Journal of Aerospace Information Systems
Vol. 11, No. 2, February 2014

Increasing Radiation Tolerance of Field-Programmable-Gate-Array-Based Computers Through Redundancy and Environmental Awareness

Jennifer S. Hane*
SEAKR Engineering, Inc., Centennial, Colorado 80111
Brock J. LaMeres,† Todd Kaiser,‡ and Raymond Weber§
Montana State University, Bozeman, Montana 59718
and
Todd Buerkle¶
Micron Technology, Inc., Boise, Idaho 83707

DOI: 10.2514/1.I010106

Radiation-tolerant computing is of great importance to the aerospace community because future missions demand more computational power. Of special interest to the aerospace community are flight computers implemented on static random-access-memory-based field-programmable gate arrays. Such computer systems allow the in-flight reconfiguration of hardware that enables the practical deployment of truly reconfigurable computers. However, commercial static random-access-memory-based field-programmable gate arrays are uniquely susceptible to ionizing radiation. This paper introduces a computer architecture for static random-access-memory-based field-programmable gate arrays that resists failures caused by ionizing radiation. The approach extends the widely accepted fault mitigation practice of triple modular redundancy and configuration memory scrubbing by adding spare circuitry and environmental awareness through an ionizing radiation sensor. This paper describes the design of the system in addition to a theoretical analysis of its reliability using a Markov model and empirical analysis using a fault injector. A full system prototype is presented that includes a custom radiation sensor and a computer system implemented on a Xilinx Virtex-6 field-programmable gate array. Both the theoretical analysis and laboratory testing show the approach to be significantly more reliable than field-programmable-gate-array-based computers using only triple modular redundancy and scrubbing.

I. Introduction

As COMPUTING systems have grown more powerful, with a concomitant shrinking of their operational circuitry, the possibility of radiation-induced faults in the hardware has become an ever more pressing concern. This is especially true for systems that must operate outside the protection of the Earth's atmosphere and magnetosphere. Spacecraft and planetary rovers must balance their computing performance requirements against the necessity of maintaining reliability in a more intense radiation environment, because radiation-tolerant hardware generally costs more and lags behind the industry standard for performance [1]. This paper presents work intended to help make these tradeoffs more favorable for space system designers by enabling the reliable use of fast, comparatively inexpensive, commercial off-the-shelf (COTS) parts. Reliability is achieved through a combination of redundancy, repair, and environmental awareness.

Of late, much attention has been focused on static random-access-memory (SRAM)-based field-programmable gate arrays (FPGAs) as computing platforms for space vehicles. The reconfigurable nature of these devices essentially allows them to morph into different specialized computing systems over the course of a mission, or serve as universal spares. Thus, they combine the high performance of customized hardware with the flexibility of traditional microprocessors. Since one FPGA can serve its spacecraft in multiple capacities, they have the potential to greatly reduce weight and space requirements for the mission. FPGAs also allow spacecraft designers to upload new configuration data (essentially modifying the hardware) after launch if an error is found or the mission requirements change.

SRAM-based FPGAs can bring many benefits to a space mission, but their use also carries unique challenges. When ionizing radiation strikes the SRAM inside an FPGA, it can change logic states that control the configuration of the circuitry, effectively changing the hardware and creating erroneous outputs. To correct such errors, one must overwrite the faulty configuration memory; simply resetting the device will not return it to normal operation. FPGAs that use a different type of configuration memory can avoid these problems, but they cannot compare to SRAM-based FPGAs in their versatility. Antifuse FPGAs can only be programmed once, which limits their use as reconfigurable computers. FPGAs based on flash memory do not support partial reconfiguration [1], which severely limits the versatility of fault mitigation techniques and the modularity of the reconfigurable computing approach.

To improve the reliability of an SRAM-based FPGA without building the entire system from slower radiation-tolerant hardware, one must make use of an architecture that employs techniques based on redundancy and/or repair to avoid errors. One such technique is triple modular redundancy (TMR). TMR triplicates the computational hardware and adds circuitry that determines the final output by majority vote. If any one of the three computational modules experiences a fault, the two good modules will overrule it. Initially, TMR is more reliable than a simplex (single module)

Received 22 December 2012; accepted for publication 24 April 2013; published online 10 February 2014.

*Hardware Design Engineer, Radiation Hardened Electronics Group, 6221 S. Racine Circle.

†Associate Professor, Electrical and Computer Engineering Department, 533 Cobleigh Hall (Corresponding Author).

‡Associate Professor, Electrical and Computer Engineering Department, 531 Cobleigh Hall.

§Ph.D. Student, Electrical and Computer Engineering Department, 542 Cobleigh Hall.

¶Product Engineer, DRAM Test Group, 8000 S. Federal Way.

system, but the probability of faults in two or more modules increases with time due to the probability of a strike in the increased circuit area. This eventually reduces the reliability of TMR below that of a simplex system. For this reason, TMR alone is not suitable for long missions [2]. A repair technique called scrubbing can be combined with TMR for better results [3]. Scrubbing is the process of continually overwriting the contents of the configuration memory with known good data (the "golden copy"), which are read from a nonvolatile storage device. This prevents the accumulation of errors that would otherwise eventually cause TMR to fail. However, since the process of reconfiguring the entire FPGA is relatively slow, TMR is still necessary to detect errors instantaneously and prevent them from propagating to the output. Scrubbing and TMR are therefore complementary approaches to fault tolerance.

Our work enhances the basic TMR-plus-scrubbing architecture by adding spare units (also known as tiles) to the system. If one of the active tiles fails, a spare can be brought online to replace it quickly, ensuring that computation continues while partial reconfiguration is used to repair the tile in the background. Our architecture also exploits knowledge of the radiation environment, gained from a specially designed radiation sensor. The sensor is designed to detect the spatial location of ionizing radiation and is mounted over the FPGA die. The sensor detects radiation passing through it and relays the coordinates of the radiation strike to the FPGA. The fault-tolerant architecture uses information obtained from the sensor to prioritize irradiated regions of the FPGA for scrubbing. This ability results in faults being corrected in a shorter amount of time, leading to greater system reliability.

The remainder of this paper is organized as follows. Section II provides additional background and summarizes previous work in this field. Section III describes the design of our system, and Sec. IV gives the details of our prototype implementation. Section V presents the derivation of fault and repair rates and the reliability analysis performed both theoretically and empirically. Section VI concludes the paper.

II. Background and Related Work

Ionizing radiation can affect electronics in a variety of ways [4]. Total ionizing dose (TID) refers to the cumulative permanent damage to an electronic device by lower energy ionizing radiation. It takes the form of charge carriers that are produced by a radiation strike and subsequently become trapped in the device's insulators, where they alter the electrical characteristics of the integrated circuits (ICs). TID causes a device to degrade slowly and inevitably over time; for this reason, space hardware is rated for the amount of TID it can withstand, and it is simply replaced after the specified dose has been exceeded.

Single event effects (SEE) make up the other major category of negative effects. Unlike TID, these faults do not permanently damage the device, but they may cause undesirable outputs while they are active. These effects result from the production of extra charge carriers when a high energy particle enters or passes through part of the device, ionizing some of the silicon atoms. If enough charge is concentrated in an area to reverse the state of a digital logic line in the system, the event is called a single event transient (SET). A SET is likely to result in a brief "glitch" before the excess charge dissipates, unless the new state of the line is captured and retained by a storage device (e.g., a flip flop). An SET that is captured in this way is identified as a single event upset (SEU), and it can have a more enduring effect on the output of the system. However, a SEU can generally be corrected with a quick system reset that restores all flip flops to a known state. SEUs that cannot be dislodged by a simple reset are known as single event functional interrupts (SEFIs). In an FPGA, any state change in the configuration SRAM qualifies as a SEFI, because a reset is not sufficient to correct it. Our work focuses on a logical fault mitigation strategy for SEEs.

A combination of TMR and scrubbing is commonly used to avoid the erroneous outputs due to SEUs and SEFIs in FPGAs used as space hardware. Xilinx, one of the major FPGA manufacturers, officially recommends these techniques** and provides tools and reference designs to aid implementation (for an example, see [5]). For additional examples of the current state of the art in TMR-plus-scrubbing designs, refer to the systems associated with the Cibola satellite project [6], the SpaceCube prototype [7], University of Tsukuba research [8], the Science Technology Satellite-3 project [9], and the ATLAS monitored drift tube computing systems [10]. Fault-tolerant multiprocessor systems that employ some form of redundancy but lack the high flexibility of an FPGA are also of interest; refer to [11,12] for examples.

Previous work at Montana State University has also employed TMR and scrubbing techniques [13]. However, the work presented in this paper goes beyond our previous efforts by including spare tiles and an environmental awareness component in the form of the radiation sensor. This sensor grants the scrubber awareness of radiation passing through the FPGA circuitry, allowing it to move to areas with potential damage and "clean" them quickly. This approach is a novel contribution to the field of radiation-tolerant computing using FPGAs. Standard scrubbers move through an FPGA's configuration memory sequentially, correcting errors as they find them, so the amount of time a SEFI remains in the memory depends on the relative location of the scrubber when the fault occurs. The ability to locate faults as they occur removes this disadvantage. It also reduces the average time needed to bring a spare tile online after one of the active tiles experiences a fault, since the system knows the location of all faulted spares and does not have to test multiple tiles before finding a working tile to activate.

III. Fault-Tolerant Computing: Our Approach

A. General Description and Justification

The fault-tolerance techniques used in our research include redundancy, repair, and environmental awareness. Each is a valuable component of the system that complements and covers the weaknesses of the others. Figure 1 shows the block diagram of a 16-tile version of our system.

The method of redundancy used is TMR with majority voting. TMR is well known as a way of avoiding failures in critical systems, and it is common in designs that must endure harsh extraterrestrial radiation environments [14]. Compared to other options, it is a relatively high-overhead technique, since it requires more than three times the hardware of an equivalent unprotected system. However, it often provides superior reliability as compared to error correction codes, self-checking pairs, and other methods of fault tolerance [15], so it remains an attractive choice. One caveat is that TMR loses its reliability advantage if the mission duration becomes too long. Despite being able to withstand a single fault, the, the tripled chip area necessary to implement TMR actually allows the hardware to capture more radiation strikes. This property can make TMR less reliable than simplex after a long enough operation time, as the probability of multiple module failures increases [10]. Adding additional spare modules that may be traded for damaged members of the active triad increases the number of survivable faults. Spares can increase the allowable mission duration substantially.

