

# Hybrid, Adaptive, and Reconfigurable Fault Tolerance

Christopher Wilson  
NSF CHREC Center  
ECE Dept., University of Florida  
Room 317, Benton Hall  
Gainesville, FL 32611-6200  
wilson@chrec.org

Alan George  
NSF CHREC Center  
ECE Dept., University of Florida &  
University of Pittsburgh  
Room 1238D, Benedum Hall  
Pittsburgh, PA 15261  
george@chrec.org

Sebastian Sabogal  
NSF CHREC Center  
ECE Dept., University of Pittsburgh  
Room 1238D, Benedum Hall  
Pittsburgh, PA 15261  
ssabogal@chrec.org

Ann Gordon-Ross  
NSF CHREC Center  
ECE Dept., University of Florida  
Room 319, Larsen Hall  
Gainesville, FL 32611-6200  
ann@chrec.org

*Abstract*—The main design challenge in developing space computers featuring hybrid system-on-chip (SoC) devices is determining the optimal combination of size, weight, power, cost, performance, and reliability for the target mission, while addressing the complexity associated with combining fixed and reconfigurable logic. This paper focuses upon fault-tolerant computing with adaptive hardware redundancy in fixed and reconfigurable logic, with the goal of providing and evaluating tradeoffs in system reliability, performance, and resource utilization. Our research targets the hybrid Xilinx Zynq SoC as the primary computational device on a flight computer. Typically, flight software on a Zynq runs on the ARM cores that by default operate in symmetric multiprocessing (SMP) mode. However, radiation tests have shown this mode can leave the system prone to upsets. To address this limitation, we present a new framework (HARFT: hybrid adaptive reconfigurable fault tolerance) that enables switching between three operating modes: (1) ARM cores running together in SMP mode; (2) ARM cores running independently in asymmetric multiprocessing (AMP) mode; and (3) an FPGA-enhanced mode for fault tolerance. While SMP is the default mode, AMP mode may be used for fault-tolerant and real-time extensions. Additionally, the FPGA-enhanced mode uses partially reconfigurable regions to vary the level of redundancy and include application- and environment-specific techniques for fault mitigation and application acceleration.

## TABLE OF CONTENTS

1. INTRODUCTION AND BACKGROUND .....	1
2. BACKGROUND .....	2
3. APPROACH .....	4
4. EXPERIMENTS AND RESULTS .....	6
5. CONCLUSION.....	9
ACKNOWLEDGEMENTS .....	10
REFERENCES.....	10
BIOGRAPHY.....	11

## 1. INTRODUCTION AND BACKGROUND

Due to continuing innovations in sensors and research into autonomous operations, space processing has been unable to satisfy computing demands for new mission requirements. A major challenge for both commercial and government space organizations is development of new, higher-performance, space-qualified processors for new missions. Space missions

include unique requirements, with dramatic restrictions in size, weight, power, and cost (SwaP-C), and reliability demands in the presence of unique hazards (radiation, temperature, vibration, vacuum), which often have no corresponding terrestrial applications, and so technology developers must consider these requirements closely [1].

Space is a hazardous environment that necessitates special considerations for computing designs to work as intended. A plethora of particles from varying radiation sources can affect electronic components [2]. Radiation effects can be broadly organized into two categories: short-term transient effects and long-term cumulative effects. Transient effects can be further classified into “soft” (recoverable/non-destructive) and “hard” (non-recoverable/destructive) errors. Soft errors widely include all types of single-event effects (SEEs) such as single-event upsets (SEU), single-event functional interrupts (SEFI), and single-event transients (SET). Hard errors typically include single-event latch-up (SEL), single-event burnout (SEB), and single-event gate rupture (SEGR). These effects are extensively covered in [3], [4], and [5]. To better prepare spaceflight projects and payloads for exposure to these hazards, NASA developed a multi-step approach for design development that addresses radiation concerns. This approach was entitled Radiation Hardness Assurance (RHA) published in 1998 by LaBel et al. in [6] and revised in [7].

General-purpose processors and FPGAs can manifest radiation errors from SEEs differently. The main source of radiation concerns for SRAM-based FPGAs is corruption in the device-routing configuration memory and app-oriented block RAMs. Configuration memory allows the FPGA to maintain its pre-programmed, architecture-specific design; therefore, an upset to configuration memory can dramatically change the desired function of the device. These memory structures along with flip-flops are particularly vulnerable to radiation. To counter errors with radiation effects, designers employ configuration memory scrubbing. Scrubbing is the process of quickly repairing configuration-bit upsets in the FPGA before they render the device inoperable [8]. Additionally, designers use Error-Correction Codes (ECC) and parity schemes for block RAMs and some FPGA configuration memory. Finally, a common approach is to triplicate design structures in the

FPGA using triple-modular redundancy (TMR). ASICs and general-purpose processors include a separate set of concerns in comparison to FPGAs. Memory and logical elements of these processors such as general-purpose registers, the program counter, Translation Lookaside Buffer (TLB) entries, memory buffers, or the branch predictor can also be upset, causing a variety of adverse effects [9]. SEEs in a processor can manifest as a program crash, a hanging process, a data error, an unexpected reset, or performance degradation [10]. There are various mitigation strategies for processors in terms of hardware, information, network, software, and time redundancy, but some of the more commonly used are error detection and correction (EDAC) in hardware, ECC, parity bit checking, redundant boot code, mirrored files, and finally time redundancy with time triple-modular redundancy (TTMR) [11].

Microprocessors and FPGAs each include their own set of architecture advantages and disadvantages. Typically, FPGAs are extremely useful for hardware acceleration of algorithms with a high degree of streaming, data-flow parallelism, while consuming less power than a general-purpose processor. FPGAs are not as efficient when confronted with control-flow applications, which are better suited on a CPU. To take advantage of both types of architectures, vendors have developed hybrid System-on-Chip (SoC) processors. These SoCs are a combination of two or more processing technologies, featured on the same chip. In recent years, these types of devices have become commercially viable for space applications and have been included in several new space missions.

NASA has considered the complexities of SoCs for space as they have gained increasing interest in the space community for small missions. NASA first published their preliminary investigations with SoCs in [12] and later developed a full SEE radiation-testing guideline [13]. The testing guideline tries to highlight basic structures and systems that SEEs would generally affect. The main emphasis in these papers is to describe how to radiation-test complex SoC devices, however, they do not go into broader detail on suggested methods in fault-tolerant computing to mitigate SEE errors.

The SoC highlighted in our research is the Xilinx Zynq-7020 device. This SoC features dual-core ARM Cortex-A9 processors and an Artix-7 FPGA fabric [14]. This technology is prominent in the space community and featured in several commercial single-board computers from a variety of vendors, such as SpaceMicro’s CHREC Space Processor v1 (CSPv1) developed by our group, Innoflight’s Compact Flight Computer (CFC-300), GomSpace’s Nanomind Z7000, and finally Xiphos’ Q7 processor.

There are many schemes for fault and error mitigation for both fixed-logic processors and reconfigurable-logic FPGAs. Our research, however, focuses on developing a fault-tolerant computing strategy that accounts for the hybrid nature of an SoC device and suggests a strategy that works cooperatively between both types of architectures. We call this framework HARFT, for hybrid, adaptive, and reconfigurable fault tolerance.

