Autonomous On-board Processing for Sensor Systems: Initial Fault Tolerance and Autonomy Results

Matthew French, John Paul Walters, Kenneth Zick
University of Southern California, Information Sciences Institute
3811 N. Fairfax Dr., Suite 200, Arlington, VA 22203
{mfrench, jwalters, kzick}@isi.edu

Abstract

By developing Radiation Hardening by Software (RHBSW) techniques leveraged from the High Performance Computing community, the A-OPSS project seeks to deliver radiation tolerant, high performance System on a Chip (SoC) processors. This SoC architecture is uniquely suited to both handle high performance signal processing tasks, as well as autonomous agent processing. This allows situational awareness to be developed in-situ, resulting in a 10-100x decrease in processing latency, which directly translates into more science experiments conducted per day and a more thorough, timely analysis of captured data.

This paper focuses on the second year's efforts which revolve around developing a fault emulator for the embedded PowerPC within Xilinx V4FX devices, validating the RHBSW techniques developed in the prior year, and projecting performance results on a representative autonomous Hyperspectral application.

1. Introduction

Over the past decade, the physics of CMOS at advanced process node feature sizes has entered a trade space where total ionizing dose (TID) is no longer much of a concern for modern devices in relatively low radiation environments such as Low Earth Orbit (LEO), however Singe Event Upsets (SEUs) have become more prevalent. Researchers have focused on both hardware and software techniques to mitigate radiation effects in SRAM-based Field Programmable Gate Arrays in order to address SEUs and make these devices radiation tolerant. This has proved to be a boon to the space community as these processors can deliver processing performance several generations ahead of traditional radiation hardened by process devices, such as anti-fuse FPGAs.

As technology has progressed, FPGAs have evolved from homogeneous sea of gates architectures, to heterogeneous system on a chip architectures containing multipliers, multi-gigabit transceivers, Ethernet cores, and even embedded PowerPC processors. The embedded PowerPC is especially attractive in that the traditional Radiation Hardening by Process (RHBP) community's development of RISC processors has traditionally has lagged the commercial market by 2-3 Moore's Law generations. Furthermore, the exponential cost trend of RHBP has stalled the release of a radiation hardened RISC processor for several years. As table 1 shows, the small embedded PowerPCs within an FPGA device can now yield several times more performance than the latest RHBP RISC processor. It should also be noted that the performance characteristics listed are for a single PowerPC core, and the FPGA devices often contain two cores per device.

Table 1 Performance Comparison of Space Processors

Processor	Mitigation Technique	Dhrystone MIPS
RAD600	RHBP	35
RAD750	RHBP	266
LEON3FT	RHBP	560
PowerPC 405	RHBSW	900
PowerPC 440	RHBSW	1300

The embedded PowerPC in particular adds significant benefit to the space community as now scientists, who may not be VHDL experts, have an easy migration path for their existing C programs to an embedded space platform. This enables a rapid application development cycle where core functionality is quickly achieved and then code migration to the FPGA fabric is performed gradually, targeting performance-critical functions. This development cycle is especially attractive to the space community as it allows a low-risk spiral development path that yields higher performance. Furthermore, a modern RISC processor enables autonomous applications as its architecture is better suited to handle the branching type operations common in artificial intelligence agents and the higher processing bandwidth is better suited to tackle real world complexities. It is also easy to build embedded space platforms with several PowerPCs and scale performance as 2 PowerPCs are available in a single FPGA, and most current space hardware contains several FPGAs.

The downside however is that the embedded PowerPCs are susceptible to SEUs and new fault mitigation techniques must be developed in order to make use of these attractive features. Traditional FPGA fault tolerance strategies can detect and correct errors within the FPGA's bitstream [1,2]; however, the bitstream does not contain the state of the embedded PowerPC cores, and consequently, errors within the PowerPC cannot be scrubbed. Additionally, there are two, not three,

PowerPCs per FPGA, so traditional Triple Modular Redundancy (TMR) techniques are awkward to apply.