Scrubbing complements TMR by preventing the error accumulation that leads to the failure of two modules at once. Repairing a module in the TMR triad as soon as possible after damage can allow the system to sustain many faults before a complete failure occurs. The speed of the scrubber is important to the success of this approach; the faster a module can be repaired, the lower the probability of a second module becoming damaged before the first one is made operable again. A scrubber remains beneficial in a TMR system with extra spares, since it plays a vital role in keeping the spares usable. Without a scrubber, faults in the configuration SRAM of dormant spares would go undetected until they were brought online as replacements and discovered to be faulty.

**Data available online at http://www.xilinx.com/esp/mil_aero/collateral/presentations/radiation_effects.pdf [retrieved 2013].

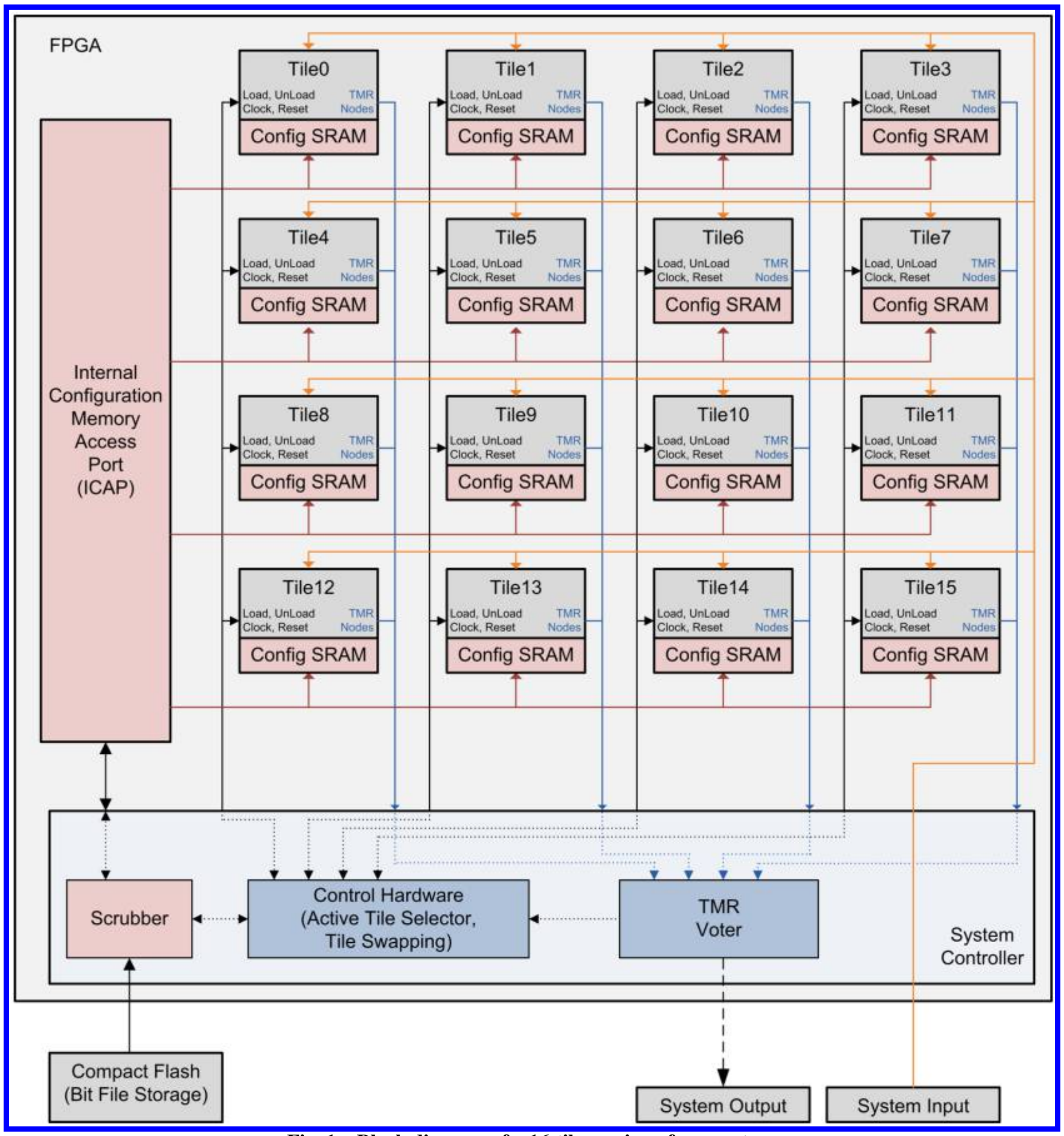


Fig. 1 Block diagram of a 16-tile version of our system.

The previous discussion of the advantages of scrubbing should not lead one to consider it advisable as the sole method of fault tolerance in a system. A scrubber is useless for protecting the system from transient faults that only cause brief glitches on the output rather than affecting the memory. It is also a relatively slow process; when a SEFI occurs in a FPGA's configuration memory, it may affect the output before the scrubber has time to correct it [16]. Block RAM (BRAM) and other types of user memory have ever-changing contents that cannot be compared to a golden copy, so scrubbing them is impossible unless error detection and correction codes are employed. Putting TMR or another form of redundancy in the system, in addition to the scrubber, helps to solve both of these problems. Redundancy also mitigates the effects of TID (something scrubbing alone cannot do). A device affected by high TID levels may not fail all at once, and the ability to replace a computational module with a spare in another part of the FPGA helps one avoid localized permanent errors due to TID.

The final ingredient of our fault-tolerant strategy is environmental awareness, which is provided by a custom-made radiation sensor [17]. As mentioned previously, the speed with which the scrubber can find and remove faults is important to the overall dependability of the system. The sensor alerts the system when radiation strikes the FPGA and provides it with the spatial coordinates of the strike, allowing it to pinpoint the location of faults almost as soon as they happen, even in dormant spares. This feature reduces the latency between the occurrence of a SEFI and its removal by the scrubber.

Thus, our system uses a combination of fault-tolerance strategies that makes the system robust against all three of the major radiation fault types that afflict FPGAs: SEUs, SEFIs, and localized regions of the FPGA damaged by TID. Like all TMR systems, it is still vulnerable to more drastic types of faults, such as multiple bit upsets (MBUs), which could damage two active tiles at once and render the output of the voter erroneous. Such a fault would require a reset or full reconfiguration of the system.

An SRAM-based FPGA was chosen as the platform for our experimental fault-tolerant system because of its high level of adaptability. FPGAs free space systems designers from the requirement of including enough spares to cover the worst-case scenarios for every system. Because it is reconfigurable, an SRAM-based FPGA or a portion thereof can function as a redundant spare for multiple customized hardware units, and it can change its function over the course of the mission. This gives the FPGA an edge over processor arrays (which are only "reconfigurable" in the sense that the interconnect between processors can be changed), such as those in [11,12]. FPGAs can even alter the type of fault tolerance they use to satisfy the needs of the moment. Although TMR is a high-overhead strategy, an FPGA would not need to use it all the time: for example, while passing through regions of space in which radiation levels are low, the FPGA could downgrade its fault tolerance and become a duplex or simplex system, freeing up resources for other activities or reducing power consumption.

In any TMR-plus-scrubbing system, protection of the voter, the scrubber, and other control circuitry is a crucial consideration. This can be achieved by housing the control circuitry in a more radiation-tolerant part. The part is likely to be slower, smaller, or more costly (perhaps all three) than the state-of-the-art COTS FPGA used for computation. Since the control circuitry does not need to perform demanding computations and can be made reasonably simple, regardless of the system's task, this approach still has advantages over placing all computational units in slower radiation-tolerant parts. This method of protecting the control circuitry is followed by the SpaceCube prototype [7]. Since our current work is a proof of concept, we prototyped all control circuitry on the same FPGA as the computational tiles; however, we assume that the control circuitry is secure when conducting fault-injection tests and preparing theoretical models of the system.

B. Redundant Many-Tile Architecture

The architecture for our system consists of 16 computational tiles, with a soft processor in each tile. Each of the 16 tiles in the system is located within a partially reconfigurable region (PRR) inside the FPGA. These regions can be reconfigured separately from the rest of the FPGA, so that if one tile suffers an error, its configuration data can be rewritten while the remainder of the FPGA continues to operate. The remainder of the FPGA, outside the PRRs, is known as the static region, and it holds the control electronics. These include the TMR voter, the tile-swapping state machine, the sensor interface, the communications interfaces, and a control processor that handles scrubbing. The size and shape of the PRRs are defined in the initial FPGA floor plan and cannot be altered during operation. Figure 2 shows the FPGA floor plan of our 16-tile system, with each rectangle representing a PRR layout inside the FPGA containing a full soft processor.

Three of the 16 processor tiles are active at any given time, while the others are held in reset to conserve power. The active tiles are routed through a multiplexer and connected to the TMR unit, which performs a majority vote on all their outputs and checks for disagreements. If one of the three tiles is found to be in error, it is taken offline and marked "damaged," and a spare tile is activated to take its place. A state machine orchestrates the process of swapping the defective module for a good spare and resetting/synchronizing all of the active tiles. For testing purposes, an SEU may be simulated on a tile by pulling the tile's output low, while a SEFI may be simulated by reconfiguring the tile with "false" circuitry that is not a processor, thereby corrupting the configuration memory and producing incorrect outputs.