## 2. BACKGROUND

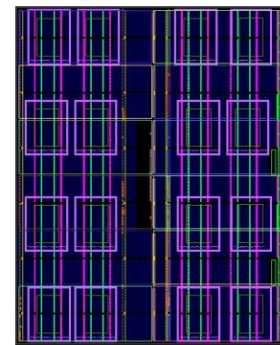
This section provides background information on some of the techniques examined in the fault-tolerant computing research described in this paper. Additionally, this section also provides an overview of related works that contributed to the final development of the HARFT prototype.

### *Partial Reconfiguration on FPGAs*

Partial reconfiguration (PR) is the process of reconfiguring a specialized section of the FPGA during operational runtime. In Xilinx devices, PR is possible through a modular design technique known as *partitioning*. In the typical FPGA programming process, FPGA configuration memory is programmed with a bitstream that specifies the design. In PR, partial bitstreams are loaded into specific reconfigurable regions of the FPGA without compromising the integrity of the rest of the system or interrupting holistic system operation. There are many benefits to using PR in space applications and missions. A designer can use PR to reduce total area utilization by swapping designs in a PR region, instead of statically placing all designs simultaneously. This scheme reduces the required amount of configuration memory and FPGA resources used, which in turn reduces the area vulnerable to SEEs. Correspondingly, a decrease in area also decreases power consumption for the device, which is valuable in small-satellite missions with particularly pressing power constraints. PR is a key component of several FPGA fault-tolerant computing strategies that designers can use in space. Finally, due to the smaller storage size of a partial bitstream (compared to a full bitstream), PR allows for faster and easier transfer of new applications to a device, enabling the spacecraft to conduct new, secondary mission experiments. References [15] and [16] provide more details for partial reconfiguration on the Zynq.

### *Radiation Tolerant SmallSat (RadSat) Computer System*

PR is a technique gaining rising attention in the SmallSat community. Radsat [17], a commercial-off-the-shelf (COTS) CubeSat developed by Montana State University and NASA Goddard Space Flight Center (GSFC), is one example that demonstrates PR-based fault tolerance.



16 MicroBlaze Processors on Virtex-6

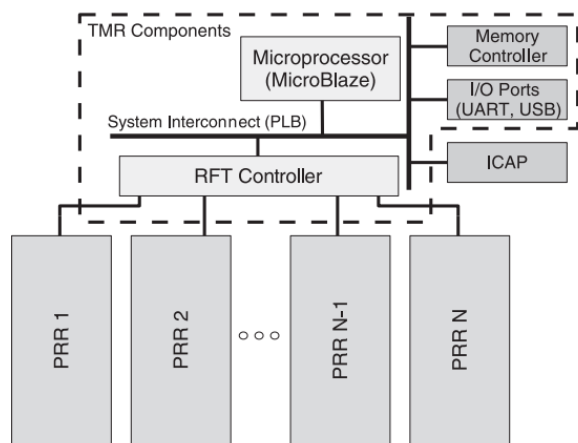
**Figure 1. RadSat FPGA Architecture Layout with Partial Reconfiguration Regions [17]**

RadSat focuses on unique fault-tolerant computing methods for the Virtex-6 FPGA. Since the Virtex-6 is not an SoC, all necessary software is executed on softcore processors (CPUs created with FPGA resources) such as the Xilinx MicroBlaze.

In their proposed system, the FPGA fabric has multiple partially reconfiguration regions (PRRs), where three of the regions run MicroBlazes in TMR, while the remainder of the PRRs are spare regions. With this technique, when the TMR system detects a fault, the damaged region is replaced with a spare region and is reprogrammed in the background using PR. To mitigate other faults, the scrubber performs blind scrubbing (simple periodic configuration writeback without checking for errors) on the PRRs, while deploying readback scrubbing (scrubbing while reading back the contents of a frame to check for errors) through the rest of the static region of the fabric. Figure 1 depicts the RadSat architecture layout and placement blocks for the PRRs.

*Reconfigurable Fault Tolerance (RFT)*

Another technique that builds on PR-based hardware is RFT. This framework, described in [18], seeks to enable a system to autonomously adapt and change fault-tolerant computing modes based on current environmental conditions. In this system, the architecture uses PRRs in parallel to create different redundancy-based, fault-tolerant modes, such as duplex with compare (DWC) and TMR. Other mitigation techniques include algorithm-based fault tolerance (ABFT) and watchdog timers. In their framework, the internal processor evaluates the current performance requirements and monitors radiation levels (with an external sensor, or by monitoring configuration upsets) to determine when the operating mode should be switched. The overall contribution of their strategy is that it allows a system to maintain high performance by swapping in various hardware accelerators in the PRRs, however, when environmental conditions deteriorate, the system can program critical applications into the regions with varying levels of redundancy and fault tolerance. Figure 3 illustrates the RFT architecture.

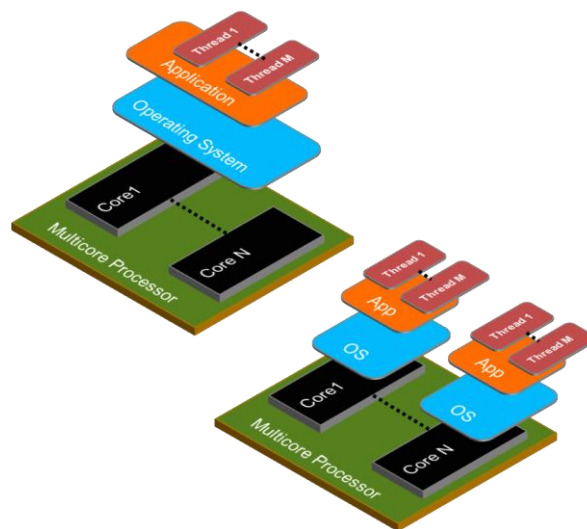


**Figure 2. RFT Architecture [18]**

*Symmetric and Asymmetric Multiprocessing (SMP / AMP)*

The Zynq is a highly capable device due to the hybrid nature of its SoC design including both ARM cores and FPGA fabric. So far, this paper has only considered techniques applicable to the FPGA fabric; therefore, this section describes unique capabilities available to the ARM processing system. The ARM cores on the Zynq are capable of running a variety of Linux (and other) operating-system kernels. The default configuration for running Linux on a development board is symmetric multiprocessing (SMP) mode. SMP is a processing model that consists of a single operating system controlling two or more identical processor cores symmetrically connected to main memory and sharing system resources. This type of configuration is beneficial for running applications configured for multithreaded processing. SMP makes it possible to run several software tasks concurrently by distributing the computational load over the cores in the system. Asymmetric multiprocessing (AMP) differs from SMP in that the system can include multiple processors running a different operating system on each core. Typical examples include a more full-featured operating system running on one processor, complemented by a smaller, lightweight, efficient kernel running on the other processor [19][20]. Figure 4 demonstrates the difference between the configurations. There are many potential benefits for this type of operation [21], including:

- Allowing a designer to segregate flight system operations and science applications for system integrity
- Providing the ability to create a lightweight virtual machine on the system
- Use of one core to be isolated as a secure-software zone for security applications
- Use of secondary core to provide a real-time component to system by running FreeRTOS or other lightweight, real-time operating systems
- AMP allows for additional fault-tolerant techniques by setting up the system for duplex with compare
- The secondary core also provides easier certification for applications due to smaller codebase size for review



**Figure 3. SMP (Top) vs. AMP (Bottom) Illustration**

### Lockstep Operation

In addition to the division of cores with AMP, lockstep operation is another type of fault tolerance that designers can apply to CPUs. Lockstep operation is in essence an extension of a single core with hardware checking [22]. Lockstep systems run the same operations in parallel. Figure 5 is a graphical depiction of the lockstep process. Lockstep systems detect and correct operation errors by comparing the outputs of the cores dependent on the number of systems that are in lockstep [23].

