Radiation Hardening by Software Techniques on FPGAs: Flight Experiment Evaluation and Results

Andrew G. Schmidt and Matthew French
Information Sciences Institute
University of Southern California
{aschmidt,mfrench}@isi.edu

Thomas Flatley
Science Data Processing Branch
NASA Goddard Space Flight Center
thomas.p.flatley@nasa.gov

***Abstract*— We present our work on implementing Radiation Hardening by Software (RHBSW) techniques on the Xilinx Virtex5 FPGAs PowerPC 440 processors on the SpaceCube 2.0 platform. The techniques have been matured and tested through simulation modeling, fault emulation, laser fault injection and now in a flight experiment, as part of the Space Test Program-Houston 4-ISS SpaceCube Experiment 2.0 (STP-H4-ISE 2.0). This work leverages concepts such as heartbeat monitoring, control flow assertions, and checkpointing, commonly used in the High Performance Computing industry, and adapts them for use in remote sensing embedded systems. These techniques are extremely low overhead (typically <1.3%), enabling a 3.3x gain in processing performance as compared to the equivalent traditionally radiation hardened processor. The recently concluded STP-H4 flight experiment was an opportunity to upgrade the RHBSW techniques for the Virtex5 FPGA and demonstrate them on-board the ISS to achieve TRL 7.**

This work details the implementation of the RHBSW techniques, that were previously developed for the Virtex4-based SpaceCube 1.0 platform, on the Virtex5-based SpaceCube 2.0 flight platform. The evaluation spans the development and integration with flight software, remotely uploading the new experiment to the ISS SpaceCube 2.0 platform, and conducting the experiment continuously for 16 days before the platform was decommissioned. The experiment was conducted on two PowerPCs embedded within the Virtex5 FPGA devices and the experiment collected 19,400 checkpoints, processed 253,482 status messages, and incurred 0 faults. These results are highly encouraging and future work is looking into longer duration testing as part of the STP-H5 flight experiment.

TABLE OF CONTENTS

1. INTRODUCTION

As the development of space-based instrument capabilities grows, so too does the resulting data produced needing to be processed, driving the demand for higher performance processing. While radiation hardened and tolerant processors exist, they are quickly outpaced by commodity processors. The SpaceCube [1] project has been investigating how to leverage commodity processing technology in space-based systems, yet retain high levels of fault tolerance, for over a decade. Utilizing FPGAs with embedded hardcore processors, the SpaceCube technology is aiming to provide 10×-100× improvements in on-board computing power while lowering the relative power consumption and cost.

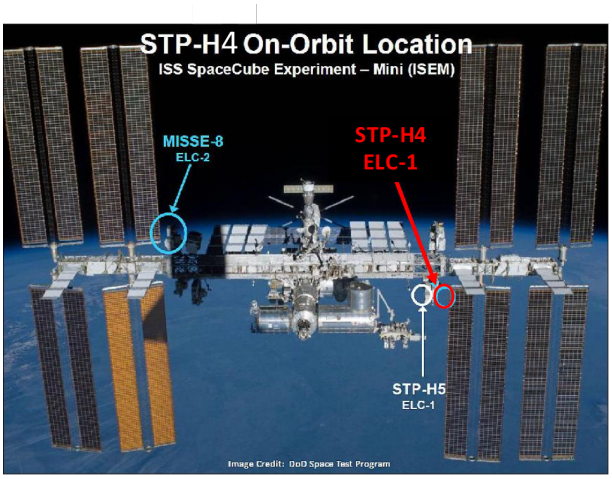

Figure 1. Space Test Program-Houston 4's ISS SpaceCube Experiment 2.0 on the International Space Station

We present our work on the implementation of Radiation Hardening By Software (RHBSW) techniques on the embedded PowerPC 440 hardcore processors found in the SpaceCube 2.0's Xilinx Virtex5 FPGA. Unlike traditional fault mitigation techniques used on FPGAs, the embedded processor's internal state cannot be readback and corrected through conventional bitstream scrubbing techniques. Furthermore, as there are only two PowerPC processors on each FPGA, implementing Triple Module Redundancy (TMR) is not possible. Instead, we leverage a number of fault mitigation techniques common in the high performance computing community to instead leverage the performance of both processors in parallel while only adding on average ≈1.3% overhead.

This paper covers our maturation of the RHBSW techniques and migrating from the SpaceCube 1.0's Xilinx Virtex4 PowerPC 405 processor [2] to the SpaceCube 2.0's Xilinx Virtex5 PowerPC 440 processor. Furthermore, we have uploaded and run the experiments on the Space Test Program-Houston 4-ISS SpaceCube Experiment 2.0 (STP-H4-ISE 2.0), which can be seen in Figure 1. We report on our experiences as we have moved through fault simulation and emulation to laser testing, and now a brief flight experiment. Prior to the decommissioning of the STP-H4 ISE 2.0 experiment, we were successfully able to upload and run a synthetic benchmark application based on common earth science applications on two PowerPC 440 processors while being monitored for single event upsets (SEUs) by a newly designed third Xilinx MicroBlaze soft core processor-based system-on-chip. Over

978-1-5090-1613-6/17/31.00

the course of the 16 day experiment we collected 19,400 checkpoints, processed 253,482 status messages, and incurred 0 faults. These results are highly encouraging and future work is looking into longer duration testing as part of the STP-H5 flight experiment.