A scrubber is also included, and it may operate in either blind or readback mode. The "blind" variant simply overwrites one whole tile at a time, whether it is damaged or not. In readback mode, the scrubber examines each frame of configuration data in the tile and compares it with the data in the tile's bit file, checking for errors. If it finds any, it proceeds to overwrite the tile, and it reports the number of corrupted data words to the user. By default, the scrubber cleans tiles sequentially and in numerical order (active tiles and BRAMs are not scrubbed, as scrubbing could disrupt their

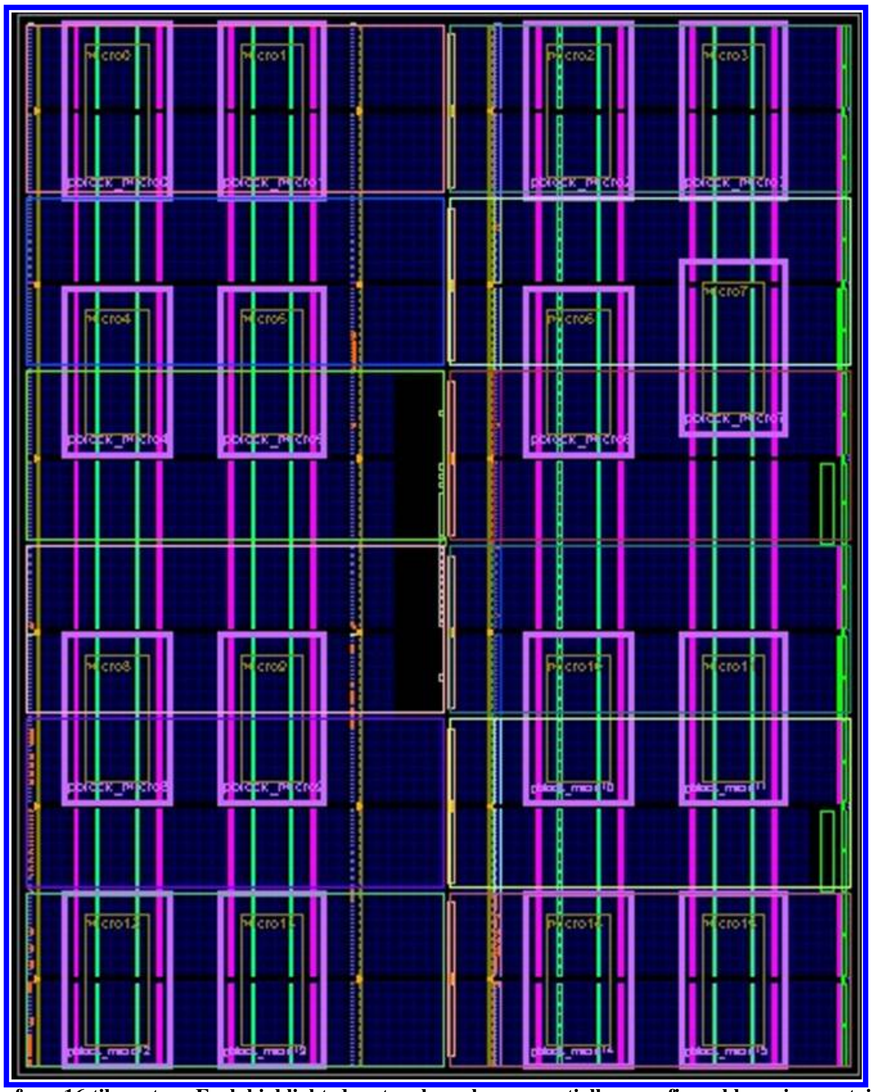


Fig. 2 FPGA floor plan of our 16-tile system. Each highlighted rectangle encloses a partially reconfigurable region containing a full soft processor.

operations and the TMR voter will catch any important errors). However, if the voter declares a tile damaged or the sensor reports radiation strikes within the tile's area, the scrubber will jump to that tile as soon as it finishes with its current target.

C. Environmental Awareness

The sensor itself is an intrinsic silicon device that is doped to create a wide-area PN junction (front-side doped with P-type and the back side doped with N-type dopants). When ionizing radiation passes through the sensor, producing a trail of charge carriers, the electric field created by the junction separates the electrons from the holes. The electrons are forced to the back of the sensor and the holes move toward the front. The charge carriers are then collected on opposite sides of the sensor, resulting in brief output current pulses. Refer to Fig. 3. The basic principle of operation is very similar to that of a solar cell, with one primary difference: the electrodes on each side are divided into distinct channels, each one serving to collect charge from one narrow strip of the sensor. The strips on opposite sides of the sensor are orthogonal to each other, creating a grid that can report the location of radiation strikes. Each high-energy particle that passes through the sensor will produce a current pulse on one front-side (X) channel and one rear-side (Y) channel. The intersection point of these two channels defines one sensor pixel and provides the location information needed by the FPGA scrubber. Refer to Fig. 4. The sensor is designed to detect radiation levels that commonly cause SEEs in modern CMOS circuitry. For low energy levels, the radiation will not pass through the sensor and not create signals on the electrodes. For higher energy particles, they will pass through the sensor, triggering a detection of a strike as they then strike the FPGA die.

The design of the sensor was optimized to produce the largest amount of current possible in response to radiation levels that cause SEEs in modern ICs. To estimate the current that results from a high-energy particle strike, one must first calculate the built-in voltage and depletion region width of the sensor's vertical PN junction as follows:

$$V_{bi} = \frac{kT}{q}\ln\left(\frac{N_J N_D}{n_i}\right) \tag{1}$$

$$w = \sqrt{\frac{2\varepsilon_r\varepsilon_o(N_J + N_D)}{qN_J N_D}(V_{bi} - V_{\mathrm{BIAS}})} \tag{2}$$

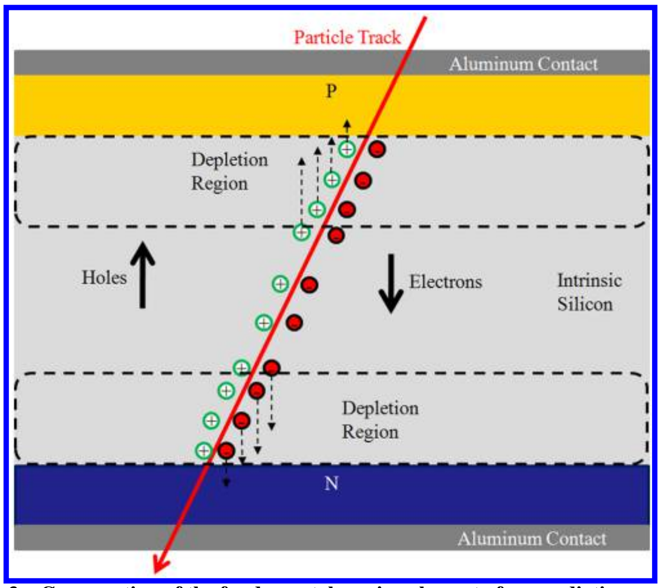


Fig. 3 Cross section of the fundamental sensing element of our radiation sensor.

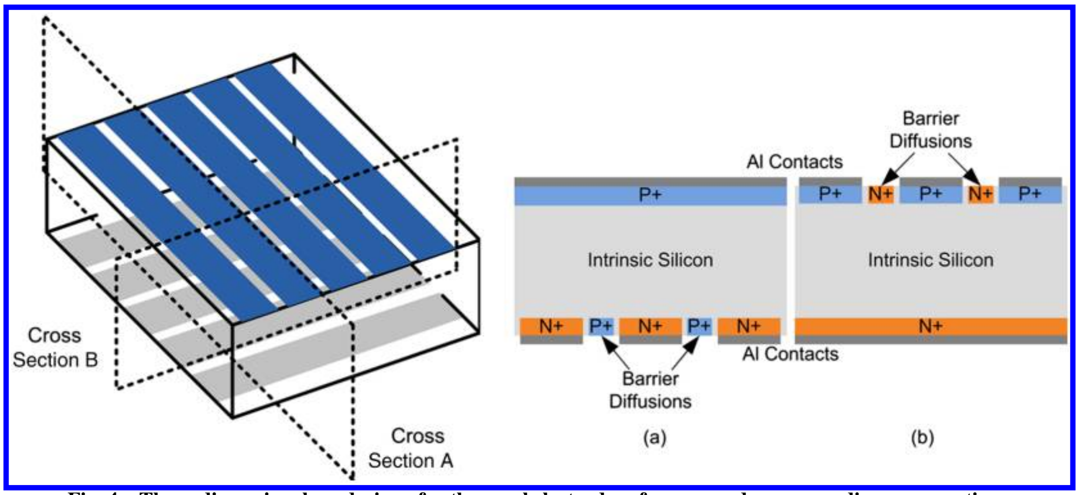


Fig. 4 Three-dimensional rendering of orthogonal electrodes of sensor and corresponding cross sections.