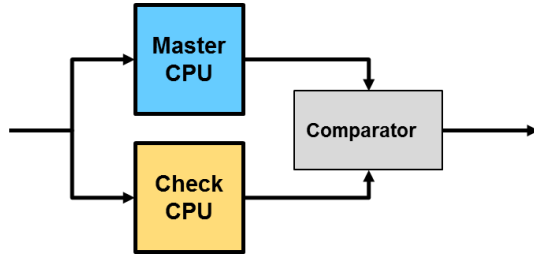


Figure 4. Lockstep Operation

### CHREC Space Processor

Researchers in our NSF Center for High-Performance Reconfigurable Computing (CHREC) developed a design concept known as CHREC Space Processor (CSP). The CSP concept features a mix of commercial technology (for best in performance, energy, cost, size, and weight) for data processing, radiation-hardened technology (for best in reliability) for monitoring and management, and selected methods in fault-tolerant computing (selected from hardware, information, network, software, or time redundancy). The first incarnation of the CSP concept is the CSPv1 flight board featuring a Xilinx Zynq-7020 SoC [24].

## 3. APPROACH

Both complex algorithms and resource-intensive processing found in new science-mission applications challenge the space-computing community. Therefore, the community has turned to identifying next-generation systems that can support a wide range of capabilities, for low power and high reliability. Several organizations have identified multicore, hybrid SoC devices as a promising architecture for space computing. To increase the reliability of such devices, this paper proposes a multifaceted strategy for fault-tolerant computing, targeting SoC devices composed of multicore CPUs and FPGA fabric. Our HARFT strategy incorporates fault-tolerant schemes with both architectures to create a robust, hybrid, fault-tolerant theme for a hybrid device.

### Flight Example

In a science mission, a spacecraft may experience varying levels of radiation from several sources including the South Atlantic Anomaly (Figure 6) and unexpected solar weather conditions. The system operates by default in the SMP mode. The configuration manager changes the mode dynamically, by reading the current upset rate detected by the scrubber, or from previously set configurations defined by the ground station.

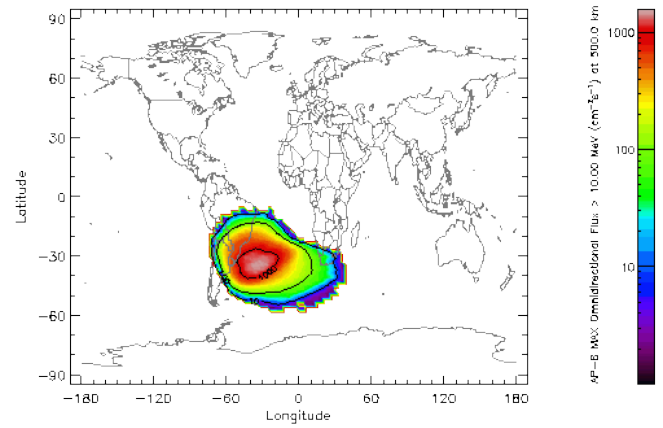


Figure 5. World Map Displaying Proton Flux at South Atlantic Anomaly [25]

### HARFT Hardware Architecture

HARFT is subdivided into three main subsystems: the hard-processing system (HPS); the soft-processing system (SPS); and the configuration manager (ConfigMan). The HPS consists of the ARM dual-core Cortex-A9 processor and its internal resources. The SPS consists of programmable-logic elements of the Artix-7 FPGA fabric. Figure 8 illustrates a high-level block diagram of the architecture design.

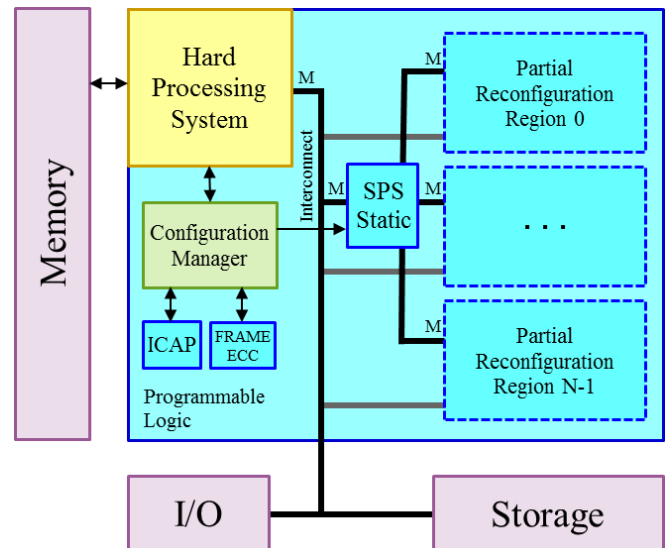


Figure 6. HARFT Architecture Diagram

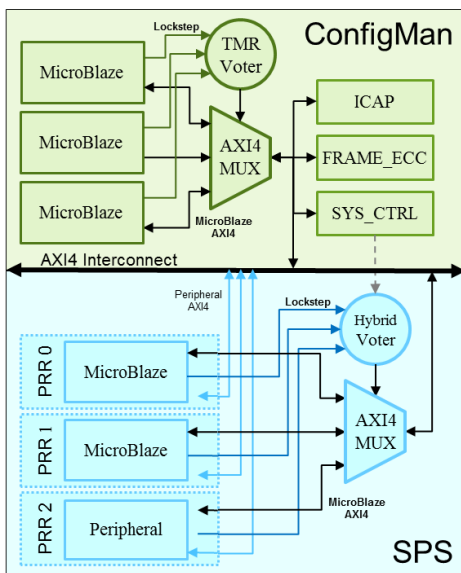
**Hard-Processing System (HPS)**—The HPS encapsulates the ARM cores and all of the processor resources. The Zynq architecture does not support lockstep operation in Cortex-A9 cores; therefore, fault-tolerant strategies on the HPS involve alternating between the SMP and AMP modes. Unfortunately, there are some limitations to AMP on Xilinx devices. Xilinx documentation notes that since there are both private and shared resources for each CPU, careful consideration is necessary to prevent resource contention. Linux manages and controls most shared resources, so it is infeasible to run Linux on both cores of the device simultaneously. CPU0 controlling shared resources from Linux forces CPU1 to run an operating system with fewer restrictions, such as FreeRTOS, or custom bare-metal

software. Consequently, software developers may have to re-write applications specifically for CPU1. Xilinx provides AMP-related projects and examples in their application notes [26]-[28].