The rest of the paper is organized as follows. In Section 2 we provide a background and related works for the SpaceCube system and RHBSW fault mitigation techniques. Then we present our design and implementation in Section 3. The experimental setup and evaluation of our results is presented in Section 4 followed by a conclusion and future work in Section 5.

2. Background and Related Work

While FPGAs are increasingly becoming more common place in embedded systems and more recently high performance computing systems [3], these systems are operating in a much more controlled environment. FPGAs operating in a space environment are susceptible to radiation effects, such as Single Event Upsets (SEUs). The SpaceCube platform does include the use of radiation tolerant FPGAs, but also includes commodity FPGA devices. Using commodity devices requires the use of fault mitigation techniques and this section provide a background and related work on the different techniques and capabilities used in the existing platforms.

Fault Mitigation and Tolerance on FPGAs

While there have been fault tolerance and mitigation technique studies for space-based systems [4], these are largely focused on the susceptibility of the FPGA fabric itself to SEUs. With FPGA devices the configuration and logic plans present the greatest challenge for fault mitigation; however, there exist different techniques to detect, correct, and recover for such faults. Bitstream scrubbing [5] is the most common technique to readback and correct bit-flips and in most cases can be done without impacting the running design, although, there remains a possibility, based on the periodicity of the scrubbing, where a fault may still occur. To account for these errors, triple modular redundancy (TMR) [6] is often implemented to immediately mask faults and configuration errors until scrubbing can correct them. The design is augmented by triplicating the computational element and inserting a voter to determine if an error has occurred in one of the three identical computations. A limitation of this approach is the additional resources required when triplicating the design.

Unfortunately, bitstream scrubbing and TMR do not work for the embedded PowerPC 440 processors as their internal registers, such as cache contents and general purpose registers, are not readable by the Internal Access Configuration Port (ICAP). Furthermore, there are only two PowerPC 440 devices embedded into each of the Virtex5 FX130 FPGAs, which means triplication is not possible on a single FPGA device. These same motivations led the development of the RHBSW techniques applied to the SpaceCube 1.0's PowerPC 405 and now drive the migration for the PowerPC 440. This effort looks at the migration effort and the results of a successful flight experiment on the STP-H4 ISE 2.0 platform.

SpaceCube Family

SpaceCube is a cross-cutting, in-flight reconfigurable Field Programmable Gate Array (FPGA) based on-board hybrid science data processing system developed at the NASA Goddard Space Flight Center (GSFC) [1]. The goal of the SpaceCube program is to provide $10\times$ – $100\times$ improvements in on-board computing power while lowering relative power consumption and cost. The SpaceCube design strategy incorporates commercial radiation-tolerant Xilinx Virtex FPGA technology and couples it with an integrated upset detection and correction architecture to provide reliable "order of magnitude" improvements in computing power over traditional fully radiation-hardened flight systems.

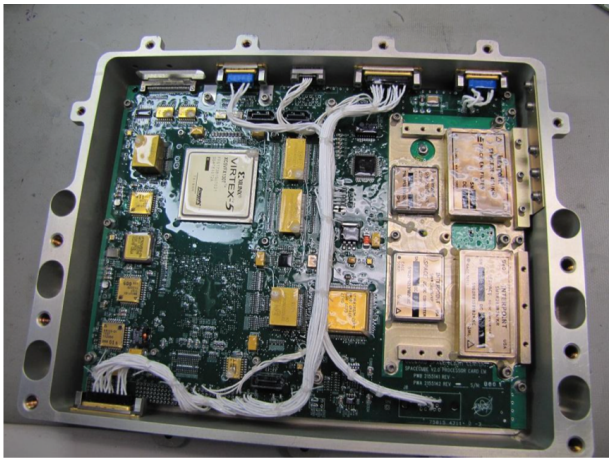

Figure 2. SpaceCube 2.0 Engineering Model with Xilinx Virtex5 FPGAs

SpaceCube 1.0—The preliminary Radiation Hardening By Software was developed for the SpaceCube 1.0's Virtex4 FX60 FPGAs which included two PowerPC 405 embedded processors. Starting in 2006 at GSFC, the SpaceCube system developed after showcasing the computational power and inherent reconfigurable advantages over typical space processors. The 1.0 platform was developed for the main avionics for the experimental payload called the Relative Navigation Sensors (RNS) [7]. A 1.0 platform was also added to the International Space Station Experiment (MISSE-7) to study the long term effects of radiation [8].

SpaceCube 2.0—The success of SpaceCube 1.0 led to the development of SpaceCube 1.5, a Virtex5 based FPGA, but remained backwards compatible with the SpaceCube 1.0 form factor. SpaceCube 2.0 was funded by NASA's Earth Science Technology Office (ESTO) and Satellite Servicing Capabilities Office (SSCO). The 2.0 system leverages seven years of board design, avionics systems design, and space flight application experiences. Figure 2 shows the engineering model of the 2.0 system.