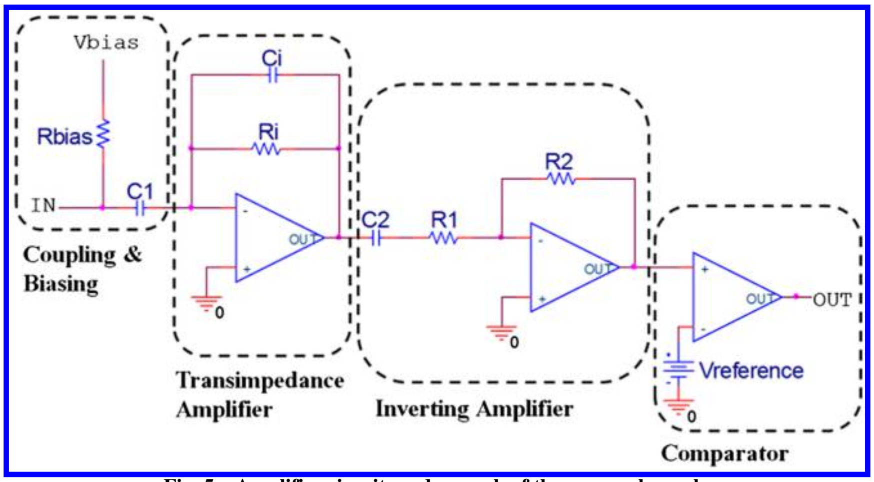


Fig. 5 Amplifier circuit used on each of the sensor channels.

where k is Boltzmann's constant, T is the temperature in kelvins, q is the charge on an electron, N_D is the background donor concentration, n_i= 1.45e^{10} cm^3 (intrinsic concentration of the silicon wafer), $\varepsilon_r = 11.9$, and ε_o is the permittivity of free space The value for N_J is either the $P+$ or $N+$ doping concentration that was previously determined. Given the values for V_{bi} and w and the linear energy transfer (LET) of a particle, one may calculate the electric field and the charge produced [Eqs. (3) and (4)]. The density of the target material is ρ, q is the charge on an electron, and the factor 3.3 is the electron hole pair generation rate in silicon in effective hole pair per electron volt. From this, the velocity of the charge carriers v can be found using Eq. 5, where μ is the mobility $\mathbf{v}(x) = \mu \cdot \mathbf{E}$ of the electrons or the holes. This in turn yields the transient time τ where l is the collection path length [Eq. (6)]. Finally, the current pulse is determined using Eq. (7):

$$\mathbf{E} = -\nabla V \approx \frac{V_{bi}}{w} \tag{3}$$

$$Q = \frac{LET \cdot \rho \cdot q}{3.3} \tag{4}$$

$$\mathbf{v}(x) = \mu \cdot \mathbf{E} \tag{5}$$

$$\tau = \frac{l}{\|\mathbf{v}(x)\|} \tag{6}$$

$$I = \frac{Q}{\tau} \tag{7}$$

For typical radiation levels that cause SEEs, the sensor outputs signals in the 10–100 μA range that last hundreds of nanoseconds depending on the LET of the particle. These pulses are too small and short to be read directly by the FPGA, so the pulses must be amplified and conditioned to a suitable digital level. A custom amplifier system is created to amplify, stretch, and digitize the pulses coming out of the sensor. An amplifier chain is used for each of the electrodes coming out of the sensor. Different gains are selected for the top-side (electrons) and bottom-side (holes) signals.

The amplifier circuit begins with an ac coupling capacitor to remove any dc offset in the signals coming from the sensor. A bias resistor is included in order to provide additional energy to the sensor to expand the depletion region of the sensing element for increased sensitivity. The second stage is a transimpedance amplifier that amplifies and stretches the current pulse. This signal is ac coupled into an inverting amplifier that applies a fixed gain to the signal. The ac coupling capacitor before this stage eliminates any effect of dc offset in the transimpedance amplifier. Finally, the signal is run into a comparator that produces a final digital signal. Figure 5 shows the schematic for the amplifier chain.

Each channel of the sensor is brought into a high-speed sampler in order to capture the pulse to determine the location of the radiation strike. Since the pulse may be shorter than the system clock of the computer system could capture, a serial-to-parallel architecture is used where the serial shift register is run off of a clock that is faster than the system clock. In our approach, we used a sampling clock that is eight times faster than the system clock. A sensor input is sampled into a shift register of eight D flip flops. After eight samples, the system clock latches the output of all eight flip flops into an 8-bit register. Each of the bits within the parallel register is fed through an OR gate. If any of the bits within the parallel register are a 1, then a radiation strike occurred within the last period of the system clock.

To collect historical information, the output of the OR gate is used as a synchronous enable line to a counter. This counter will increment off of the system clock if a strike occurred in the past period. To determine the location of an XY intersection, the output of the OR gate on an X channel is AND‘d with the output of an OR gate on a Y channel. An AND gate is placed on the intersection of each X channel with every other Y channel. This gives a total of (X)·(Y) signals representing intersections, or what we refer to as pixels. A final counter is fed by the output of the AND gate in order to collect historical information about radiation strikes that passes all the way through the sensor. Figure 6 shows the logic diagram of the high-speed sampler.

D. Sensor and Field-Programmable-Gate-Array Assembly

There are a variety of techniques that can be used to integrate the sensor with the FPGA die. The first is to directly mount the sensor die to the FPGA die and then use wire bond interconnect to interface between the components. This has the advantage that the sensor is spatially close to the

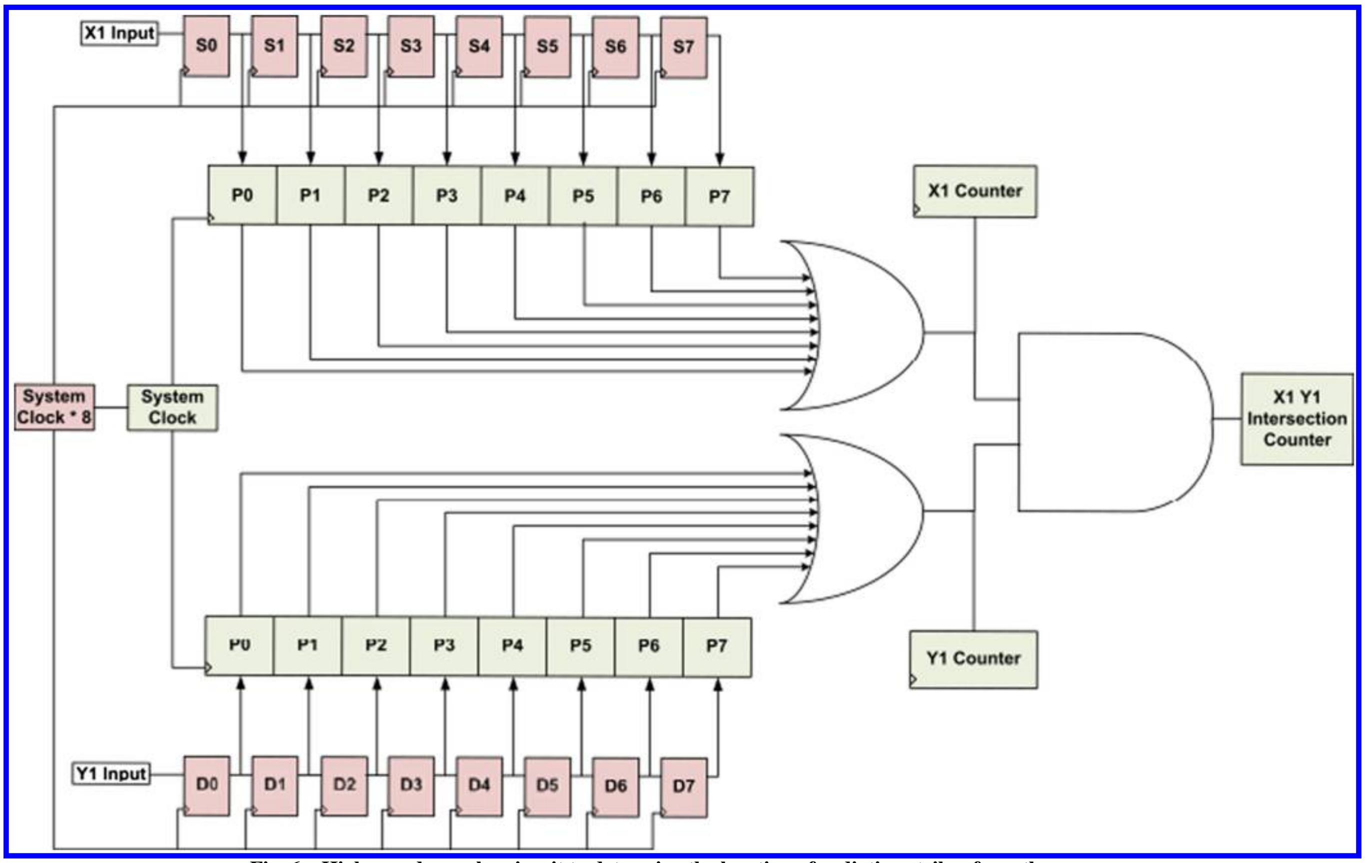


Fig. 6 High-speed sampler circuit to determine the location of radiation strikes from the sensor.

FPGA and will more accurately predict strike locations for particles that are not completely orthogonal to the FPGA die surface. Another approach is to have the sensor on its own printed circuit board (PCB) and mounted directly above the fully packaged FPGA, and use connectors to interface the signals. This has the advantage that a COTS FPGA can be used without the need for altering the packaging process. This approach is shown in Fig. 7.

In both packaging approaches, multiple sensors can be used in a stacked confirmation to provide trajectory information about incoming radiation. The direct die attach approach will have less accuracy for trajectory calculation due to the close proximity of the sensors. This paper presents the use of a single sensor with the FPGA, but two sensors could easily be accommodated.

IV. Design and Prototyping of Our System

A. Many-Tile Computer System Prototype

A 16-tile version of our system was implemented on the Xilinx ML605 demonstration board. This platform contains a Xilinx Virtex-6 XC6VLX240T-1FFG1156 FPGA. This platform also contains several convenient peripherals that make laboratory testing easier. The Virtex-6 FPGA was the largest FPGA that supported partial reconfiguration when we began work on this project, and it was therefore a natural choice of target device. In our design, we implemented the control circuitry and TMR voter on the same FPGA fabric as the 16-tile computing fabric. This was done for ease of implementation. In a final system, all control circuitry would reside outside of the COTS FPGA fabric on a device that is more resilient to SEEs: typically an older-generation FPGA with larger feature sizes and, in turn, lower performance. The partitioning of an ideal system is shown in Fig. 1, and a picture of our FPGA platform is shown in Fig. 8.

