device/day
Worst week	2,981.997	Worst week	15,654.750
Worst day	16,313.250	Worst day	85,640.400
Worst 5 min	62,247.000	Worst 5 min	326,781.000

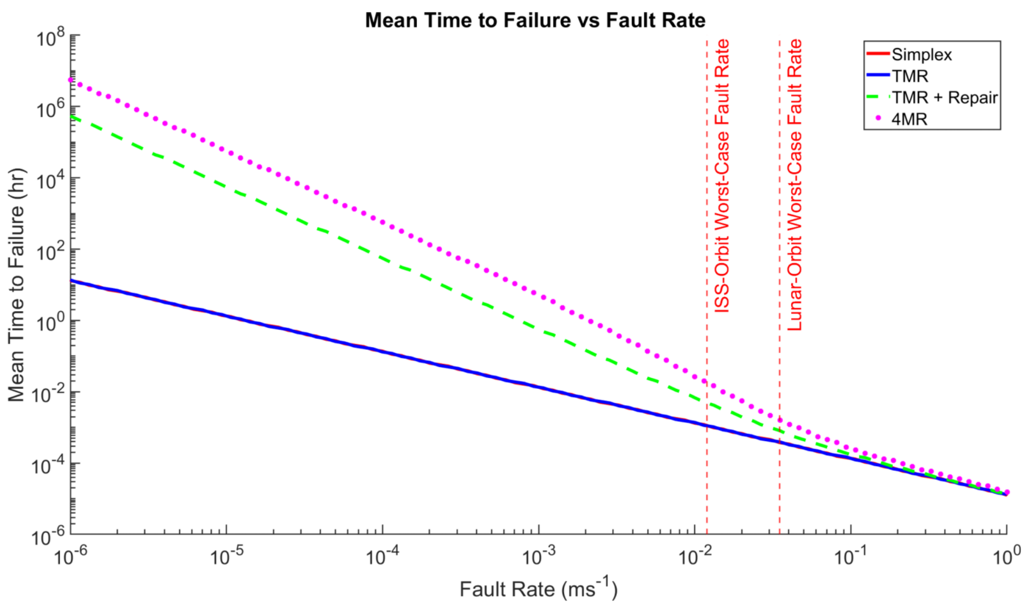

Fig. 8 MMTF versus fault rate for varying architectures.

Table 4 CREME96 modeling of radiation environments, 30% microprocessor de-rating

Architecture	% Improvement over simplex system	
	ISS-orbit worst case	Lunar-orbit worst case
TMR	4.7	4.9
TMR + repair	327	117
4MR	1556	327

checkpoints play a vital role in the timing of PRs, two approaches to planning checkpoints have to be considered by software developers. The first approach would be to space checkpoints over long periods of time, allowing for multiple PRs to complete before checking tile outputs. The second approach would be to shorten the time between checkpoints, thus reducing the amount of time for a faulty tile to continue without repair. Though the former approach would perform well in low-fault conditions, it is recommended that the latter approach be taken so as to quickly assess and correct faults that may occur in a tile, instead of letting errors accumulate between spaced checkpoints.

The time needed to complete a full or a PR depends on the frequency of the external monitoring device performing the bitstream upload. Thus, the μ value of a slave serial configuration can be found by running Vivado's configuration time calculator tool. The slave serial configuration time previously given assumed a 16 MHz serial link from a microcontroller; however, this does not guarantee the fastest possible configuration. Given a controller configuration frequency of 100 MHz, the configuration time drops to 39 ms minimum and μ drops to 0.0039. As can be seen in Fig. 9, plotted by a MATLAB script, the MTTF increases as μ decreases.

Of additional consideration is the improvement in reliability between the 4MR system with best-performing μ and the simplex system. This architecture exhibits an improvement percentage of upward to 3567% of the simplex system in ISS-orbit and upward to 9359% of the simplex system in lunar orbit.