*Soft-Processing System (SPS)*—The SPS constitutes a scalable number of PRRs and a static-logic component. Each PRR can be configured as either a Xilinx MicroBlaze processor or an auxiliary hardware accelerator. MicroBlazes instantiated in the PRRs operate in lockstep and aggregate as one redundant processor. The static logic in the SPS contains a hybrid comparator/voter with AXI4 bus arbitration, reset control, and PR glue logic.

### Configuration Manager (ConfigMan)

An essential component of HARFT is the ConfigMan. This component is an independent, triplicated MicroBlaze system executing operations in lockstep, residing in the static logic of the programmable-fabric design. The ConfigMan is multipurpose, and can perform operations such as FPGA configuration-memory scrubbing, act as a fault monitor by recording upset events, and adapt the system by triggering fault-tolerant mode changes. The ConfigMan accesses the FPGA configuration memory using the AXI Hardware Internal Configuration Access Port (AXI\_HWICAP) IP core (ICAPE2 primitive) and obtains the configuration memory frame ECC syndrome using a custom AXI-based IP core (FRAME\_ECCE2 primitive) [29].



**Figure 7. ConfigMan and SPS-SL Architecture Diagram**

*ConfigMan Scrubbing*—To perform scrubbing, the ConfigMan instructs the ICAPE2 to readback one FPGA frame. During this readback, the FPGA frame passes automatically through the FRAME\_ECCE2 block to compute the ECC syndrome. The ConfigMan reads the FPGA frame from the AXI\_HWICAP buffer into local memory and reads the ECC syndrome from the FRAME\_ECCE2 block. If the syndrome is zero, then there was no error detected and the ConfigMan proceeds to inspect the next FPGA frame. If the syndrome is nonzero then an error is present and the syndrome is decoded to

determine the word and bit location of the fault (Note: some errors are detectable but are uncorrectable, which are resolved with a full system reset). An FPGA frame is corrected by flipping the faulty bit in the frame stored in local memory, as located by the ECC syndrome. The ConfigMan instructs the ICAPE2 for FPGA frame write-back to correct the frame in configuration memory. There are 7692 frames in the Zynq-7020 device, with 101 words per frame, and 32 bits per word. More information detailing these interactions can be found in [30].

*ConfigMan Mode-Switching Mechanics*—When the fault-tolerant mode changes, the ConfigMan transfers partial bitstream(s) from DDR memory to the AXI\_HWICAP for PR. A mode switch that increases the number of processors (e.g., simplex to duplex) requires a reset of the SPS to resynchronize the MicroBlazes for lockstep operation. However, when the mode switch decreases the number of processors (e.g., TMR to simplex), no reset is required since the leftover MicroBlazes remain synchronized. ConfigMan handles PR efficiently when switching modes; only the necessary regions are reconfigured.

*ConfigMan Mode Switching Process*—ConfigMan triggers mode switching in two ways. The first is an adaptive-mode switching based on incoming upsets and recorded faults by the ConfigMan. Since the ConfigMan is programmable, the user can program various algorithms, such as the windowing strategy in [18]. The second mode switch occurs when the ConfigMan receives a command from the ground station to place the system into a particular mode for a specific period of time. An example of this need is for an incoming solar flare, where controllers on the ground can force the ConfigMan prior to the event to change the fault-tolerant strategy in advance.

### SPS Static Logic

The second essential component of HARFT is the SPS-Static Logic (SL). The SPS-SL is, in essence, a custom IP core that is a hybrid comparator/voter combined with an AXI Multiplexer. Each of the MicroBlazes from the PRRs includes lockstep signals, which partially contain the processor state of the MicroBlaze (IP\_AXI Instruction Bus and DP\_AXI Data Bus). These signals are inputs to the SPS and multiplexed to the output depending on the current fault-tolerant mode configuration. Figure 9 illustrates the ConfigMan and SPS-SL interactions.

### Fault-Tolerant Mode Switching

The ConfigMan dynamically switches between three main fault-tolerant modes during flight operations. These modes refer to a specific configuration of the HPS and the SPS on the device. Figure 10 shows a graphical diagram highlighting the modes.

(1) *SMP + Accelerators*—In this mode, Linux runs on both Cortex-A9 cores in SMP mode. The PRRs are allocated for hardware acceleration. This mode is the highest-performance mode; the HPS provides high-performance software execution, accelerating applications by using parallel

computing tools, such as OpenMP, and leveraging hardware accelerators instantiated in the FPGA.

(2) *AMP + Accelerators*—In this mode, Linux runs on only one Cortex-A9 core (CPU0). Depending on the mission constraints, a real-time operating system (RTOS), such as FreeRTOS can run on CPU1 for real-time operations. Alternatively, CPU1 can run the bare-metal equivalent to the Linux CPU0 application in a duplex-like mode, using shared memory to pass status and health updates. In this scenario, the PRRs can also be allocated to hardware acceleration.

(3) *FPGA-Enhanced Fault Tolerance (FEFT)*—The final reliability mode refers to a number of sub-configurations available in the FPGA fabric. The configurations describe combinations of either MicroBlaze processors or hardware accelerators in the FPGA fabric (e.g., two MicroBlaze processors in two PRRs, with remaining PRRs as hardware accelerators). These configurations feature at least one MicroBlaze in a PRR, with the rest of the PRRs filled with hardware accelerators. If there is more than one MicroBlaze, they will operate in lockstep. Once this mode engages, the MicroBlaze(s) will take control of key flight-system applications. This mode is the most reliable; however, the MicroBlazes operate at a much slower clock frequency than the ARM cores on the HPS system, and therefore have much lower performance.

#### Mode Switching

The ConfigMan is responsible for switching modes in the FPGA while in the FEFT mode. To switch between SMP and AMP, a simple script renames the boot files, since each configuration has different settings for U-Boot and corresponding first-stage boot loader.

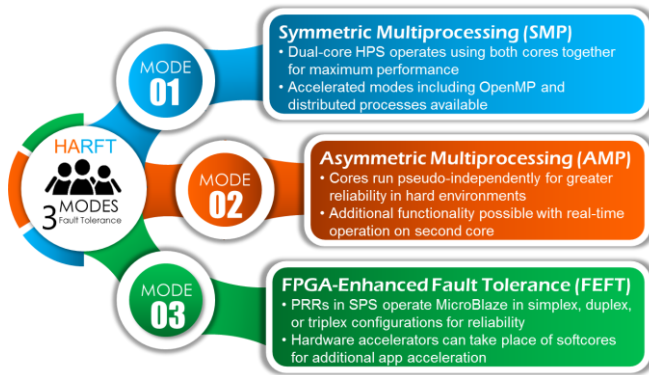


Figure 8. Illustrated Fault-Tolerant Modes

#### Challenges

When designing a system using HARFT, the developer should consider several issues for a specific mission. We recommend HARFT for those familiar with Xilinx software development, FPGA development, and Linux development. Configuration for AMP requires designers to change configuration settings in U-Boot and make modifications to the stand-alone board support package (BSP) for the first-stage boot loader and additional applications. For this design, Xilinx provided the custom BSP supporting AMP on the Zynq. Additionally, we do not recommend switching

tool versions in development, since the build process varies drastically in different Xilinx versions. At present, HARFT uses Vivado 2015.4 and SDK, and we encountered several issues using Vivado including randomly disconnecting signals, and changing parameters and configurations.