ISS SpaceCube Experiment 2.0

The ISE 2.0 experiment, which is a follow-on SpaceCube v1.0 payload on MISSE-7, is installed on the DoD Space Test Program Houston 4 (STP-H4) payload that was activated on the ISS in August 2013. The purpose of the STP-H4 ISS SpaceCube Experiment 2.0 is to demonstrate NASA Goddard Space Flight Center (GSFC) SpaceCube 2.0 advanced hybrid on-board science data processor technology in low Earth orbit. The ISE 2.0 consists of a Power Unit, SpaceCube v2.0 Engineering Model, a set of Earth-viewing high definition cameras, and instrumentation to detect and measure terrestrial gamma-ray flashes from lightning, which serve as data sources for the on-board processing demonstration.

The SpaceCube v2.0 EM is used to control the cameras and FireStation instrument. The communication link between ISE 2.0 and the ISS goes through the main avionics of STP-H4, which is a SpaceCube v1.0 system. The 2.0 system has the following key technology elements:

- research critical to enable "next generation" missions by providing the on-board computing power necessary to handle future ultra-high data rate instruments and advanced mission applications.
- successful demonstration of the ISE 2.0 experiment will include the processing of high definition Earth imagery and potentially unprecedented insight into the recently discovered phenomena of terrestrial gamma ray flashes.
- successful completion of the ISE 2.0 processing experiment will significantly increase the Technology Readiness Level of the system and significantly reduce the risk for future missions that wish to adopt this technology.
- successful completion of the ISE 2.0 gamma ray experiment may provide ground-breaking scientific discoveries in the fields of Heliophysics and Earth Science.

3. Design and Implementation

The RHBSW techniques developed and implemented include checkpoint and restart, heartbeat monitoring, control flow assertions, and watchdog timers. These various techniques are implemented in software as supported library function calls and *pragmas* that can be automatically inserted in the user's application. Furthermore, the techniques are software programmable and can be adapted for the application to provide a balance between software fault tolerance support and performance overhead. This section briefly describes the Xilinx implemented PowerPC 440 processor and then provides an overview of the software fault tolerance techniques implemented on these processors. These techniques are then applied on the SpaceCube 2.0 as part of the STP-H4 ISE 2.0 mission. The [2] provides a full description of originally developed software fault tolerance techniques details.

Xilinx Virtex5's PowerPC 440 Processor Details

The initial implementation of the software fault tolerance techniques were implemented on the Xilinx Virtex4's PowerPC 405 embedded processor [2, 9] on the SpaceCube 1.0 architecture [1]. In this work, the techniques have been migrated to the PowerPC 440 in order to achieve a higher processor performance efficiency and access to more FPGA resources. In order to migrate the fault mitigation strategies and understand fault injection results, it is important to understand the variant of the PowerPC 440 in the Virtex5 FPGA from Xilinx. The PowerPC 440 devices are embedded within the FPGA and are 32-bit RISC, Harvard Architecture processor. A block diagram of the PowerPC is shown below in Figure 3. The caches are each 32KB, 64-way set-associative, with 32-byte cache lines. The Memory Management Unit (MMU) is software controlled, but is, generally, only used by an operating system.

Table 1 lists the performance efficiency comparisons of standard RadHard devices with the SpaceCube 1.0 and 2.0 systems. The Virtex5 PowerPC 440 device by the SpaceCube 2.0 is 25× more power efficient than the RAD750 device. By extending the RHBSW techniques to the PowerPC 440, this work brings an order of magnitude of processing capabilities to systems that still need fault mitigation and can enable applications to run within the SpaceCube platform to further advance the scientific capabilities for earth science and other

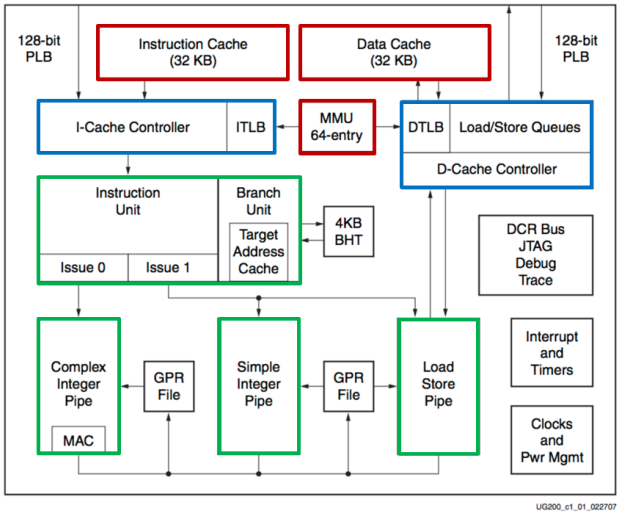

Figure 3. Block diagram of Xilinx Virtex5 PowerPC 440 processor

Table 1. Processor performance and power comparisons with SpaceCube 2.0 PowerPC 440 based FPGA

Processor	MIPS	Power	MIPS/W
MIL-STD-1750A	3	15 W	0.2
RAD6000	35	10-20 W	2.33
RAD750	300	10-20 W	20
SPARC V8	86	1 W	86
LEON 3FT	60	3-5 W	15
GSFC SpaceCube 1.0	3000	5-15 W	400
GSFC SpaceCube 2.0	**5000**	**10-20 W**	**500**

missions. These performance opportunities set a firm baseline for the allowed overhead and the expected performance capabilities that the RHBSW techniques must provide.