In this paper we describe our recent progress on developing software-based fault tolerance strategies for PowerPC devices embedded within Xilinx Virtex 4 FX60 FPGAs. We provide a brief overview of our techniques, the fault emulator we developed, the results of our initial test campaign, and our progress in developing an autonomous Hyperspectral imaging application using A-OPSS technology.

2. Radiation Hardening by Software

Our work targets *scientific* applications operating on traditional space-based FPGA board-level architectures consisting of an FPGA and a radiation-hardened controller. These applications are more tolerant to data upsets and, to a limited extent, may trade reliability for increased performance in space. To that end, our primarily goal is to detect and correct control flow and other catastrophic errors that would otherwise hang or crash the embedded PowerPCs. We ignore small data errors that can be corrected in post-processing on the ground. We use heartbeat monitoring, control flow assertions, and watchdog timers to detect errors. To mitigate the detected errors, we implement a user-level checkpoint and rollback library. The goal of these techniques is to provide a flexible, low overhead approach as compared to TMR.

All of these techniques are designed to complement one another and to work in conjunction with a radiation-hardened controller. Heartbeats allow the radiation-hardened controller to monitor coarse-grained execution and status messages. Control flow assertions and watchdog timers are used by the executing PowerPC to ensure that execution continues in a predictable manner without skipping or repeating major code segments and to ensure that computational progress is being made. Finally, checkpointing and rollback are used by the executing PowerPC to periodically capture its state of execution. If a fault is found, the PowerPC may roll back to the most recent checkpoint before continuing computation. The ability of a PowerPC to restart from its most recent checkpoint avoids unnecessary wasted computation. For a more detailed description of the RHBSW fault tolerant techniques, please refer to [3, 4]

3. The Memory Sentinel and Injection System (MSIS)

Validating our RHBSW techniques requires robust testing using progressively more realistic test conditions: software emulation, radiation testing, and in-situ testing. While each of these steps provides increasing realism, the trade-off is that each step is also exponentially more costly and time consuming. So having a quick, realistic software injector is critical to refining early RHBSW approaches and making more efficient use of costly, sparse testing environments. While several groups have developed fault emulators for the FPGA fabric, the research community has made little progress on fault emulators for the embedded PowerPC. The most significant being the Simple Portable Fault Injector – PPC (SPFI-PPC). While the SPFI-PPC is a good first step, it lacks coverage of the caches and relies on GDB-based corruption over a JTAG communication interface, which in practice proved unreliable beyond tens of injections.

Due to these limitations, we developed a custom hardware/software-based fault injector named the Memory Sentinel and Injection System (MSIS) [5]. MSIS is a fault injector for the PowerPCs 405(s) in Xilinx Virtex-II Pro and Virtex-4 FX FPGAs which emulate faults by flipping bits on the executing hardware. The MSIS introduces software faults to an application by flipping bits in the processor general-purpose registers, special purpose registers, or the instruction or data caches.

Table 2: PowerPC 405 Sensitive Bits

Feature	Size
ICache	16KB+1408B tag + 64 control bits
DCache	16KB+1408B tag + 64 control bits
General Purpose Registers	32x32 bits
Special Purpose Registers	32x32 bits
Execution Pipeline	10x32 bits
ALU/MAC	1200 bits
Timers	3x64 bits
MMU	72x68 bits
Misc	1024 bits
Total	**292,820 bits**

When a fault is injected, its details are logged so post-injection analysis can be performed to determine the cause of a failure.

The MSIS uses a split software/hardware implementation. The hardware components, termed HW-MSIS in Figure 1, is responsible for generating injection interrupts as well as modifying cache contents through using monitor ring. The software component, referred to as SW-MSIS, is responsible for modifying register contents, cache line flushing/invalidation, and logging. A more detailed description of the MSIS design and capabilities can be found at [6].

The key benefit of this fault injection approach is that this is the only known method of injecting faults into the caches. As we show in Table 2, the instruction and data caches together account for more than 95% of the 405's sensitive bits. Understanding cache fault behavior is crucial as the PowerPC 405's cache parity circuit as implemented within Xilinx FPGAs contains a known hardware error that prevents its use [7]. Without the parity circuit enabled, the PowerPC 405 does not automatically correct cache parity errors resulting from SEUs and other techniques must be developed.