Further analysis of this system will lead to the consideration of single points of failure, specifically the voter and the data memory

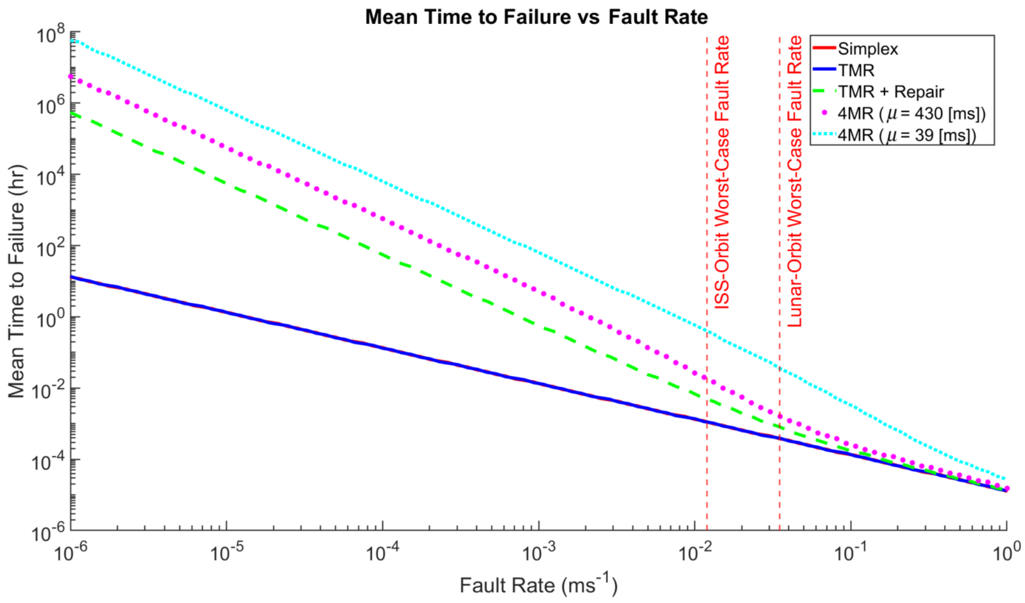

Fig. 9 MMTF versus fault rate for varying architectures, best-performing μ for 4MR system.

scrubber. Though these areas of the device are subject to SEUs, like any other component in the FPGA, their relatively small footprints reduce their probability of experiencing damage from faults. The SEM will protect the configuration memory of these elements from damage, but, otherwise, an error in their circuitry may require a full reconfiguration. This could be mitigated through implementing these elements on an external, radiation-hardened component, but such a strategy would defeat the purpose of developing a COTS design. Rather, the recommended approach would be to establish partially reconfigurable sections of the device dedicated to each component and to employ PR if damage is observed.

Finally, the possibility of tile desynchronization by a few clock cycles, even after checkpoint synchronization occurs, can become a point of concern in device operation. The choice to attempt synchronization again to constrain these times remains valid, or to create a voter that considers outputs across a wider range of clock cycles to account for errors.

VII. Conclusions

The computer architecture presented in this paper, named *RadPC*, demonstrates the ability to respond to faults induced by space radiation. As many current radiation-fault mitigation strategies in space mission computing systems have proven to be infeasible due to cost and complexity, alternate methods to achieve radiation tolerance are necessary. In addressing the impact of SEEs within the circuitry of FPGA-based computers, several methods for mitigating their impact on performance were discussed and implemented within the system. A multiprocessor system inspired and adapted from the common method of TMR was implemented, with the ability to repair impacted tiles and restore functionality. This capability is achieved using PR and SEM to account for SEE-affected portions of the FPGA fabric. Redundant memory devices, checkpoint flags, and a voter system ensure processor tile synchronization. This system was run through a Markov chain model to characterize its ability to maintain performance despite radiation-induced faults. With expected radiation strikes provided by a model output through CREME96, the design demonstrates the ability to provide a feasible, low-cost, low-power, COTS architecture to address radiation-induced faults reliably and expediently.