Software Fault Tolerance Techniques

This section covers the designed and implemented techniques, which can also be seen in Figure 4. These include checkpoint an restart, heartbeat monitoring, control flow assertions, and watchdog timers. In addition, we detail how the techniques were updated to support the SpaceCube 2.0 and the SPT-H4 flight experiment.

Checkpoint and Restart—This work extends the checkpoint and restart techniques common to the high performance com-

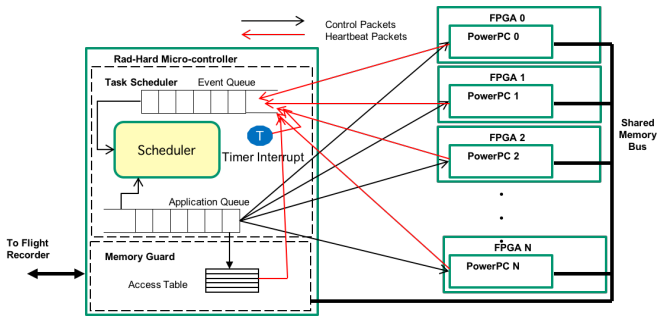

Figure 4. Diagram of the Radiation Hardening By Software techniques running on the SpaceCube 2.0's Virtex5 FPGAs

puting community and implemented in the initial PowerPC 405 designs by increasing the fidelity of the checkpoints, the number of saved checkpoints, and provides a redundant retry mechanism to restart the processor in the event a fault persists through the checkpoint and restart procedure. A checkpoint library has been developed that allows the application to capture its current running state. A checkpoint consists of pausing the running application, capturing the memory segments of the processor's registers, and writing the data to storage. The flight experiment downlinks these checkpoints in order to analyze the data and determine if there were any data corruptions and validate the processor proceeds correctly in the event of a fault. Checkpoint sizes and frequency were selected to not overwhelm the downlink bandwidth, yet provide sufficient state to allow the processor to recover when a restart is required. To more thoroughly test the robustness, a checkpoint of 60 seconds was selected with a size of 2 KB. Each checkpoint was stored in on-chip BRAM, although more recent designs support off-chip DRAM to dramatically increase the size and number of checkpoints supported during the applications run-time. A library has been extended to provide configurable and paramertizable support for an application and its developer to use checkpoint and restart within the RHBSW framework. The larger on-chip memory sizes on the Virtex5 FPGA enable larger checkpoints and for multiple checkpoints to be stored as compared to the original Virtex4 implementation.

Heartbeat Monitoring—Each processor is responsible for sending heartbeat messages to the controller processor. The frequency and size of the messages can be tuned to maintain a low overhead. In our experiments the message size is 16-bytes with the capability of enqueuing 1024 heartbeat messages. A message is sent at least every 2 seconds, although more status messages can be enqueued based on the activity of the processor. The processor sends heartbeat messages to the controller to indicate status as it proceeds through the computation. The messages provide a log of the processors experimental state and identify if a particular fault was detected by the processor, what the correcting action was, and allows the controller verify the processor is making progress in the computation. In the event a fault occurs that makes the processor unresponsive or no longer sends heartbeats, the controller can determine the appropriate action to take in order to get the processor back in a functioning state. For the purposes of the flight experiment, these messages from both PowerPC 440 processors are aggregated and downlinked daily for analysis. In this work we extended the messaging framework to support a hybrid multi-PowerPC and MicroBlaze infrastructure, detailed in Section 3. Previously a control/DUT interface was developed where one PowerPC 405 acted as the controlling device and the second PowerPC 405 was the device under test. By adding a scalable multiprocessor messaging system, the technique can more reliably be implemented and tested on the PowerPC 440 and future FPGA devices, such as the forthcoming Xilinx UltraScale+ Zynq Quad ARM A53-based FPGA.

Control Flow Assertions—Another technique is the use of control flow assertions to enable the source code to perform a self-check while running to determine if an upset has corrupted the program counter, a loop counter, etc [10]. With this method, each PowerPC 440 is capable of evaluating if the assigned computation is progressing as expected. If the processing element detects a control flow fault, the failure status is communicated to the control processor using heartbeats. The assertions are inserted at the source code level through the use of *pragmas* with a developed utility that checks for loop iterations, if-statements, and other user defined locations. A second utility transforms these statements into standard C code with control flow variables which are checked during the run-time execution of the application. For this work two forms of control flow assertions have been implemented. The first ensures the program's execution is progressing as expected. An assertion is raised if any of the control flow points are skipped or if the same point is crossed consecutively. The second form of assertion monitors the program to verify that the application is moving through the different control flow points. Status messages are sent via the heartbeat monitoring back to the controller processor which can validate the status and respond accordingly. For this effort no modifications to the tools or source-to-source compiling is required, except to leverage the PowerPC 440's cross-compiler. The original application's assertions used throughout the simulation, emulation, and laser testing experiments (described in Section 3) have been used in the flight experiment to ensure continuity and identify if the control flow logic can be adequately covered using these techniques.