3.1. MSIS Test Application and Evaluation

In order to evaluate the MSIS, we developed a synthetic test application modeled after a pair of space-based scientific applications. The application is composed of computational kernels from both hyperspectral and synthetic aperture radar (SAR) imaging. From hyperspectral we borrow a representative thresholding kernel, and from SAR we borrow the complex multiply and FFT kernels. Data sizes are kept small (size 128 FFTs) in order to execute entirely in block RAM, yet will still turn over cache contents. The small data sizes aid in making the results easier to analyze but is not a requirement of the MSIS. All general-purpose registers are used within the test application as are both 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.

The application repeatedly performs 1-dimensional FFTs and complex multiplication followed by thresholding in order to mimic both hyperspectral and SAR imaging. At system startup a golden output is calculated that is used to verify results during the injection campaign. A backup of the golden output is also maintained in order to ensure the accuracy of the golden output throughout the injection campaign. If at any time a data error is found (either the golden outputs or a computed result), the PowerPC logs the error to the UART, resets itself, re-computes golden outputs, and continues the injection campaign.

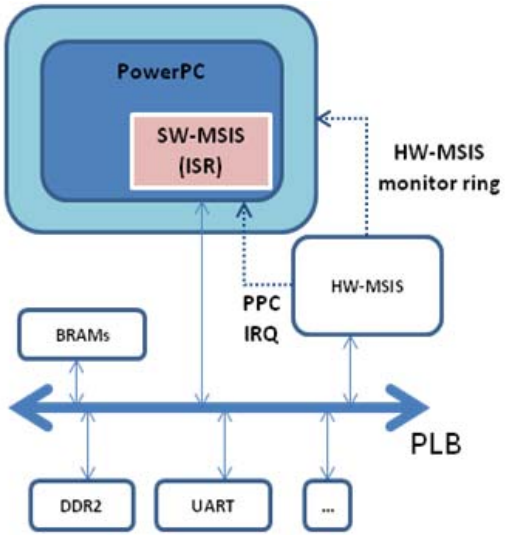

Figure 1: The MSIS architecture, showing SW-MSIS, HW-MSIS and monitor ring.

At startup, the application completes a calibration phase where an average execution time is derived. The execution time is used by the MSIS to provide an upper bound on the execution during which the MSIS may inject an error. After calibration, the application enters the injection phase, which consists of the algorithm repeatedly executing and validating within an infinite loop. The SW-MSIS interrupts the processor and injects an error into the test at a random clock cycle within the bounds of the execution time derived during the calibration stage. At the end of each trial, the PowerPC under test writes the test results to the UART, which is logged to a local file system.

3.2. Injection Results

True injection campaigns require a statistically significant number of injections, however even basic embedded programs contain 100's KBs of instructions, operating on 100's of MBs of data in just a few seconds, making the number of injections required for statistical stability extremely large. Running these experiments will take quite some time and we will release increasingly larger sets of data over the course of the project.

In this paper, we present two initial injection campaigns designed to confirm that the PowerPC 405 behaves as expected, given bit error injections and that our fault tolerance approaches hold to first order. We cannot predict the behavior of an arbitrary injection - indeed many injections will simply not propagate to the application level. Some injections, however, are quite predictable. For example, manipulating the stack pointer or program counter are extremely likely to put the processor into an undefined state (i.e. hang the processor). Other injections, such as cache injections, are far more subtle and depend on program execution – whether the

cache values were consumed or simply evicted, for example.

Our initial test campaign consists of 3,000 baseline injections without RHBSW (Table 3) and 3,000 injections with RHBSW techniques added (Table 4). In these tables fault injection data is categorized into several categories. Good Data (no action) means that the fault injected had no effect on the resulting data test vector. Good Data (rollback) means that a fault was injected which was successfully detected and by performing a rollback operation the correct result was obtained. A Data Error condition means the program completed execution, however after injection computed an incorrect result. Reset indicates that the processor was hung. In the baseline testing the processor needed to be reset completely, with RHBSW techniques, the radiation hardened controller is able to reset the PowerPC and restart it at the last known valid checkpoint.