Each of our 16 tiles contains a full Xilinx MicroBlaze soft processor. For demonstration purposes, each tile processor was programmed to compute digits of π. The control portion of the system also uses a single MicroBlaze processor. The control MicroBlaze uses the Virtex-6's internal configuration access port (ICAP) to scrub the 16 tiles of compute section. Each PRR has its own partial bit file; passing the data in that file to the ICAP results in a clean overwrite of the configuration memory for that PRR, while the rest of the memory remains untouched. Mask files, generated by the Xilinx tools along with the partial bit files, are used to determine which frames in the memory correspond to BRAMs so that the

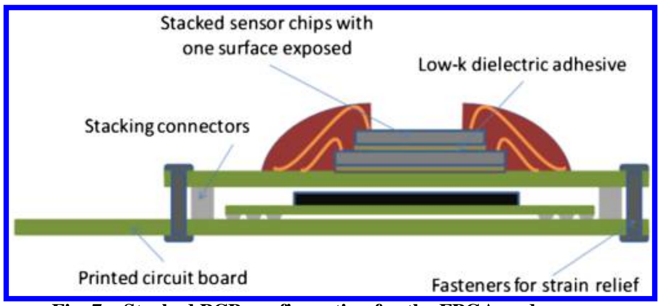

Stacked sensor chips with one surface exposed
Low-k dielectric adhesive
Stacking connectors
Printed circuit board
Fasteners for strain relief

Fig. 7 Stacked PCB configuration for the FPGA and sensors.

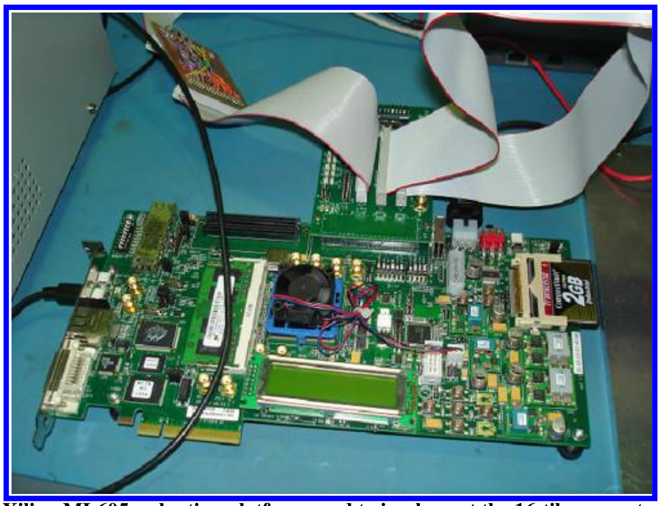


Fig. 8 Xilinx ML605evaluation platform used to implement the 16-tile computer system.

BRAMs can be left unscrubbed. The bit and mask files are stored on a compact flash card, which the master MicroBlaze accesses through the SystemACE interface included on the ML605. The system reads in the digitized radiation sensor signals on its general I/O pins. The high-speed event detector circuitry also resides inside the control section of the FPGA.

B. Sensor System Prototype

The sensor was fabricated at the Montana Microfabrication Facility at Montana State University. The overall size of the sensor is 20×20 mm, making it large enough to cover the any commercially available FPGA die. The sensor was designed with 16 channels on each side to give a total of 256 sensitive pixels. A 300-μm-thick intrinsic silicon wafer was used as the base material for fabrication. Table 1 gives the calculated and measured parameters of the sensor. Figure 9 shows a picture of both the sensor wafer after fabrication and a final diced and packaged sensor. The sensor system was designed for modularity for ease of testing. The sensor was packaged on its own PCB, while the 32 channels of amplifier electronics were implemented on a separate 100×100 mm PCB into which the sensor was plugged. Figure 10 shows the amplifier PCB.

C. Prototype Assembly

Our entire system is designed for a stacked configuration that will fit within a 100 mm^3 volume. This is to adhere to the CubeSat standard for future flight testing. The sensor board is designed to be stacked on top of a custom FPGA board and a power board. The system can also accommodate two amplifier boards (each with a sensor) in order to perform trajectory calculations of incoming radiation. For our prototype, the amplifier and power boards were implemented in the 100×100 mm form factor in addition to a signal breakout board that interfaced the radiation signals to the ML605 FPGA platform (Fig. 11).

D. Laboratory Setup

To monitor the status of the system and inject faults for testing purposes, our system includes a parallel-to-universal-serial-bus (USB) conversion board containing a Morphic IC-II to allow communication via USB with a graphical user interface (GUI) that runs on a supporting computer. The GUI provides a visual representation of the internal status of the FPGA, displaying which tiles are active, dormant, and damaged;

Table 1 Sensor parameters

Symbol	Quantity	Value
R_{P+}	$P+$ sheet resistivity	46.07 Ω/square
R_{N+}	$N+$ sheet resistivity	11.46 Ω/square
X_{P+}	$P+$ doping junction depth	$1.691 \cdot 10^{-4}$ cm
X_{N+}	$N+$ doping junction depth	$1.335 \cdot 10^{-4}$ cm
ρ_{P+}	$P+$ resistivity	$7.79 \cdot 10^{-3}\ \Omega \cdot$ cm
ρ_{N+}	$N+$ resistivity	$1.53 \cdot 10^{-3}\ \Omega \cdot$ cm
N_{P+}	$P+$ doping concentration	$1.15 \cdot 10^{19}$ cm^{-3}
N_{N+}	$N+$ doping concentration	$4.53 \cdot 10^{19}$ cm^{-3}
V_{bi-P+}	Front-side built in voltage	0.619 V
V_{bi-N+}	Back-side built in voltage	0.655 V
w_{P+}	Front-side depletion width	61.57 μm
w_{N+}	Back-side depletion width	63.31 μm
μ_h	Hole mobility	470.46 cm^2/V-s
μ_e	Electron mobility	1413.87 cm^2/V-s
E_{P+}	Front-side electric field	100.54 V/cm
E_{N+}	Back-side electric field	103.46 V/cm
v_h	Hole velocity	47,300 cm/s
v_e	Electron velocity	146,279 cm/s

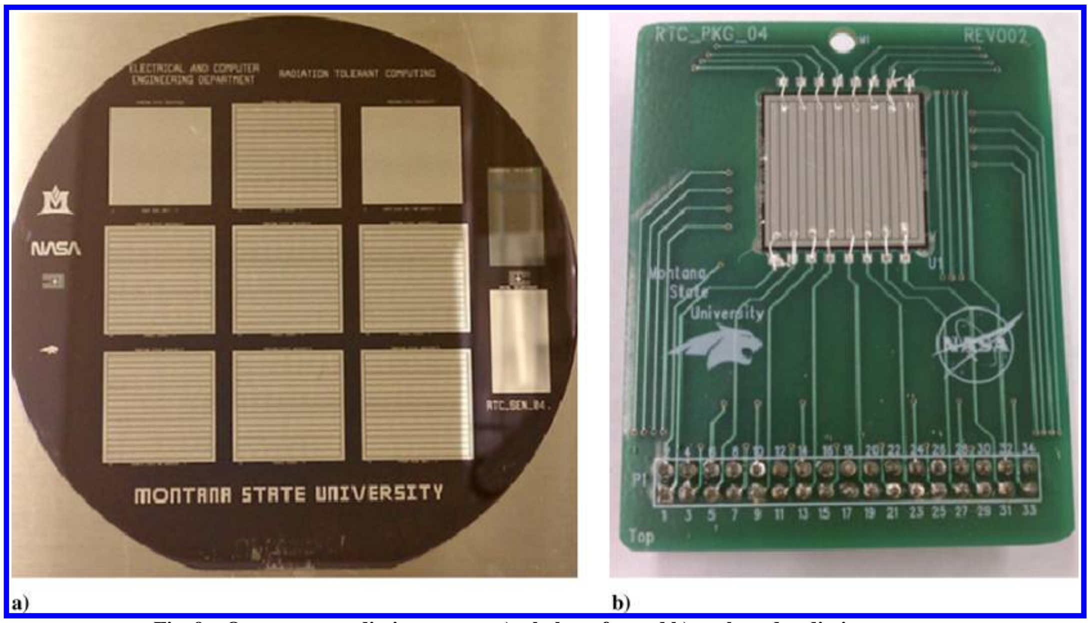


Fig. 9 Our custom radiation sensor: a) whole wafer and b) packaged radiation sensor.

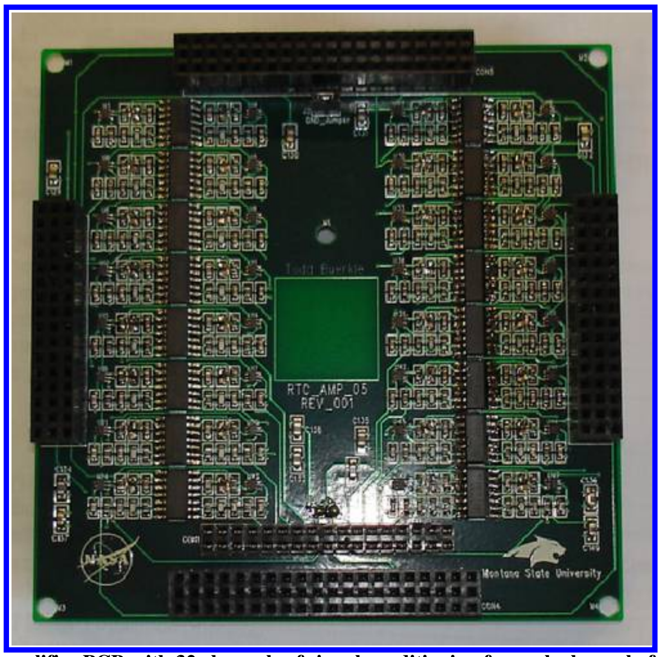


Fig. 10 Amplifier PCB with 32 channels of signal conditioning for each channel of the sensor.