Watchdog Timers—Each PowerPC 440 has a built in watchdog timer which can be enabled and cleared through software function calls to allow each processor itself to restart. The period and behavior of the watchdog timer can be controlled from software. For these experiments the watchdog timer is set to timeout after 5 seconds of not being cleared. Each time a heartbeat message is generated the watchdog timer is cleared. If the watchdog interrupt is disabled, or otherwise interfered with, the watchdog timer will reset the PowerPC. An application event heartbeat notifies the controller when the processor is reset, so an error can be logged. The original RHBSW techniques leveraged soft IP timers and interrupt controllers in addition to its watchdog timer. This work eliminates the dependence on the timer and interrupt controller for the watchdog purpose and updates the library to utilize the PowerPC 440's watchdog timer libraries developed and provided as part of the board support package by Xilinx.

Fault Testing

The RHBSW techniques have been developed and evaluated across a number of different platforms and fault injection studies. This includes the preliminary analysis as part of the Xilinx Virtex4 ML410 development board, the SpaceCube 1.0 breadboard, the Xilinx Virtex5 ML510 development board, and now the SpaceCube 2.0 flight hardware as part of the STP-H4 ISE 2.0 experiment. This work is the culmination of the experiences and technology developed throughout the past fault testing experiments.

Fault Emulation—The initial efforts to study and evaluate SEUs and the PowerPC a fault emulator was developed to simulate and then emulate, in real-time, bit-flips in the PowerPC's registers and cache [11]. This allowed us to enhance the software fault tolerance and provide a baseline to suspectability of registers and cache to upsets and resets. From this early work we have adapted the mitigation techniques to focus on the registers that have demonstrated a higher sensitivity to faults, such as register *r14* which in our experiments has led to a failure $\approx$22.5% of the time a bit-flip occurs that result in $\approx$25% of the data errors in our experiments [2].

Laser Testing—The fault emulation environment was then expanded to include laser testing [12], where a laser allows for precise control of the injection target to the micron level,

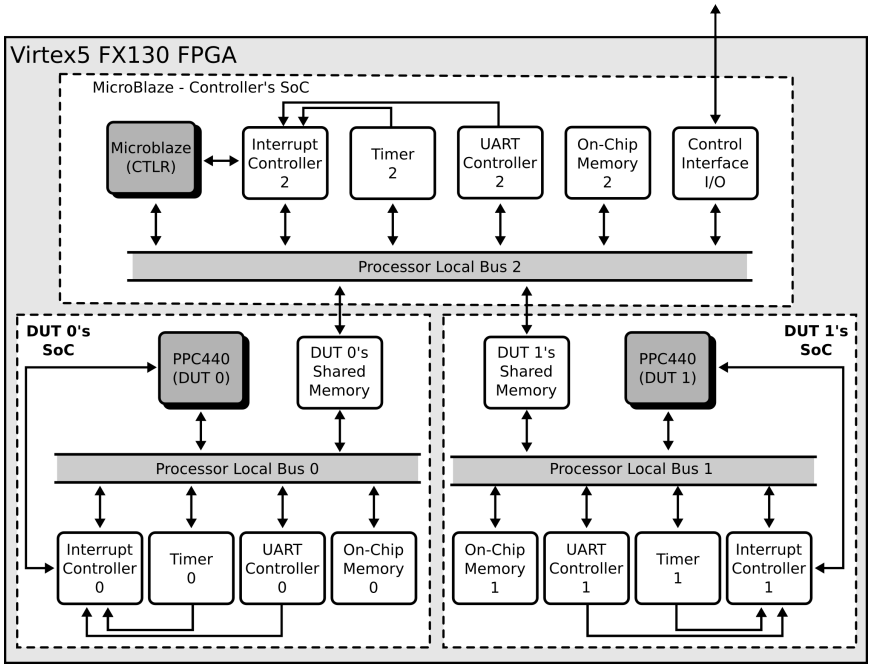

Figure 5. Block diagram of flight experiment's integrated three system-on-chips consisting of two PowerPC 440 processors as the devices under test and one MicroBlaze processor as the test controller and monitor

allowing injection of a single pulse at a specified x y coordinate. Fault emulation (injecting faults into an actual machine such as a prototype) provides an inexpensive solution to long running fault injection campaigns, enabling users to collect thousands of injections continuously over a period of days or weeks. The laser test further affirmed the effectiveness of the developed software fault mitigation techniques.

Flight Experiment

Ultimately, the focus of this work is to provide software fault mitigation and demonstrate its capabilities as part of a flight experiment. Our work initially was focused on porting the RHBSW techniques from the SpaceCube 1.0 Virtex4 platform to the SpaceCube 2.0 Virtex5 platform. Once the porting was completed, we were presented with an opportunity to run on one FPGA on the SpaceCube 2.0's platform as part of the STP-H4 ISE 2.0 experiment became available. The Goddard Space Flight Center team provided the necessary infrastructure to first test the experiment on a ground system with a simulated ISE interface in order to first test the command and control between the primary FPGA of the SpaceCube 2.0 and the Device Under Test FPGA.