#### Flight Configuration and Use Model

We designed HARFT to perform optimally in low-Earth orbit (LEO) and environments that include a typical profile of generally lower upset rates with short bursts of time with relatively higher upset rates. The limits of HARFT are closely tied with the radiation-effect limits of the Zynq, and HARFT was specifically structured for the CSPv1 flight unit configuration. Developers may wish to fly the 7-series Zynq with caches disabled due to the behavior described in [34]. We also recommend ECC on the DDR memory due to the need to store bitstreams between configurations. Finally, radiation-hardened or -tolerant (with multiple images) non-volatile storage is recommended, so that boot images for SMP and AMP modes remain uncorrupted.

## 4. EXPERIMENTS AND RESULTS

This section discusses experiments and HARFT prototype development to evaluate our ideas and architectural design. First, we discuss general experiments, which verify the limitations of the processor modes and expected behavior on a testbed. These experiments show the strong need for adaptive flexibility in a changing radiation environment. Next, this section provides a brief overview of the radiation-effects methodology introduced in [31] that determines the estimated effectiveness of our proposed method. Finally, we describe the developed prototype for HARFT, highlight conducted metrics and benchmarks, show the FPGA resource utilization and scrubber performance, and discuss expected HARFT behavior due to radiation effects.

#### Processor Experiments

We examine several processor tests as part of the problem-determination phase of this research and for familiarization with AMP configuration on the Zynq. These tests consist of configuring the operating system for each test, and then halting one of the cores or corrupting the program counter (PC) in order to crash the program using the built-in debugging tools.

*Basic SMP Experiment*—This simple experiment confirms that unexpected errors (which could be the result of an SEE) in one of the cores in SMP mode will lead to a system crash. This outcome is significant because if SMP does not crash from an upset in one of the cores then AMP would not be necessary. Xilinx (Xilinx Linux) ran across both CPU0 and CPU1 in SMP mode. We conducted 10 runs for each test (halting and crashing) on both processing cores. When one of the cores halts the system, the behavior is not deterministic, and occasionally, in several tests, the system would continue to operate, while in others the system suffered a crash. When the PC of one of the cores changes to an unexpected address, the system always results in a crash.

*Basic AMP Experiment*—This experiment shows the resilience of an AMP-configured design, and establishes that it performs as expected on a hardware testbed. In this experiment, CPU0 runs Xilinx, CPU1 runs a bare-metal application, and a MicroBlaze runs another bare-metal application. Once again, we conducted 10 runs for both types of tests on each of the processors. When either of the processor cores halts, the other core continues to function nominally, and the MicroBlaze remains unaffected. Similarly, when one of the cores has its PC set to an unexpected address, the other core, as well as the MicroBlaze, continues operation as intended.

*Reliability Modeling*

To analyze HARFT, we create a dynamic fault-tree model as described in [31] as part of a CubeSat reliability methodology. This methodology relies on tools including CRÈME and PTC Windchill Predictions to build a model of the processing system and programmable logic.

*CRÈME*—CRÈME is a state-of-the-art tool for SEE-rate prediction. The tool allows the user to generate upset rates for individual components in varying Earth orbits. CRÈME also allows a user to simulate different conditions of an orbit as it relates to solar weather and galactic cosmic rays [32].

*Modeling Methodology*—The work in [31] provides a methodology for estimating the reliability of SmallSat computers in radiation environments. Our analysis uses the microprocessor submodule model to show upset rates of the programmable-logic and processing-system portions of the Zynq. In this submodule, each mode has a constructed dynamic fault tree (DFT) that models the Zynq architecture. For our analysis, we use proprietary Weibull curves (inputs into CRÈME) gathered for the main Zynq components in the processing system and programmable logic from radiation test reports. CRÈME then generates the upset rates based on the specified orbit. The DFT-submodule “basic events” have the previously calculated CRÈME upset rates as inputs.

*HARFT Prototype Description*

As a proof-of-concept for HARFT, we create a prototype design using a Digilent ZedBoard containing the Zynq-7020 SoC. While our HARFT description encompasses a number of possible configuration options, this section describes a single configuration that we built as a prototype.

**Table 1. PRR Resource Utilization**

Resource	PRR0	PRR1	PRR2	Total
Slice-LUTS	2428	2433	2440	13931
Slice-Registers	1884	1884	1884	11303
BRAM Tile	0	0	0	1
RAMB36	0	0	0	0
RAMB18	0	0	0	2
DSP48E1	6	6	6	18

*HPS Configuration*—In the prototype, the HPS is configured with a ZedBoard running a branch of Xilinx Linux. U-Boot

and the device tree are modified to add the necessary design-specific drivers, force single-processor operation (for AMP), and restrict DDR memory access available for the system (DDR memory must be reserved for the MicroBlaze and to store configurations). CPU1 runs a simple bare-metal application or FreeRTOS.

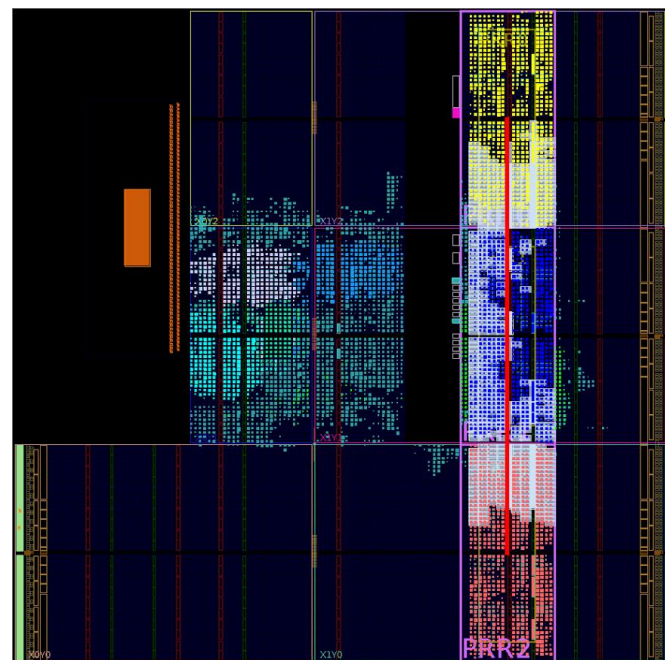
*SPS Configuration*—HARFT supports any number of desired PRRs within the resource constraints; for this prototype, we selected three PRRs. With three PRRs, possible modes for FEFT include Simplex, Duplex, and Triplex. The MicroBlazes are instantiated within the design and configured for maximum performance without caches or TLBs.

*ConfigMan Configuration*—The ConfigMan maintains a user-configurable number of thresholds to switch modes. If the ConfigMan detects a number of faults exceeding a threshold while scrubbing, it triggers a new configuration.