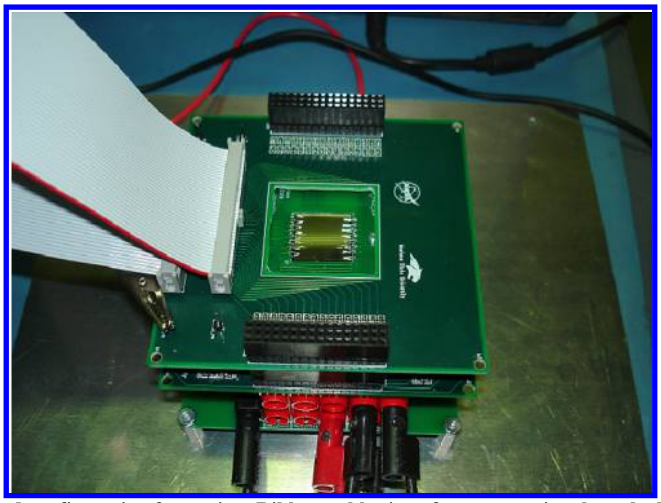
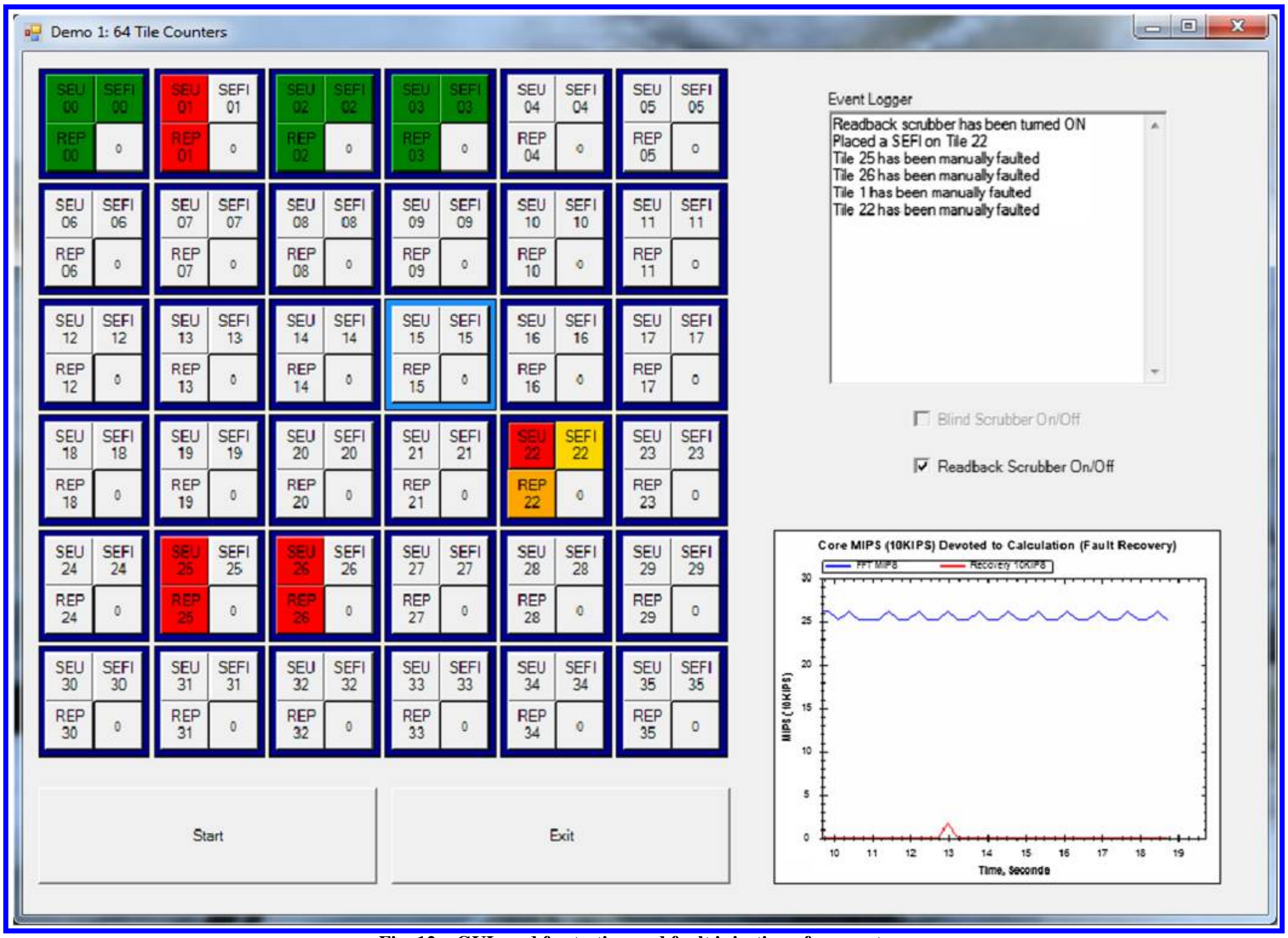


Fig. 11 Our system in its stacked configuration for testing. Ribbon cables interface sensor signals to the FPGA ML605 evaluation board.



Readback scrubber has been turned ON
Placed a SEFI on Tile 22
Tile 25 has been manually faulted
Tile 26 has been manually faulted
Tile 1 has been manually faulted
Tile 22 has been manually faulted

☐ Blind Scrubber On/Off
☑ Readback Scrubber On/Off

Fig. 12 GUI used for testing and fault injection of our system.

the current location of the scrubber; etc. It also allows the user to send commands to the system in order to inject simulated faults, turn the scrubber on and off, or manually repair faults. A screenshot of the GUI is shown in Fig 12.

V. Reliability Analysis

To gauge the reliability of our system, we performed both theoretical and empirical analyses to determine the mean time before failure (MTBF) when in a representative environment. The MTBF is compared to a system that contains only TMR plus scrubbing in order to see the improvement in reliability by adding spares [18] and environmental awareness.

A. Derivation of Fault Rates

We obtained realistic fault rates to use in our models by running calculations using Vanderbilt University's online CREME96 tool [19]. Given inputs that define an orbit, space weather conditions, and the fault cross section of a device, CREME96 will predict how many faults that device would experience each second, on average, if it resided onboard a satellite in that orbit. To the best of our knowledge, there is no publically available data on transient radiation faults that could be used as an input to the CREME96 tool for the Virtex-6. However, Xilinx claims that the Virtex-6 is less vulnerable to transient radiation faults than its predecessor, the Virtex-5 [20]. Therefore, data obtained from radiation tests on the Virtex-5 in [21,22] are conservative approximations to what might be expected for the Virtex-6. We generated fault rates for four orbits under various space weather conditions. In each case, the fault rate was averaged over the entire orbit. The four orbits used include the International Space Station orbit (a low Earth orbit), the Molniya 1-80 (a highly elliptical orbit), the Satcom 5 (a geosynchronous orbit), and the EXP-1 Prime (the orbit of Montana State University's student-built CubeSat). These will be hereinafter referred to as ISS, HEO, GEO, and E1P, respectively. For our purposes, "standard" space weather conditions occur at solar maximum with peak protons and a stormy magnetosphere, without a solar flare event. "Worst week" conditions are based on a seven-day running average taken during the most intense part of a flare during solar maximum. "Peak five minutes" conditions are similarly defined for the worst 5 min of such a storm.

The fault rates generated by CREME96 are given in faults per device per second. However, not every fault in the device will result in an erroneous output. The actual number of critical faults per second depends heavily on the nature of the design implemented on the FPGA; some designs have more "sensitive bits," or bits that cause an output error if flipped, than others. Extensive time-consuming testing would be required to reveal what proportion of the bits in our particular design are sensitive. Instead, a worst-case estimate of 35% sensitive bits was used, based on measured sensitive bit densities for a variety of circuit designs on a Virtex-4 FPGA [23]. Therefore, in our model, the rate at which tile failures occur is 35% of the CREME96 output fault rate. This adjustment for sensitive bits is only applied if the sensor is NOT used, since the sensor has no way of determining whether an incoming particle will hit a sensitive bit, and it will declare any tile that is struck to be damaged (making it unusable by the TMR system until it is repaired). Table 2 shows the fault rates extracted from the CRÈME96 tool. Figure 13 shows the path of each orbit studied and the corresponding fault rate at each point of its path.

Table 2 Orbital fault rates from CREME9 (faults/device/second)

	Average	Worst week	Peak 5 min
ISS	0.0003479	3.544	72.96
HEO	0.08788	120.2	2398
E1P	0.003464	29.93	612.3
GEO	0.002494	149.8	3059

Table 3 Measured scrubbing rates

Parameter	Value
Clock rate	25 MHz
Blind	2.97 s
Readback, undamaged	5.31 s
Readback, damaged	6.35 s

B. Derivation of Repair Rates

Empirical measurements were used to obtain the repair rate of the system. Due to differences in tile location within the FPGA, the time needed to scrub a tile varies, so averages were taken across all 16 tiles. For the system described in this paper, which includes the integrated radiation sensor, we assumed that sensor information would allow the scrubber to jump to faulted tiles immediately after the fault occurred. Therefore, when modeling this system, we set the repair rate equal to the reciprocal of the time needed to scrub one tile (a damaged tile, in the case of the readback scrubber). This repair rate remains constant, regardless of the system's state. Also measured was the time difference between performing readback scrubbing of a tile when the tile is damaged versus undamaged.

The damaged tile scrubbing takes longer than for an undamaged tile because, when reading an undamaged PRR bit file, action is only taken if the compare results in a difference. Table 3 shows the scrubbing times for our system with a 25 MHz system clock.