Table 3: Summary Baseline Injection Results

Result	*Percent*
Good Data (no action)	*86%*
Good Data (Rollback)	*0%*
Data Error (no action)	*5%*
Data Error (rollback)	*0%*
Reset	*9%*

Table 4: Summary Fault Tolerant Injection Results

Result	*Percent*
Good Data (no action)	*85%*
Good Data (Rollback)	*9%*
Data Error (no action)	*4%*
Data Error (rollback)	*0%*
Reset and Rollback	*2%*

Here, we can see that roughly 85% of all faults have no effect on the program execution. This is due to not only a significant number of don't care bits being flipped, but also due to the temporal sensitivity of data in a processor in that data or instructions already executed being corrupted will have no effect on the result. With no protection, 9% of injections will produce a hard fault and need to reset the processor, while 5% of the errors will produce a soft failure resulting in data corruption. Our initial RHBSW techniques have a dramatic impact. First, there is no hard failure condition any more as the radiation hardened controller can reset and rollback what would normally be a hung processor. Additionally, the number of injections that result in this potential condition is dropped to only 2% of the time. Another strong benefit is that 4% of injections that would normally have produced a data error, are now caught and mitigated. As a result of our RHBSW techniques, only 4% of injections result in a data error. While in percentage terms this is an increase from 86% to 96% chance of producing the correct result, this represents over an order of magnitude increase in Mean Time Between Data Error (MTBDE).

While these injections do not represent a comprehensive evaluation of the PowerPC 405 and the fault tolerant techniques, for example, we cannot target the execution pipelines or any external buses, they do align well with our expectations of the processor given the injections observed and are promising for future investigation. Under the remainder of the project, we will collect and correlate data across software emulation, radiation testing, and in space test results. Collection is underway for orders of magnitude more software injection fault data. Laser injection test campaigns are scheduled. Most importantly, A-OPSS software has recently been uploaded onto a SpaceCube 1.0 processor card on board the International Space Station as part of the MISSE-7 experiment and results are currently being collected. Final fault mitigation analysis will seek to correlate across these three experiments.

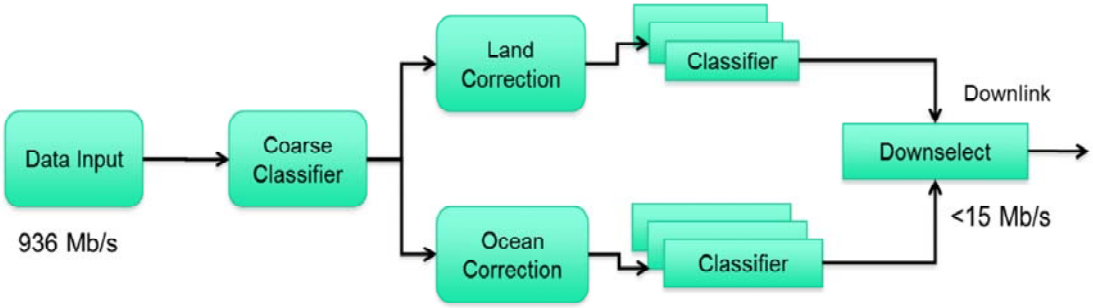

Figure 2: Hyperspectral imaging autonomous framework.