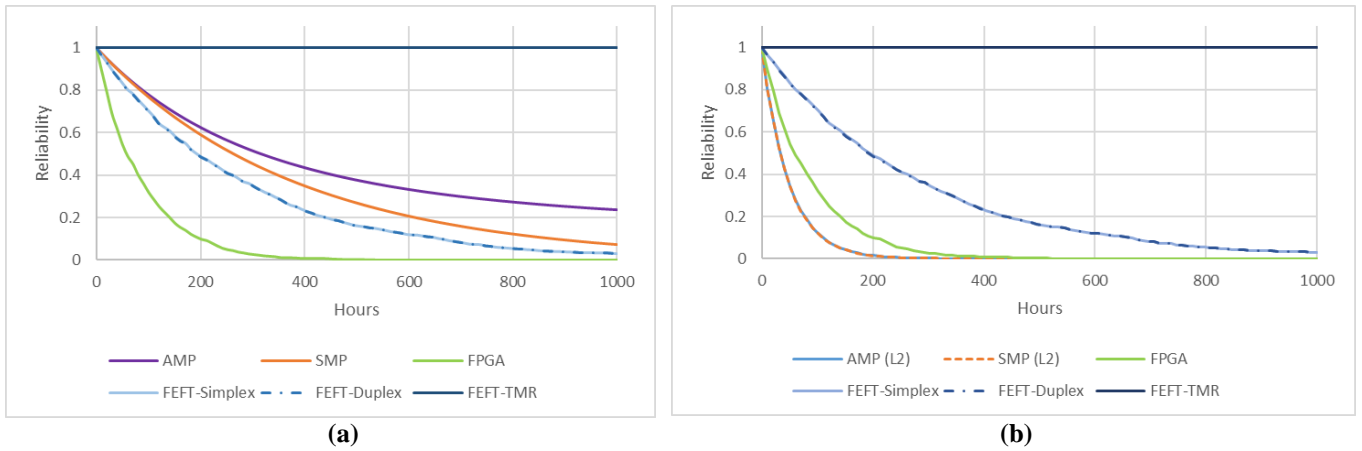
*Additional Hardware Configuration*—The prototype contains cores that would not be needed in a flight configuration including UARTS, PMOD UARTS, LED core, and switches. We place these cores explicitly for project debugging and testing.

**Table 2. Prototype Total Resource Utilization**

Resource	Used	Available	Util%
Slice-LUTS	13931	53200	26.19
Slice-Registers	11303	106400	10.62
BRAM Tile	1	140	0.71
RAMB36	0	140	0.00
RAMB18	2	280	0.71
DSP48E1	18	220	8.18



**Figure 9. FPGA Configuration Area**



**Figure 10. HARFT Reliability with (a) L2 Cache Disabled or (b) L2 Cache Enabled**

Table 1 and Table 2 list the resource utilization for the three PRRs and the complete prototype on the device. Figure 11 shows the entire placed and routed design. The cyan highlight denotes the ConfigMan, the light purple denotes the SPS-SL, and the light blue denotes the hardware test cores (UARTs, LEDs, including bus logic etc.). Lastly, the yellow, blue, and red regions represent the three PRRs.

*HARFT Prototype Analysis*

For this analysis, we calculate upset rates for LEO. These rates show the reliability of each mode respective to one another. The reliability of these modes in different orbits can be extrapolated from the relationship between the modes established from LEO results.

Figure 12 shows the reliability of the main modes of HARFT. Additionally, a reliability curve representing the FPGA, if every bit on the device is considered essential, is provided as a reference and is labeled “FPGA” in the graph. For the FEFT-mode calculations, we assume that any upset temporarily interrupting the processor is a failure. Using this model, FEFT-Duplex and FEFT-Simplex show similar rates because a single upset would cause either to fail, however, in practice FEFT-Duplex would detect the error, while FEFT-Simplex would continue until the device failed or the scrubber detected the error.

For these calculations, Xilinx guidelines state that 10% of configuration-memory bits are significant in any design. For the model, this fraction of each PRR and the static area is calculated and scaled to 10% of the total sensitive device-configuration bits.

**Table 3. FPGA Scrubbing Duration**

Operation	Duration (sec)
Readback (Entire FPGA)	14.5246
Readback (Frame)	0.001888
Writeback (Entire FPGA)	19.9478
Writeback (Frame)	0.002593

Results demonstrate that, as expected, level-two (L2) cache has a significant effect on the overall reliability of the

system. The L2 cache is responsible for a majority of upsets on the processing system, therefore, Figure 12a shows the reliability of all HARFT system modes with L2 cache disabled, while Figure 12b shows the same but with L2 cache enabled. Figure 12b shows that the most reliable mode for the system is the FEFT-TMR mode. In the chart, this reliability is near one. The result is due to the low number of faults expected in LEO, while the scrubber correction rate is extremely high as seen in Table 3, even under the worst-case scrubbing scenario (needing to read the entire FPGA and then writing to the correct frame). Figure 12a shows AMP is more reliable than SMP, while both are slightly more reliable than FEFT-Duplex and FEFT-Simplex.

As cited above, Figure 12b shows the same LEO example with L2 cache enabled. Since the L2 cache is responsible for the dominant portion of errors, the reliability of the modes is re-ordered. AMP and SMP modes have the worst reliability compared to all of the FEFT modes.

**Table 4. Computational Density Device Metrics**

Processor	Computational Density (GOPS)		
	INT8	INT16	INT32
ARM Cortex-A9 Dual-Core	32.02	16.01	8.00
ARM Cortex-A9 Single-Core	16.01	8.00	4.00
MicroBlaze	0.125	0.125	0.125

*Performance Modeling*

We calculate device metrics, as described in [33], using the theoretical maximum performance for each of the three modes, illustrated in Table 4. While floating-point calculations are available, this table only displays integer operations for brevity and to compare with benchmark results, which are integer only. It should be noted that, while the SMP mode may have lower reliability, it has dramatically increased performance over the FEFT mode.

To provide an alternate view of processor performance, we benchmark with CoreMark the featured processors. CoreMark is a benchmark developed by the Embedded



Microprocessor Benchmark Consortium with the goal of measuring the performance of embedded-system CPUs. Table 5 displays the results of the benchmarks and confirms the theoretical trends calculated for device metrics. We note that the L2 cache does not appear to improve the performance of the CoreMark benchmark. This result is explained by the benchmark's low memory usage versus the 32KB level one (L1) instruction and data caches. The data and bss segments amount to about 16KB, which is about half of the capacity of the L1 data cache. The text segment is about 67KB, which is larger than the size of the L1 instruction cache and may incur a slight performance penalty due to cache misses justifying the differences in the results.

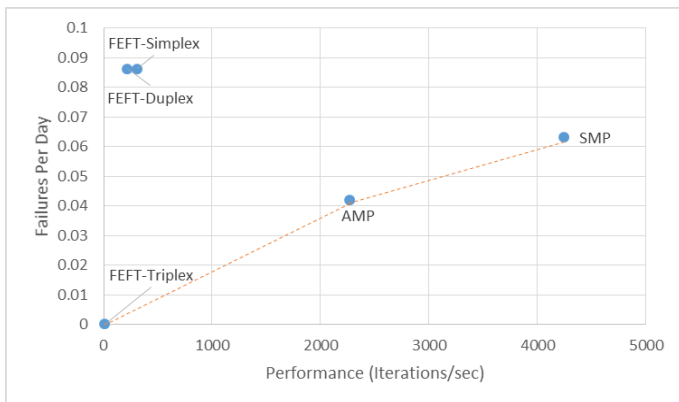
**Table 5. CoreMark Benchmarking**

Configuration	Iterations/sec
HPS (ARM) with Caches Single Core	1993.6778
HPS (ARM) w/o L2 Cache Single Core	1971.2254
SPS (MicroBlaze)	9.5975

We compile the benchmark used in Table 5 with "PERFORMANCE\_RUN" configuration and -O2 compiler optimizations. The number of iterations tested varied: 1000 for MicroBlaze, 100,000 for ARM. This discrepancy results from the enormous time requirement for the MicroBlaze to execute 100,000 iterations. The precision error is minimal at over 10 seconds of execution.