Initially, a single PowerPC 440 was going to be made available to our experiments, meaning the control and monitoring functionality would be handled outside of the FPGA. However, during the course of the project the entire FPGA was made available to our experiment, meaning both PowerPC 440s would be allowed to run our RHBSW tests. As a result, a third system-on-chip was developed to act as the controller and handle the exchanges between the main FPGA and our experiment's FPGA within the SpaceCube 2.0 system. Since both PowerPC 440 hard processors were configured to run the fault mitigation experiments, a soft processor and system-on-chip was implemented for this effort, seen in Figure 5. While long-term plans are for the soft processor to exist in a radiation hardened device, this experiment chose to use a Xilinx MicroBlaze soft processor and system-on-chip in order to keep the experiment within a single FPGA and not require modifications to the remainder of the SpaceCube 2.0 system.

Finally, in order to evaluate the running system as part of the flight experiment, the GSFC team developed an efficient FPGA bitstream uploading procedure to reconfigure the FPGA3 in their SpaceCube 2.0 system with out synthesized and implemented design. While the technique of uploading and reconfiguring an FPGA in space has already been demonstrated, the complication in this case was how the FPGA3 configuration memory was stored. In order to assure the full bitstream was uploaded without any transmission faults, the compressed bitstream was split into 106-byte segments due to upload bandwidth limitations. The entire upload procedure took ≈8 hours, run overnight, and then verified using CRC checksums. Once the upload completed of the ≈360 KB compressed bitstream, the GSFC team uncompressed and deployed the bitstream into the SpaceCube 2.0's nonvolatile memory and initiated a reconfiguration of FPGA3 to start our experiment.

4. EVALUATION

The flight experiment consists of running a synthetic benchmark application on the two embedded PowerPC 440's on one FPGA on the SpaceCube 2.0 system and downlinking the results over the course of the experiment. A soft core MicroBlaze processor is used to monitor the two PowerPCs and provide control and status message handling to the rest of the SpaceCube 2.0 platform. This section first describes the benchmark and how it is used within the experiment, the overall setup of the RHBSW experiment within the SpaceCube 2.0 framework, and the collection of results and analysis of the experiment.

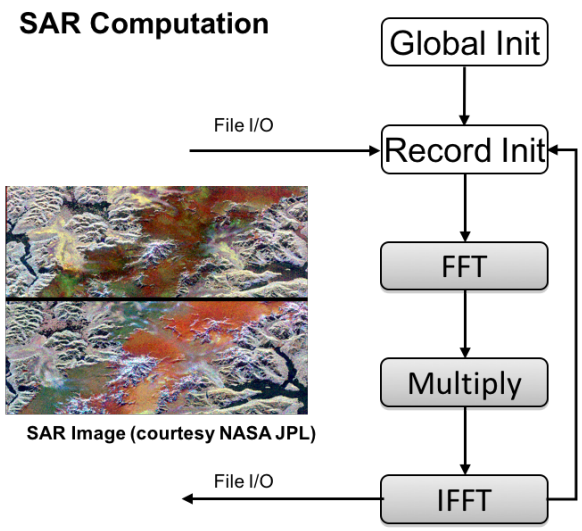

Figure 6. Synthetic Aperture Radar benchmarks application overview

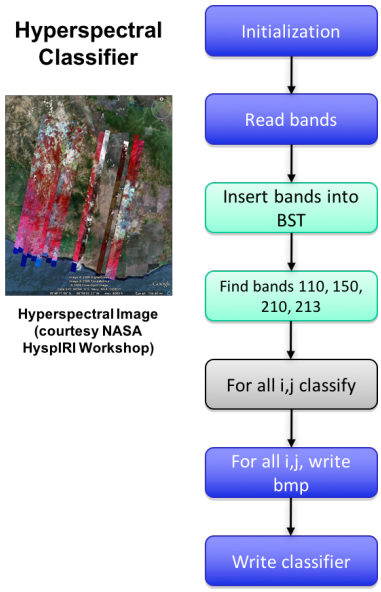

Figure 7. hyperspectral thermal image classification benchmarks application overview

Synthetic Benchmark

To demonstrate the RHBSW techniques and determine if any faults have occurred, we have combined the major elements of a Synthetic Aperture Radar application and hyperspectral image processing classifier to provide a controlled, yet representative example of computation to run on the Xilinx Virtex5 PowerPC 440 processors. From SAR we implement the complex multiple and FFT kernels with data sizes small enough to operate in block RAM, yet large enough to force cache turn over. The use of smaller data sets allow for quicker results checking and analysis so the effective compute vs. error checking ratio is >90%. The computation flow of the SAR application is illustrated in Figure 6. The hyperspectral thermal image classifier application across different bands of the input image (110, 150, 210, and 213), shown in Figure 7. The hyperspectral code was part of a larger Autonomous Onboard Processing for Sensor Systems (AOPSS) [13] where the FPGA could reconfigure and run different classifiers based on the current location and needs of the application. For this work both the SAR and hyperspectral classifier are running in software on the Xilinx Virtex5's PowerPC 440 processors. In this synthetic benchmark all of the general purpose registers are used along with both instruction and data caches. Most special purpose registers are not referenced directly within the application; however, manipulating the SPRs at runtime often results in undesirable side effects. For example, disabling/enabling cacheable regions, debug modes, and interrupts. No operating system was used in our tests.