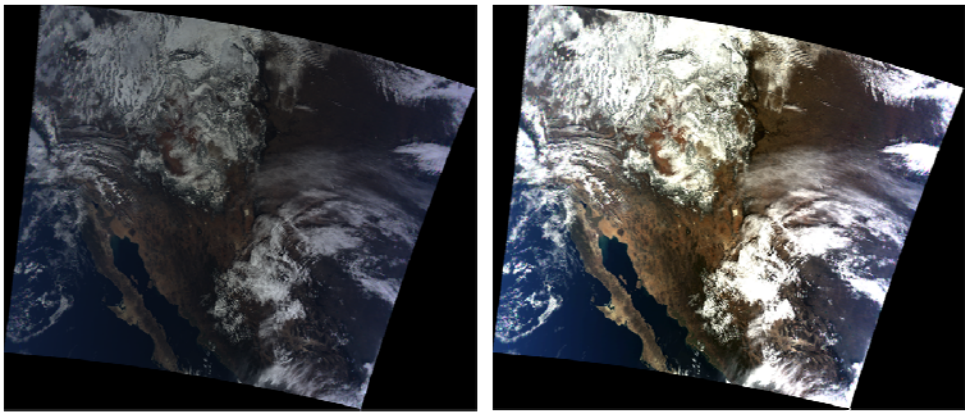

Figure 3: Hyperspectral image before and after atmospheric correction.

4. Autonomous Onboard Processing

With an increase in the amount of computational throughput made possible by commodity high performance processors and low overhead fault tolerance, new applications can be considered for on-board processing. Our final thrust is to demonstrate the performance increase yielded and applications enabled by A-OPSS RHBSW techniques. To demonstrate the kinds of capabilities the A-OPSS fault tolerance approaches yield, the team focused on applications representative of the Decadal Survey HyspIRI mission, which uses high throughput Thermal Infrared Scanner (132 Mbps) and Hyperspectral Visibe ShortWave InfraRed (804 Mbps) instruments, while having only a 15 Mbps downlink channel. This mission provides a great many use scenarios for onboard processing, from high compression algorithms, to pre-processing and selective download of high priority images, to full on-board classification.

We have created a framework for an autonomous system which integrates all classifier types and based on pre-assigned priorities, performs look ahead computations to determine if an urgent event is detected, i.e. fires, prioritizes that image for downlink, and sends only the relevant pixels of data, as opposed to the entire hyperspectral data cube. The framework can be seen in Figure 2. Here, we take in the input data and perform a quick classification to autonomously detect if this is a land or ocean image. Based on the coarse classifier the appropriate atmospheric correction algorithm is used. An example of the input and output of atmospheric correction can be seen in Figure 3. After the data is properly calibrated, several classifiers can be run in parallel, and based on the classifiers, selects only the relevant pixels of the hyperspectral data cube for downlink.

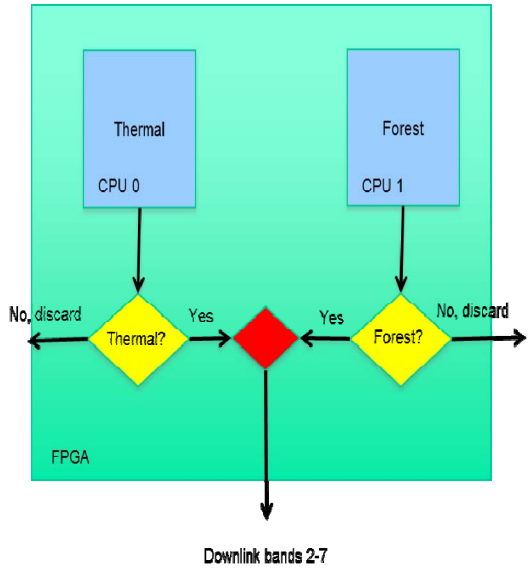

Figure 4: Basic, rapid hyperspectral mapping to SpaceCube.

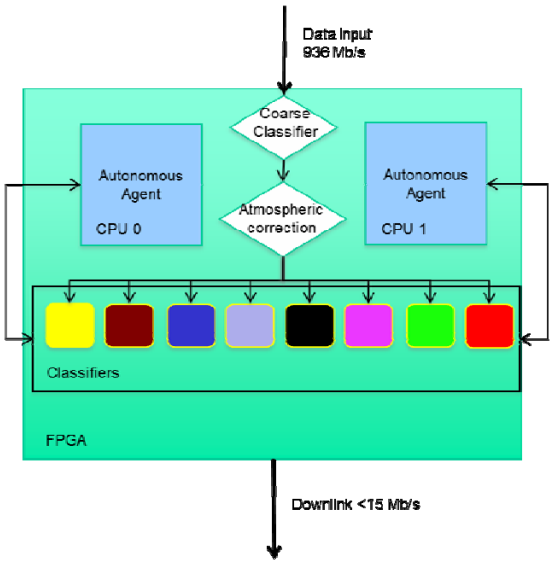

Figure 5: High performance hyperspectral mapping to SpaceCube.