There are no specific application accelerators developed thus far for the prototype. Since the amount of speedup varies with application and hardware design, we assume that each PRR accelerator adds 100 iterations/sec to highlight and establish the general-profile trends for the reliability modes.

Figure 13a shows the reliability vs. performance of different modes depending on varying fault rates with L2 cache disabled. We estimate the calculations for SMP mode by doubling the single-core results of Table 5 (the CoreMark benchmark is single-threaded). The graph highlights the Pareto optimal line for the varying configurations and indicates that it is only useful to switch between AMP, SMP, and FEFT-Triplex when L2 cache is disabled.



(a)

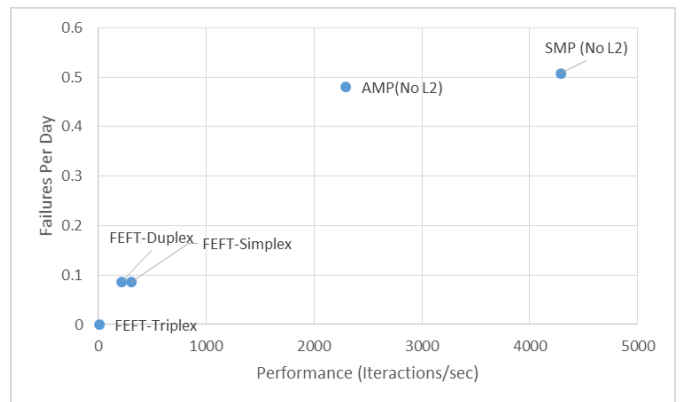
Figure 13b shows the same results with L2 cache enabled. The HPS shows drastically higher performance, however, it is much more prone to upsets. FEFT-Duplex and FEFT-Simplex are more viable in this configuration since they provide higher reliability than the HPS modes while still maintaining higher performance than FEFT-Triplex. This chart illustrates the flexible trade space for switching modes on a prototypical LEO mission.

## 5. CONCLUSION

This paper presents a novel, hybrid, fault-tolerant framework, HARFT, designed specifically to adapt to the dual architecture capabilities and needs of SoC devices. We built a specific HARFT configuration to test and verify the structural and design features as proof of concept. HARFT features three dynamically configured modes: (1) SMP + Accelerators; (2) AMP + Accelerators; and (3) FEFT + Sub-configurations. The benchmarking and reliability analysis list these modes in order of the highest to lowest in performance, and from lowest to highest in reliability. A custom-designed IP core, ConfigMan, simultaneously scrubs the FPGA for faults, determines upset rate, and dynamically reconfigures the fault-tolerant mode. Our experiments in this paper verify the functionality of the prototype, especially with regard to the behavior of the processing system modes in AMP and SMP. The analysis highlights that, since L2 cache is prone to upsets, the HARFT mode selection changes, depending on if the mission designer enabled or disabled the L2 cache. Finally, with these methods on a hybrid SoC, a spacecraft may adapt to changing environmental conditions in order to achieve a high level of both performance and reliability for each mission scenario.

### Future Plans

There are several features that we propose to improve the functionality and performance of HARFT, which could be investigated in future development. Several of these additional features are not complex; however, we did not include these features due to time restrictions. Key additions include dynamic recovery of system by working processors, checkpointing of system state, optimizing timing and FPGA performance, and finally, the use of machine intelligence in ConfigMan for mode switching.



(b)

**Figure 11. Upsets Per Day vs. Performance with (a) L2 Cache Disabled or (b) L2 Cache Enabled**

## ACKNOWLEDGEMENTS

This work was supported by the CHREC Center members and by the I/UCRC Program of the National Science Foundation under Grant No. IIP-1161022. The authors wish to thank Jason Gutel for preliminary AMP development, Dr. Adam Jacobs for guidance and knowledge related to RFT, Ed Carlisle for assistance in simple verification experiments, David Wilson for initial prototype studies, and Tyler Lovelly and Andy Milluzzi for providing device metrics for the Zynq and MicroBlaze, all from our Center. We also wish to thank John McDougall at Xilinx for providing BSPs for AMP.