C. Theoretical Analysis of Mean Time Before Failure

A Markov model was used to theoretically estimate the mean time before failure of the system under various fault rates, while artificial fault injection was used to measure the MTBF experimentally. A Markov model describes a system as a directed graph in which each node is a state and each edge represents a transition between states. We modeled our design by creating one state to correspond to each possible number of

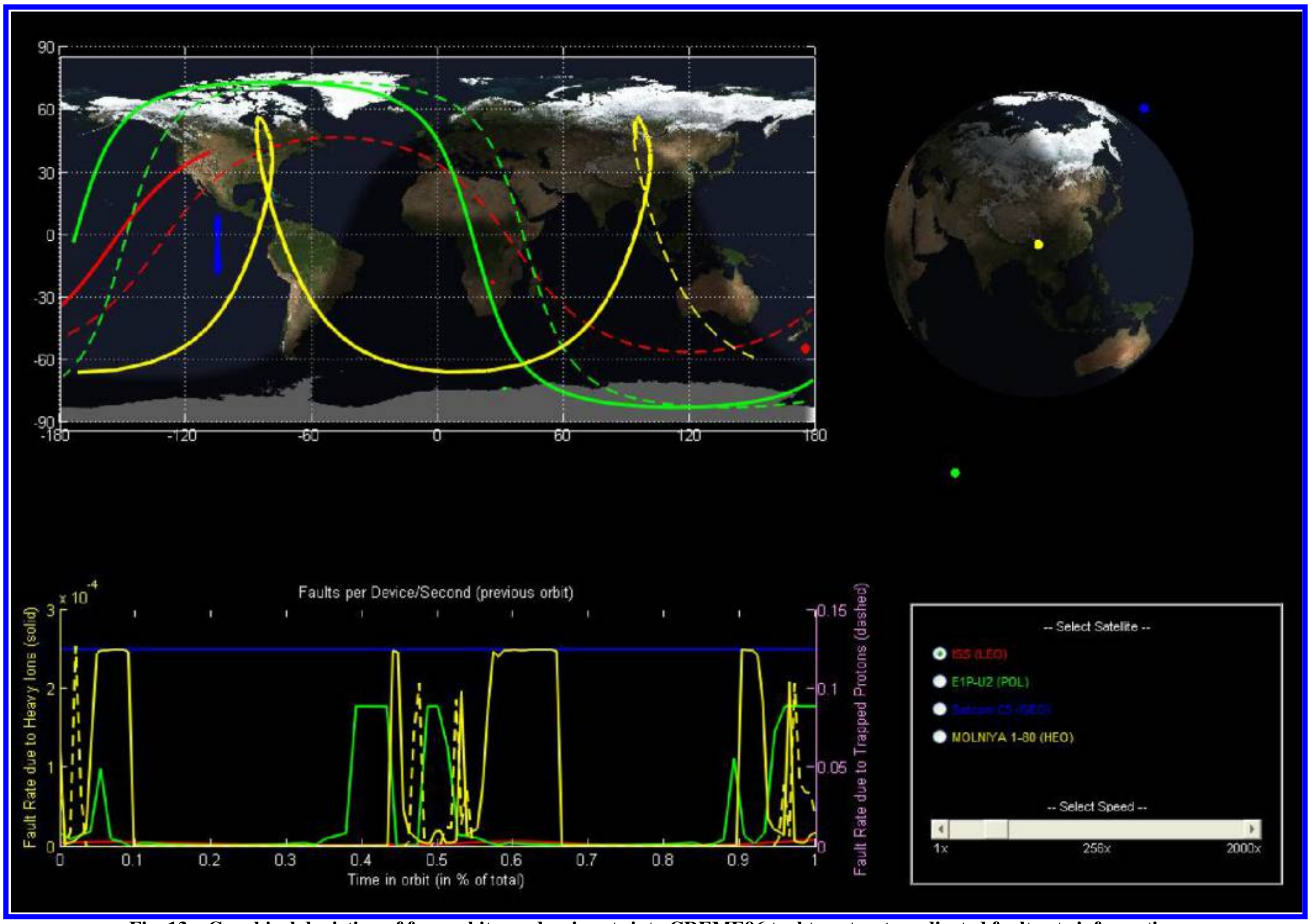

Faults per Device/Second (previous orbit)
Fault Rate due to Heavy Ions (solid)
Fault Rate due to Trapped Protons (dashed)
Time in orbit (in % of total)
-- Select Satellite --
ISS (LEO)
E1P-U2 (POL)
Satcom C5 (GEO)
MOLNIYA 1-80 (HEO)
-- Select Speed --
1x
256x
2000x

Fig. 13 Graphical depiction of four orbits used as inputs into CREME96 tool to extract predicated fault rate information.

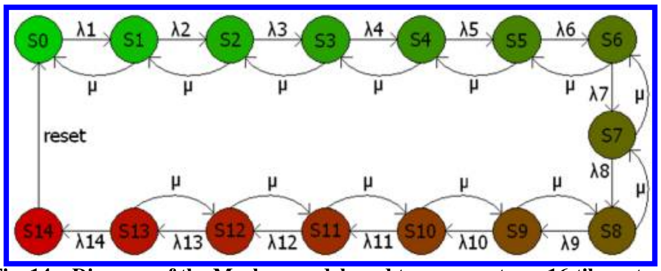


Fig. 14 Diagram of the Markov model used to represent our 16-tile system.

undamaged spares. The system experiences a state transition whenever a spare suffers a fault or is repaired by the scrubber. The graph corresponds to a system of differential equations, which may be solved to obtain the probability of the system reaching its failed state. We define the failure state as only two good tiles remaining. The MTBF is then easily calculated from the probability of entering the failed state. It was necessary to define two parameters for each state in the module: the rate of transition to a state with one less good tile (the fault rate λ), and the rate of transition to a state with one more good tile (the repair rate μ).

Figure 14 shows the Markov graph for our 16-tile system. In this figure, S0 represents the initial state of the system in which all spares are in usable condition, and S14 represents the failed state, in which only two usable tiles remain. The failure rate λ decreases as the number of damaged spares increases because new radiation strikes become more likely to hit previously damaged tiles. If the scrubber can find damaged tiles near-instantaneously, as we assume for the system that includes the radiation sensor, the rate of scrubbing or repair μ remains constant.

Using the fault rates λ from Table 2 and the repair rates μ from Table 3, the MTBF can be computed using the Markov model. To set a baseline, the MTBF was first calculated for a TMR-plus-scrubbing system. This system represents the accepted approach to SEE fault mitigation in SRAM-based FPGAs. We then add our novel contributions (spares and environmental awareness) and recalculate the MTBF to see if reliability is increased. Table 4 shows the MTBF for the baseline TMR-plus-scrubbing system. In this table, the MTBF is reported for each of the four orbits for which fault rate information was collected. It also reports the MTBF for various weather conditions (average, worst week, or peak 5 min) of each orbit. The MTBF is also reported for both blind and readback (RB) scrubber configurations.

The TMR-plus-scrubbing system is then augmented to include 13 spare tiles, and the MTBF is recomputed. Table 5 gives the new MTBF, and Table 6 reports the percent increase in MTBF compared to the baseline (Table 4). The addition of spares increased MTBF dramatically in all environments. Even during the peak 5 min radiation environment, the system is able to produce an improvement in MTBF of over 1000%.

Table 4 MTBF for baseline TMR-plus-scrubbing system

		Average, s	Worst week, s	Peak 5 min, s
Blind	ISS	1.08E + 08	3.19E + 00	1.07E − 01
	HEO	1.77E + 03	6.43E − 02	3.20E − 03
	E1P	1.09E + 06	2.69E − 01	1.25E − 02
	GEO	2.09E + 08	5.14E − 02	2.50E − 03
RB	ISS	6.00E + 07	2.73E + 00	1.06E − 01
	HEO	1.03E + 03	6.39E − 02	3.20E − 03
	E1P	6.07E + 05	2.63E − 01	1.25E − 02
	GEO	1.17E + 08	5.12E − 02	2.50E − 03

Table 5 MTBF for TMR-plus-scrubbing-plus-spaces system

		Average, s	Worst week, s	Peak 5 min, s
Blind	ISS	3.57E + 43	7.83E + 01	1.25E + 00
	HEO	3.75E + 11	7.41E − 01	3.59E − 02
	E1P	4.46E + 29	3.30E + 00	1.41E − 01
	GEO	3.74E + 45	5.90E − 01	2.81E − 02
RB	ISS	8.26E + 41	5.49E + 01	1.23E + 00
	HEO	2.10E + 10	7.33E − 01	3.59E − 02
	E1P	1.08E + 28	3.16E + 00	1.41E − 01
	GEO	8.63E + 43	5.85E − 01	2.81E − 02

Table 6 Increase in MTBF after addition of spares

		Average, %	Worst week, %	Peak 5 min, %
Blind	ISS	3.31E + 35	2356.07	1067.45
	HEO	2.12E + 08	1051.79	1021.88
	E1P	4.10E + 23	1127.98	1031.20
	GEO	1.78E + 37	1047.86	1024.00
RB	ISS	1.38E + 34	1912.98	1058.51
	HEO	2.05E + 07	1046.32	1021.88
	E1P	1.78E + 22	1103.77	1028.80
	GEO	7.40E + 35	1042.38	1024.00

Table 7 MTBF for TMR-plus-scrubbing-plus-spares-plus-sensor system

		Average, s	Worst week, s	Peak 5 min, s
Blind	ISS	1.31E + 46	3.47E + 01	4.39E − 01
	HEO	2.21E + 13	2.60E − 01	1.26E − 02
	E1P	1.50E + 32	1.17E + 00	4.95E − 02
	GEO	1.37E + 48	2.07E − 01	9.90E − 03
RB	ISS	6.77E + 41	1.60E + 01	4.25E − 01
	HEO	7.82E + 09	2.55E − 01	1.26E − 02
	E1P	8.45E + 27	1.08E + 00	4.93E − 02
	GEO	7.09E + 43	2.04E − 01	9.80E − 03

Table 8 Increase in MTBF after addition of spares plus sensor

		Average, %	Worst week, %	Peak 5 min, %
Blind	ISS	1.21E + 38	989.57	311.04
	HEO	1.24E + 10	304.51	293.75
	E1P	1.38E + 26	336.57	296.00
	GEO	6.55E + 39	302.92	296.00
RB	ISS	1.13E + 34	488.49	301.51
	HEO	7.63E + 06	298.90	293.75
	E1P	1.39E + 22	310.16	294.40
	GEO	6.08E + 35	297.85	292.00

Next, the sensor is added to the system to create a TMR-plus-scrubbing-plus-spares-plus-sensor configuration. Table 7 shows the theoretical MTBF for this system, and Table 8 shows the percent improvement compared to the baseline system (Table 4). The addition of the sensor dramatically increases the MTBF beyond just adding spares for the readback scrubber configuration. This improvement comes from the ability to repair damaged spare tiles before they are brought online. This puts the system into a more reliable state in the model without affecting the operation of the active triad. However, it does not improve the MTBF as much as spares alone for the readback scrubber configuration. An explanation for this result is provided by the fact that the sensor cannot detect whether incoming radiation hits a sensitive or insensitive bit on the FPGA. An insensitive bit is a circuit on the FPGA that can be altered by an SEE but does not actually affect the active circuitry because it is not used. Therefore, the sensor will repair tiles that may not be technically damaged.