Acknowledgments

This material is based upon work supported by NASA under Grant/Contract/Agreement Nos. 80 MSFC20C0032, 80NSSC18 P2129, 80NSSC20 K0107, 80NSSC19M0037, 80NSSC18 K1671, 80NSSC17M0037, NNX16AI75A, NNX14AL03A, NNX13AR03A, NNX12AM50G, NNX10AN91A, and NNX10AN32A.

References

[1] Barth, J. L., Dyer, C. S., and Stassinopoulos, E. G., "Space, Atmospheric, and Terrestrial Radiation Environments," *IEEE Transactions on Nuclear Science*, Vol. 50, No. 3, 2003, pp. 466–482. https://doi.org/10.1109/TNS.2003.813131

[2] Claeys, C., *Radiation Effects in Advanced Semiconductor Materials and Devices*, Springer, Berlin, 2010, pp. 1–4, 181–183. https://doi.org/10.1007/978-3-662-04974-7

[3] Adams, A., and Holmes-Siedle, L., *Handbook of Radiation Effects*, Oxford Univ. Press, New York, 2004, pp. 4, 138–163. https://doi.org/10.5860/CHOICE.31-5512

[4] Barnaby, H. J., "Total-Ionizing-Dose Effects in Modern CMOS Technologies," *IEEE Transactions on Nuclear Science*, Vol. 53, No. 6, 2006, pp. 3103–3121. https://doi.org/10.1109/TNS.2006.885952

[5] Uzel, R., and Özyildirim, A., "A Study on the Local Shielding Protection of Electronic Components in Space Radiation Environment," *2017 8th International Conference on Recent Advances in Space Technologies (RAST)*, IEEE, New York, 2017, pp. 295–299. https://doi.org/10.1109/RAST.2017.8003007

[6] Gambes, J. W., and Maki, G. K., "Rad-Tolerant Flight VLSI from Commercial Foundries," *Proceedings of the 39th Midwest Symposium on Circuits and Systems*, Vol. 3, IEEE, New York, 1996, pp. 1227–1230. https://doi.org/10.1109/MWSCAS.1996.593127

[7] Anelli, G., Campbell, M., Delmastro, M., Faccio, F., Floria, S., Giraldo, A., Heijne, E., Jarron, P., Kloukinas, K., Marchioro, A., Moreira, P., and Snoeys, W., "Radiation Tolerant VLSI Circuits in Standard Deep Submicron CMOS Technologies for the LHC Experiments: Practical Design Aspects," *IEEE Transactions on Nuclear Science*, Vol. 46, No. 6, 1999, pp. 1690–1696. https://doi.org/10.1109/23.819140

[8] Yu-Lam, D., Lan, J., McMurchie, L., and Sechen, C., "SEE-Hardened-by-Design Area-Efficient SRAMs," *2005 IEEE Aerospace Conference*, IEEE, New York, 2005, pp. 1–7. https://doi.org/10.1109/23.819140

[9] Makihara, A., Midorikawa, M., Yamaguchi, T., Iide, Y., Yokose, T., Tsuchiya, Y., Arimitsu, T., Asai, H., Shindou, H., Kuboyama, S., and Matsuda, S., "Hardness-by-Design Approach for 0.15 /spl mu/m Fully Depleted CMOS/SOI Digital Logic Devices with Enhanced SEU/SET Immunity," *IEEE Transactions on Nuclear Science*, Vol. 52, No. 6, 2005, pp. 2524–2530. https://doi.org/10.1109/TNS.2005.860716

[10] Keys, A., Adams, J., Frazier, D., Patrick, M., Watson, M., Johnson, M., Cressler, J., and Kolawa, E., "Developments in Radiation-Hardened Electronics Applicable to the Vision for Space Exploration," *AIAA SPACE 2007 Conference and Exposition*, AIAA Paper 2007-6269, 2012. https://doi.org/10.2514/6.2007-6269