To date, this application is running end to end on a generic desktop computer. It is demonstrating autonomy by being able to analyze its own sensed data and make simple determinations such as determining land vs ocean, the correct type of classifiers to utilize, and to selectively downlink image regions of interest. Portions of the system have been ported to the PowerPC within the FPGA hardware. While this is an important step, further refinement will occur over the upcoming year in moving some of the computations to the FPGA fabric where much higher degrees of parallelism can be realized, further increasing system throughput.

In Figure 4 and Figure 5 we depict our proposed initial and final hardware implementations. In our initial implementation we will leverage the PowerPCs to perform classification with up to two concurrent classifiers. This design assumes pre-corrected imagery and allows us to provide a proof of concept for our autonomous work. It does not include our fault tolerance routines.

The final implementation, shown in Figure 5, includes support for atmospheric correction parallelized within the FPGA fabric. Rather than using the PowerPCs for classification, they are repurposed for autonomous operations. The classifiers (e.g. thermal, sulfur, etc.) will also be implemented in the fabric. Not only does this design effectively leverage many aspects of the FPGA, but it allows for high performance correction and classification by leveraging the strengths of the FPGA as a stream processing architecture. Initial study of a variety of classifiers has revealed that these are extremely well suited to residing in the FPGA fabric as they typically consist of only a handful of simple algebraic operations and the hyperspectral image lends itself to highly data parallelizable solutions. Depending upon FPGA size and utilization, an efficient FPGA-based implementation of a hyperspectral classifier could hold 100's of classifiers in parallel. Our final implementation will also include our fault tolerance layer, providing software-based fault tolerance to the PowerPCs and coordinating fault tolerance with the FPGA fabric, to show full expected performance results.

5. Summary

A high performance and low overhead fault tolerance strategy targeting scientific applications on the SpaceCube 1.0 platform has been enhanced with initial results showing an order of magnitude increase in Mean Time Between Data Error and a complete elimination of processor hangs. Initial study of representative hyperspectral applications also proves promising due to high levels of data parallelism and fine grained parallelism acehivable within FPGA System on a Chip architectures enabled by A-OPSS RHBSW techniques. In the upcoming year, fault analysis data will be refined and

correlated between software fault emulation, laser testing, and space based results. The project will also deliver expected performance results on an optimized, representative hyperspectral imaging application demonstrating autonomous operations.

6. References

[1] http://www.xilinx.com/esp/aero_def/see.htm

[2] Brian Pratt, Michael Caffrey, Paul Graham, Keith Morgan, and Michael J. Wirthlin, *"Improving FPGA Design Robustness with Partial TMR"*, IEEE International Reliability Physics Symposium (IRPS) pp. 226-232, April 2006.

[3] M. French, J. Walters, M. Bucciero, "Autonomous On-board Processing for Sensor Systems: High Performance Fault Tolerance Techniques," NASA Earth Science Technology Forum, Crystal City, VA, June 2010.

[4] M. Bucciero, J. Walters, M. French, "Radiation Hardening by Software for the Embedded PowerPC," IEEE Aerospace, Big Sky Montana, February 2011.

[5] M. Bucciero, J. P. Walters, R. Moussalli, S. Gao, M. French, "The PowerPC 405 Memory Sentinel and Injection System," IEEE conference on Field Customizable Computing Machines, May, 2011.

[6] M. Bucciero, J. Walters, M. French, "The PowerPC 405 Memory Sentinel and Injection System," IEEE Symposium on Field Customizeable Computing Machines, Salt Lake City, UT, May, 2011

[7] Xilinx Virtex-4 PowerPC 405 Errata, http://www.xilinx.com/support/answers/20658.htm, 2011.