## REFERENCES

- [1] R. Doyle et al., "High Performance Spaceflight Computing (HPSC) Next-Generation Space Processor (NGSP) A Joint Investment of NASA and AFRL," *Int. Symp. Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, Montreal, Canada, June 17-19, 2014.
- [2] S. Sayil, "Space Radiation Effects on Technology and Human Biology and Proper Mitigation Techniques", Texas Space Grant Consortium (TSGC) Higher Educ. Grant Final Rep., July 2010.
- [3] R. Maurer, M. Fraeman, M. Martin, and D. Roth, "Harsh Environments: Space Radiation Environment, Effects, and Mitigation," *Johns Hopkins APL Tech. Dig.*, vol. 28, no. 1, pp. 17-29, 2008.
- [4] D. White, "Considerations Surrounding Single Event Effects in FPGAs, ASICs, and Processors," Xilinx White Paper, March 7, 2012.
- [5] F. Sturesson, "Single Event Effects Mechanism and Effects," *Space radiation and its effect on EEE components*. EPFL Space Center, June 9th, 2009.
- [6] K.A. LaBel, A.H. Johnston, J.L. Barth, R.A. Reed, C.E. Barnes, "Emerging Radiation Hardness Assurance (RHA) issues: A NASA approach for space flight programs," *IEEE Trans. Nucl. Sci.*, pp. 2727-2736, Dec. 1998.
- [7] K.A. LaBel and J. Pellish, "Notional Radiation Hardness Assurance (RHA) Planning for NASA Missions: Updated Guidance," *NASA Electronics Parts and Packaging Program (NEPP) 2014 Electronics Technology Workshop (ETW)*, NASA Goddard Space Flight Center, Greenbelt, MD, June 17-19, 2014.
- [8] R. J. Andraka, P.E. Brady, and J. L. Brady, "A Low Complexity Method for Detecting Configuration Upset in SRAM based FPGAs," *6th Proc. Military and Aerospace Programmable Logic Devices Conf. (MAPLD)*. Washington D.C., September 9-11, 2003.
- [9] W. Robinson et al., "Soft Error Considerations for Multicore Microprocessor Design," in *Int. Conf. Integrated Circuit Design and Technology, (ICICDT'07)*. IEEE, pp. 1-4, 2007.
- [10] H. Quinn, T. Fairbanks, J. Tripp, G. Duran and B. Lopez, "Single-event effects in low-cost, low-power microprocessors", *IEEE Radiation Effects Data Workshop (REDW)*, 2014, pp. 1-9
- [11] R. DeCoursey, R. Melton, and R. Estes, "Non radiation hardened microprocessors in space-based remote sensing systems", *Proc. SPIE Sensors, Systems, and Next-Generation Satellites*, vol. 6361, 2006.
- [12] S. Guertin, "SOC Processors Radiation and Developments," *NASA Electronics Parts and Packaging Program (NEPP) Electronics Technology Workshop (ETW)*, NASA Goddard Space Flight Center, Greenbelt, MD, June 22-24, 2010.
- [13] S. Guertin, "SOC SEE (Radiation) Guideline Update," *NASA Electronics Parts and Packaging Program (NEPP) 2014 Electronics Technology Workshop (ETW)*, NASA Goddard Space Flight Center, Greenbelt, MD, June 22-24, 2010.
- [14] *Zynq-7000 All Programmable SoC Overview - DS190*. (2016, Sep 27). [Online]. Available: [http://www.xilinx.com/support/documentation/data\\_sheets/ds190-Zynq-7000-Overview.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf)
- [15] C. Kohn, "Partial Reconfiguration of a Hardware Accelerator with Vivado Design Suite for Zynq-7000 AP SoC Processor," Xilinx Application Note (XAPP1231), March 20, 2015.
- [16] *Vivado Design Suite User Guide Partial Reconfiguration - UG909*. (2016, Apr 6). [Online]. Available: [http://www.xilinx.com/support/documentation/sw\\_manuals/xilinx2016\\_1/ug909-vivado-partial-reconfiguration.pdf](http://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_1/ug909-vivado-partial-reconfiguration.pdf)
- [17] B. LaMeres et al., "RadSat - Radiation Tolerant SmallSat Computer System," *29th Annu. AIAA/USU Conf. on Small Satellites*, Logan, UT, August 8-13, 2015.
- [18] A. Jacobs, G. Cieslewski, A. D. George, A. Gordon-Ross, and H. Lam, "Reconfigurable Fault Tolerance," *ACM Trans. Reconfigurable Tech. and Syst.*, vol. 5, no. 4, pp. 1-30, Dec. 2012.
- [19] S. McNutt, "AMP up Your Next SoC Project," *Xcell Software Journal*, Issue 3, pp 28-33, First Quarter, 2016.
- [20] *Zynq-7000 All Programmable SoC Software Developers Guide*. (2015, Sep. 30). [Online]. Available: [http://www.xilinx.com/support/documentation/user\\_guides/ug821-zynq-7000-swdev.pdf](http://www.xilinx.com/support/documentation/user_guides/ug821-zynq-7000-swdev.pdf)
- [21] A. Taylor, "A Double-Barreled Way to Get the Most from Your Zynq SoC," *Xcell Journal*, Issue 90, pp. 38-45, First Quarter, 2015.
- [22] K. Greb, "Design How-To: Matching processor safety strategies to your system design," *EE Times*, Oct 2011.
- [23] *MicroBlaze Processor Reference Guide - UG984*. (2014, April 2). [Online]. Available: [http://www.xilinx.com/support/documentation/sw\\_manuals/xilinx2014\\_2/ug984-vivado-microblaze-ref.pdf](http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_2/ug984-vivado-microblaze-ref.pdf)
- [24] D. Rudolf et al., "CSP: A Multifaceted Hybrid System for Space Computing," *Proc. of 28th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, August 2-7, 2014.
- [25] *World map of the AP-8 MAX Integral Proton Flux*. [Online]. Available: <https://www.spennis.oma.be/help/background/traprad/traprad.html>
- [26] J. McDougall. (2013, Feb. 14). *Simple AMP Running Linux and Bare-Metal System on Both Zynq SoC Processors - XAPP1078*. [Online]. Available: [http://www.xilinx.com/support/documentation/application\\_notes/xapp1078-amp-linux-bare-metal.pdf](http://www.xilinx.com/support/documentation/application_notes/xapp1078-amp-linux-bare-metal.pdf)
- [27] J. McDougall. (2014, Jan. 24). *Simple AMP: Bare-Metal System Running on Both Cortex-A9 Processors - XAPP1079*. [Online]. Available: <http://www.xilinx.com/>

support/documentation/application\_notes/xapp1079-amp-bare-metal-cortex-a9.pdf

- [28] J. McDougall. (2014, Jan. 24). *Simple AMP: Zynq SoC Cortex-A9 Bare-Metal System with MicroBlaze Processor – XAPP1093*. [Online]. Available: [http://www.xilinx.com/support/documentation/application\\_notes/xapp1093-amp-bare-metal-microblaze.pdf](http://www.xilinx.com/support/documentation/application_notes/xapp1093-amp-bare-metal-microblaze.pdf)
- [29] Xilinx 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide for HDL Designs. [Online]. Available: [http://www.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_7/7series\\_hdl.pdf](http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/7series_hdl.pdf)
- [30] A. Stoddard, "Configuration Scrubbing Architectures for High Reliability FPGA Systems," BYU ScholarsArchive All Theses and Dissertations Paper 5704, 2015.
- [31] C. Wilson, A. George, and B. Klamm "A Methodology for Estimating Reliability of SmallSat Computers in Radiation Environments," *2016 IEEE Aerospace Conf.*, Big Sky, MT, Mar 5 - 12, 2016.
- [32] B. Sierawski and M. Mendenhall. CRÈME. [Online]. Available: <https://creme.isde.vanderbilt.edu/>
- [33] T. M. Lovelly, K. Cheng, W. Garcia, and A. D. George, "Comparative Analysis of Present and Future Space Processors with Device Metrics," *Proc. of Military and Aerospace Programmable-Logic Devices Conference (MAPLD)*, San Diego, CA, May 19 - 22, 2014.
- [34] A. Wilson, A. Wilson, and M. Wirthlin, "Neutron Testing of the Linux Kernel Operating on the Zynq SOC," *Int. Workshop on FPGAs for Aerospace Applicat.*, 2015.

## BIOGRAPHY



**Christopher Wilson** is a doctoral candidate in ECE at the University of Florida on assignment at the University of Pittsburgh. He is a research assistant and team leader of the group on hybrid space computing in CHREC. His research interests include fault-tolerant techniques on hybrid architectures and radiation effects on commercial devices.



**Sebastian Sabogal** is a doctoral student in ECE at the University of Pittsburgh. He is a research assistant of the group on hybrid space computing in the new lead site of CHREC at Pitt. His research interests include reconfigurable and fault-tolerant computing.



**Alan George** is Professor of ECE at the University of Florida, where he serves as Director of the NSF Center for High-performance Reconfigurable Computing (CHREC). He received the B.S. degree in CS and M.S. in ECE from the University of Central Florida, and the Ph.D. in CS from the Florida State University. Dr. George's research interests focus upon high-performance architectures, networks, systems, services, and apps for reconfigurable, parallel, distributed, and fault-tolerant computing. He is a Fellow of the IEEE. In January 2017, the lead site of CHREC, his students, and he will move to the University of Pittsburgh, where Dr. George will serve as Ruth and Howard Mickle Endowed Chair and the Department Chair of Electrical and Computer Engineering in the Swanson School of Engineering at Pitt.



**Ann Gordon-Ross** received her B.S. and Ph.D. degrees in Computer Science and Engineering from the University of California, Riverside (USA) in 2000 and 2007, respectively. She is currently an Associate Professor of ECE at the University of Florida and is a faculty member in the NSF Center for High-Performance Reconfigurable Computing (CHREC). Her research interests include embedded systems, computer architecture, low-power design, reconfigurable computing, dynamic optimizations, hardware design, real-time systems, and multi-core platforms.