[11] Sterpone, L., and Violante, M., "Analysis of the Robustness of the TMR Architecture in SRAM-Based FPGAs," *IEEE Transactions on Nuclear Science*, Vol. 52, No. 5, 2005, pp. 1545–1549.
https://doi.org/10.1109/TNS.2005.856543

[12] Garvie, M., and Thompson, A., "Scrubbing Away Transients and Jiggling Around the Permanent: Long Survival of FPGA Systems Through Evolutionary Self-Repair," *Proceedings of 10th IEEE International On-Line Testing Symposium*, IEEE, New York, 2004, pp. 155–160.
https://doi.org/10.1109/OLT.2004.1319674

[13] Kao, C., "Benefits of Partial Reconfiguration," *Xcell Journal*, Vol. 55, Oct. 2005, pp. 65–67.

[14] Gauer, C., LaMeres, B. J., and Racek, D., "Spatial Avoidance of Hardware Faults Using FPGA Partial Reconfiguration of Tile-Based Soft Processors," *2010 IEEE Aerospace Conference*, IEEE, New York, 2010, pp. 1–11.
https://doi.org/10.1109/AERO.2010.5446663

[15] Julien, C. R., LaMeres, B. J., and Weber, R. J., "An FPGA-Based Radiation Tolerant SmallSat Computer System," *2017 IEEE Aerospace Conference*, IEEE, New York, 2017, pp. 1–13.
https://doi.org/10.1109/AERO.2017.7943634

[16] Hogan, J. A., Weber, R. J., and LaMeres, B. J., "Reliability Analysis of Field-Programmable Gate-Array-Based Space Computer Architectures," *Journal of Aerospace Information Systems*, Vol. 14, No. 4, 2017, pp. 247–258.
https://doi.org/10.2514/1.I010481

[17] Soft Error Mitigation v4.1 Product Guide, Xilinx, 2018, https://www.xilinx.com/support/documentation/ip_documentation/sem/v4_1/pg036_sem.pdf.

[18] Bates, T., and Bridges, C. P., "Single Event Mitigation for Xilinx 7-Series FPGAs," *2018 IEEE Aerospace Conference*, IEEE, New York, 2018, pp. 1–12.
https://doi.org/10.1109/AERO.2018.8396520

[19] 7 Series FPGAs Overview (v2.6), Xilinx, 2012, https://www.xilinx.com/support/documentation/ip_documentation/sem/v4_1/pg036_sem.pdf.

[20] Using a Microprocessor to Configure 7 Series FPGAs via Slave Serial or Slave SelectMAP Mode, Xilinx, 2012, https://www.xilinx.com/support/documentation/application_notes/xapp583-fpga-configuration.pdf.

[21] Vivado Design Suite TCL Command Reference Guide, Xilinx, 2014, https://www.xilinx.com/support/documentation/application_notes/xapp583-fpga-configuration.pdf.

[22] McMurtrey, D., Morgan, K. S., Pratt, B., and Wirthlin, M. J., "Estimating TMR Reliability on FPGAs Using Markov Models," 2008.
https://doi.org/10.3923/ajsr.2017.128.138

[23] Weber, R. J., "Reconfigurable Hardware Accelerators for High Performance Radiation Tolerant Computers," Ph.D. Thesis, College of Engineering, Montana State Univ., Bozeman, MT, 2014.

[24] Tylka, A. J., Adams, J. H., Boberg, P. R., Brownstein, B., Dietrich, W. F., Flueckiger, E. O., Petersen, E. L., Shea, M. A., Smart, D. F., and Smith, E. C., "CREME96: A Revision of the Cosmic Ray Effects on Micro-Electronics Code," *IEEE Transactions on Nuclear Science*, Vol. 44, No. 6, 1997, pp. 2150–2160.
https://doi.org/10.1109/23.659030

G. P. Brat
Associate Editor