Experimental Setup

The synthetic benchmark application repeatedly performs the 1-dimensional FFTs and complex multiplication followed by thresholding in order to mimic both SAR and hyperspectral imaging, shown in Figure 8. At system startup a golden output is calculated that is used to verify results during the subsequent loop iterations of the synthetic benchmark. During the experiment if a data error or control flow assertion is detected, the PowerPC logs the error MicroBlaze controller, resets itself, recomputes golden outputs, and continues the experiment. To check for silent data corruptions each time the application records a new checkpoint, the data is packetized by the MicroBlaze and sent to the control FPGA for downlink. This allows us to analyze each checkpoint and replay any portions of the applications on our ground testbed to trace back the upsets.

These same procedures were in place during our simulation, emulation, and laser testing so as to ensure uniform testing and evaluation of the RHBSW techniques. The main difference with respect to the flight experiment was the ability to use both PowerPCs as devices under test and leverage the unused FPGA fabric to implement a MicroBlaze controller to monitor the experiment. While a single upset would only interfere with one of the two PowerPCs, having both operational at the same time throughout the duration of the experiment provided a larger cross section of sensitive bits within the two processors to be evaluated.

The experiment relies on redundant controls and checks to make sure all faults are detected, recorded, and downlinked for analysis. The two PowerPC 440s send messages to the MicroBlaze. In the event the PowerPCs fail, this is exactly the data these experiments are interested in collecting, and ideally correcting. If the MicroBlaze fails or an upset disables the FPGA a watchdog timeout to the control FPGA within the SpaceCube 2.0 platform will log the event and automatically reconfigure the FPGA. As a result the MicroBlaze and PowerPCs restart their experiment counters. When this data is downlinked our offline analysis tools detect the reset. Due to the SpaceCube 2.0's existing FPGA fault mitigation techniques, such as bitstream scrubbing, the analysis of these failures falls outside of the scope of this flight experiment. As explained in Section 2, our RHBSW techniques are designed to complement existing FPGA fault mitigation techniques by providing coverage to the PowerPCs which are not covered through bitstream scrubbing.

Flight Test Results

The initial flight experiment was an add-on to the original STP-H4 ISE 2.0 mission as the necessary resources to run our supplemental experiment had become available towards the end of the mission. As a result, it was understood that our experimental run-time would be on the order of a

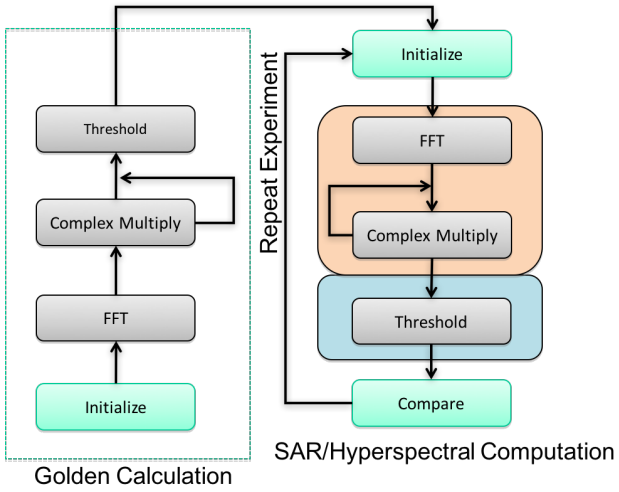

Figure 8. Computational overview of experimental application running on PowerPCs during the flight experiment

few months. Ultimately, due to a power switching unit trip on the ISS that removed power to STP-H4, our experiment was able to run for 16 days. During the course of the 16 days the experiment collected 19,400 checkpoints, processed 253,482 status messages, and incurred 0 faults. Each of the systems performed as expected, and no silent data corruptions were detected based on offline analysis of the checkpointed data. The RHBSW techniques provided a quick and effective mechanism to add fault mitigation to our synthetic benchmark application while only adding 1.3% overhead to the system. This flight experiment was quite a fortunate opportunity to further demonstrate the RHBSW techniques and pave the way for a much longer duration experiment on a possible SPT-H5 mission.

5. Future Work

Moving forward this work is continuing to enhance and develop more sophisticated RHBSW techniques. The original development of this work and the migration of the RHBSW techniques from the Virtex4 PowerPC 405 to the Virtex5 PowerPC 440 were part of the forthcoming STP-H5 SpaceCube-Mini flight experiment. The research provides flight validation of the STP-H5 SpaceCube-Mini and advanced on-board processing capabilities for Earth Science/atmospheric chemistry. These validated capabilities increase the TRL of these technologies from TRL 6 to TRL 8 and reduce the programmatic risk of using these technologies on future missions. Future Earth Science, Space Science, Exploration and Satellite Servicing missions are able to implement this enabling computing technology to perform complex on-board functions that are previously limited to ground based systems, such as on-board product generation, data reduction, calibration, classification, event/feature detection, data mining and real-time autonomous operations. Our goal is to run these same experiments on the STP-H5 SpaceCube-Mini for a longer duration flight experiment.