A slight modification of the system could solve this problem: rather than having the sensor declare any struck tile to be damaged, it could prioritize them for scrubbing but leave them available for attempted use by the TMR voter. This could increase the time needed for tile swaps slightly, since the system would occasionally swap in a damaged tile and have to perform a second swap to obtain a good one. However, since the tile swap time is substantially less than the scrubbing time, such a change should still improve performance.

D. Analysis of Mean Time Before Failure

A fault injection system was developed to compare the theoretical MTBF rates predicted by the Markov models to empirical measurements. The fault injection system produced random faults in the FPGA at the rates from Table 2. The USB interface was used to send the fault rate information continually to the FPGA. The control circuitry on the FPGA then causes randomly located faults in the 16-tile fabric based on the fault rate information. The fault rate was increased until the system failed. This plot verified that our Markov models were accurate. Figure 15 shows the measured MTBF versus the fault rate.

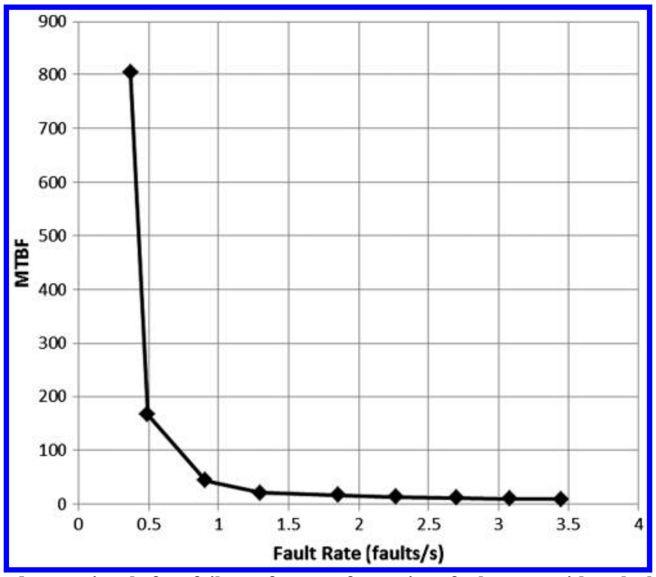


Fig. 15 Measured mean time before failure of system for various fault rates, with a clock rate of 25 MHz.

VI. Conclusions

In this paper, a novel approach to fault tolerance in static random-access-memory (SRAM)-based field-programmable gate arrays (FPGAs) was presented. Triple modular redundancy (TMR) with many additional spares, scrubbing, and environmental awareness were combined to create a versatile and resilient fault-tolerance strategy. The environmental awareness component was provided by a multipixel radiation sensor, which served the system by providing information about the locations of radiation strikes on the chip. Theoretical and experimental tests revealed that the addition of spares and a sensor can greatly increase the robustness of the system compared to the currently used TMR-plus-scrubbing approach. The addition of numerous spares to the basic TMR system produced improvements under all fault conditions, often increasing reliability by many orders of magnitude. The addition of spares plus a sensor further increased reliability when using a readback scrubber. The system presented in this paper aims at increasing the reliability of SRAM-based commercial off-the-shelf FPGAs in space applications in order to enable high-performance reconfigurable computer systems for space applications.

Acknowledgments

This work was supported in part by a grant from the NASA Experimental Program to Stimulate Competitive Research (NNX10AN32A). The authors would like to thank the engineers from the Advanced Avionics Program at the NASA Marshall Space Flight Center for their collaboration on this research.

References

[1] Jacobs, A., George, A. D., and Cieslewski, G., "Reconfigurable Fault Tolerance: A Framework for Environmentally Adaptive Fault Mitigation in Space," *19th International Conference on Field Programmable Logic and Applications*, IEEE, Piscataway, NJ, 2009, pp. 199–204.

[2] Ostler, P. S., Bridgford, B., Swift, G., and Napier, M., "A Triple Module Redundancy Scheme for SEU Mitigation of Static Latch-Based FPGAs," *Military/Aerospace PLD (MAPLD) Conference*, NASA, Washington, D.C., 2004.

[3] Garvie, M., and Thompson, A., "Scrubbing Away Transients and Jiggling Around the Permanent: Long Survival of FPGA Systems Through Evolutionary Self-Repair," *Proceedings of the 10th IEEE International On-Line Testing Symposium*, IEEE, Piscataway, NJ, 2004, pp. 155–160.

[4] Quinn, H., and Graham, P., "Terrestrial-Based Radiation Upsets: A Cautionary Tale," *Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2005)*, IEEE, Piscataway, NJ, 2005, pp. 193–202.

[5] Miller, G., Carmichael, C., and Swift, G., "Single-Event Upset Mitigation for Xilinx FPGA Block Memories," XAPP 962, Ver. 1.1, Xilinx, San Jose, CA, March 2008, http://www.xilinx.com/support/documentation/application_notes/xapp962.pdf [retrieved 2013].

[6] Caffrey, M., Morgan, K., Roussel-Dupre, D., Robinson, S., Nelson, A., Salazar, A., Wirthlin, M., Howes, W., and Richins, D., "On-Orbit Flight Results from the Reconfigurable Cibola Flight Experiment Satellite (CFESat)," *Proceedings of the 17th IEEE Symposium on Field Programmable Custom Computing Machines*, IEEE, Piscataway, NJ, 2009, pp. 3–10.

[7] Seagrave, G., "SpaceCube: A Reconfigurable Processing Platform for Space," *Military/Aerospace PLD (MAPLD) Conference*, NASA, Washington, D.C., 2008.

[8] Kawai, H., "A Tile-Based Fault-Tolerant Approach for a Long-Life VLSI System," M.S. Thesis, Graduate School of Systems and Information Engineering, Univ. of Tsukuba, Tsukuba, Japan, 2007.

[9] Kang, D., Jhang, K., and Oh, D., "Design and Implementation of a Radiation Tolerant On-Board Computer for Science Technology Satellite-3," *NASA/ESA Conference on Adaptive Hardware and Systems*, NASA, Washington, D.C., 2010, pp. 17–23.

[10] Fras, M., Kroha, H., Reimann, O., Weber, B., and Richte, R., "Use of Triple Modular Redundancy (TMR) Technology in FPGAs for the Reduction of Faults Due to Radiation in the Readout of the ATLAS Monitored Drift Tube (MDT) Chambers," *Topical Workshop on Electronics for Particle Physics*, IOP Publishing, Ltd., London, 2010.

[11] Kraja, F., and Acher, G., "Using Many-Core Processors to Improve the Performance of Space Computing Platforms," *Proceedings of the 2011 IEEE Aerospace Conference*, IEEE, Piscataway, NJ, 2011, pp. 1–17.

[12] Singh, K., Agbaria, D., Kang, I., and French, M. C., "An Evaluation of Software Fault Tolerance Techniques on a Tiled Architecture," *9th Annual International Conference on Military and Aerospace Programmable Logic Devices*, NASA, Washington, D.C., 2006.

[13] Gauer, C., LaMeres, B. J., and Racek, D., "Spatial Avoidance of Hardware Faults Using FPGA Partial Reconfiguration of Tile-Based Soft Processors," *Proceedings of the 2010 IEEE Aerospace Conference*, IEEE, Piscataway, NJ, 2010.

[14] Ricky, W. B., "A Primer on Architectural Level Fault Tolerance," NASA Scientific and Technical Information (STI) Program Office, Rept. NASA/TM-2008-215108, Feb. 2008.

[15] Rollins, N., Fuller, M., and Wirthlin, M. J., "A Comparison of Fault-Tolerant Memories in SRAM-Based FPGAs," *Proceedings of the 2010 IEEE Aerospace Conference*, IEEE, Piscataway, NJ, 2010.

[16] Gericota, M. G., Lemos, L. F., Alves, G. R., Barbosa, M. M., and Ferreira, J. M., "A Framework for Fault-Tolerant Real Time Systems Based on Reconfigurable FPGAs," *Proceedings of the 11th IEEE Conference on Emerging Technologies and Factory Automation*, IEEE, Piscataway, NJ, 2006, pp. 131–138.

[17] Buerkle, T., LaMeres, B. J., Kaiser, T., Gowens, E., Smoot, L., Heetderks, T., Schipf, K., Clem, L., Schielke, S., and Luhr, R., "Ionizing Radiation Detector for Environmental Awareness in FPGA-Based Flight Computers," *IEEE Sensors Journal*, Vol. 12, No. 6, 2012, pp. 2229–2236.

[18] Hane, J., "A Fault-Tolerant Computer Architecture for Space Vehicle Applications," *Proceedings of the 11th IEEE Conference on Emerging Technologies and Factory Automation*, M.S. Thesis, Electrical and Computer Engineering Dept., Montana State Univ., Bozeman, MT, 2012.

[19] Tylka, A. J., Adams, J. H., Jr., 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, Dec. 1997, pp. 2150–2160.
doi:10.1109/23.659030

[20] Lesea, A., and Alfke, P., "Xilinx FPGAs Overcome the Side Effects of Sub-40nm Technology," Xilinx Rept. WP256, San Jose, CA, Oct. 2011.

[21] Quinn, H., Morgan, K., Graham, P., Krone, J., and Caffrey, M., "Static Proton and Heavy Ion Testing of the Xilinx Virtex-5 Device," *Proceedings of the 2007 Radiation Effects Data Workshop*, IEEE, Piscataway, NJ, 2007, pp. 177–184.

[22] Berg, M., "Xilinx Virtex 5 Proton Accelerated Radiation Test," Xilinx Rept. D062209_XCVLX30T, San Jose, CA, 2010, http://radhome.gsfc.nasa.gov/radhome/papers/D062209_XCVLX30T.pdf [retrieved 2013].

[23] Monson, J. S., Wirthlin, M., and Hutchings, B., "A Fault Injection Analysis of Linux Operating on an FPGA-Embedded Platform," *International Journal of Reconfigurable Computing*, Vol. 2012, Dec. 2011, Paper 850487.

G. Brat
Associate Editor