In addition to the SpaceCube-Mini, work is underway to determine future SpaceCube architecture, called SpaceCubeX. These systems will likely migrate to the ARM A9 and A53-based processing subsystems. While many of our RHBSW techniques will rapidly port over to these new devices, we are anticipating the need to run fault mitigation experiments. As a result, an analysis of the simulation, emulation, and flight infrastructure for the devices is required. Providing a more broadly applicable and adaptable RHBSW framework is a long-term goal and these flight experiments provide invaluable data to help us all move forward.

References

[1] T. Flatley, "Spacecube: A family of recon gurable hybrid on-board science data processors," in *NASAIESA Conference on Adaptive Hardware and Systems*, June 2012.

[2] M. Bucciero, J. P. Walters, and M. French, "Software fault tolerance methodology and testing for the embedded powerpc," in *Proceedings of the 2011 IEEE Aerospace Conference*, march 2011, pp. 1–9.

[3] A. P. et al., "A reconfigurable fabric for accelerating large-scale datacenter services," in *41st Annual International Symposium on Computer Architecture (ISCA)*, June 2014.

[4] H. M. Quinn, P. S. Graham, M. J. Wirthlin, B. Pratt, K. S. Morgan, M. P. Caffrey, and J. B. Krone, "A test methodology for determining space readiness of xilinx sram-based fpga devices and designs," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 10, pp. 3380–3395, Oct 2009.

[5] B. Bridgford, C. Carmichael, and C. W. Tseng, "Single-event upset mitigation selection guide (xapp987)," march 2008.

[6] C. Carmichael, "Triple module redundancy design techniques for virtex fpgas (xapp197)," july 2006.

[7] D. Petrick, "Application of spacecube in a space flight system," in *Military and Aerospace Programmable Logics Devices Conference*, 2009.

[8] "Materials international space station experiment - 7," http://www.nasa.gov/missionpages/station/research/experiments/653.htrnl.

[9] G. Allen, G. Swift, and G. Miller, "Upset characterization and test methodology of the PowerPC405 hard-core processor embedded in xilinx field programmable gate arrays," in *Radiation Effects Data Workshop, 2007 IEEE*, july 2007, pp. 167 –171.

[10] R. Vemu and J. A. Abraham, "Ceda: control-flow error detection through assertions," in *12th IEEE International On-Line Testing Symposium (IOLTS'06)*, July 2006.

[11] M. Bucciero, J. P. Walters, R. Moussalli, S. Gao, and M. French, "The PowerPC405 memory sentinel and injection system," in *Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines*, ser. FCCM '11. IEEE Computer Society, 2011, pp. 154–161.

[12] J. P. Walters, K. M. Zick, and M. French, "A practical characterization of a nasa spacecube application through fault emulation and laser testing," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2013, pp. 1–8.

[13] M. French, "Autonomous, on-board processing for sensor systems: Initial fault tolerance and autonomy results," in *Earth Science Technology Forum*, 2011.

BIOGRAPHY

Andrew G. Schmidt *is a Research Lead and Senior Computer Scientist at the University of Southern California's Information Sciences Institute. He is the technical lead of the SpaceCubeX project investigating heterogeneous computing platforms for next generation earth science missions. He received his Ph.D. in electrical engineering from the University of North Carolina at Charlotte in 2011 investigating efficient utilization of heterogeneous compute resources through static analysis and runtime performance monitoring. He received his M.S. and B.S. in computer engineering from the University of Kansas in 2007 and 2005. His research interests include heterogeneous and reconfigurable computing, fault tolerant and resilient computing, high performance computing, and embedded systems. Dr. Schmidt has over 30 publications in the areas of reconfigurable computing based technology and is a member of IEEE and ACM.*

Matthew French *is a Research Director in the Computational Systems and Technology group at USC/ISI where he facilitates large scale, cross-discipline research which spans the institute. In this capacity, Mr. French serves as the director of the SecUre and Robust Electronics (SURE) center at ISI, which conducts research and provides services in hardware trust, security, reliability, and resiliency. Mr. French also oversees the Reconfigurable Computing Group at ISI, which performs research in application mapping, hardware / software co-design, CAD tools, and front-end ASIC design. Mr. French has over 40 publications and 2 patents, and serves on the program committee for several conferences in his field. Mr. French holds M.E. and B.S. degrees in Electrical Engineering from Cornell University.*

Thomas Flatley *is a Computer Engineer at the NASA Goddard Space Flight Center, and is currently Branch Head of the Science Data Processing Branch. Mr. Flatley's current work includes the coordination of embedded science data processing technology development and hardware accelerated science data processing activities, serving as Principal Investigator on multiple flight processing experiments, with the primary goal of developing reconfigurable computing technology and hybrid systems for flight and ground science data processing applications. He is also a key member of the GSFC CubeSat/SmallSat technology working group, manages numerous collaborations with government, industry and academic partners, and serves as liaison between technology developers and end users in the science community. Mr. Flatley was named a Goddard Senior Fellow in 2016.*