Applying Radiation Hardening by Software to Fast Lossless Compression Prediction on FPGAs

Andrew G. Schmidt, John Paul Walters, Kenneth M. Zick, and Matthew French
Information Sciences Institute, University of Southern California
{aschmidt,jwalters,kzick,mfrench}@isi.edu

Didier Keymeulen, Nazeeh Aranki, Matthew Klimesh, and Aaron Kiely
Jet Propulsion Laboratory, California Institute of Technology
{didier.keymeulen,nazeeh.aranki,matthew.klimesh,aaron.kiely}@jpl.nasa.gov

***Abstract*—As scientists endeavor to learn more about the world's ecosystems, engineers are pushed to develop more sophisticated instruments. With these advancements comes an increase in the amount of data generated. For satellite based instruments the additional data requires sufficient bandwidth be available to transmit the data. Alternatively, compression algorithms can be employed to reduce the bandwidth requirements. This work is motivated by the proposed HyspIRI mission, which includes two imaging spectrometers measuring from visible to short wave infrared (VSWIR) and thermal infrared (TIR) that saturate the projected bandwidth allocations.**

We present a novel investigation into the capability of using FPGAs integrated with embedded PowerPC processors to adequately perform the predictor function of the Fast Lossless (FL) compression algorithm for multispectral and hyperspectral imagery. Furthermore, our design includes a multi-PowerPC implementation which incorporates recently developed Radiation Hardening by Software (RHBSW) techniques to provide software-based fault tolerance to commercial FPGA devices. Our results show low performance overhead (4–8%) while achieving a speedup of 1.97× when utilizing both PowerPCs. Finally, the evaluation of the proposed system includes resource utilization, performance metrics, and an analysis of the vulnerability to Single Event Upsets (SEU) through the use of a hardware based fault injector.

TABLE OF CONTENTS

1. INTRODUCTION

As the scientific community continues to improve its understanding of the world's ecosystems, so too do the tools used to acquire the data. The growing desire to learn pushes engineers to develop better and more capable instruments, which as a result, produces more data that must also be processed. On earth these large data sets can be evaluated on large clusters and supercomputers; however, in space these vast compute resources are absent due to limitations in power and physical space. Furthermore, radiation hardened processor's performance lag are several generations behind commercial processors due to increased fabrication costs and low demand. Therefore, satellites that can produce upwards of terabits of data a day must rely on sufficient downlink bandwidth to transfer the instrument's recorded data back to earth. To aide in the growing disparity between recorded data and transmitted data, this work looks at how commodity processors can be used for data compression.

Specifically, this work evaluates utilizing previously developed Radiation Hardening by Software (RHBSW) techniques which have been proven for autonomy [1–3], for platform FPGAs with embedded PowerPC 405 processors within the its fabric, on the predictor function of the Fast Lossless (FL) compression of multispectral and hyperspectral imagery [4]. We are motivated by the need to provide high performance processing in environments requiring fault tolerance, such as the proposed NASA/JPL HyspIRI mission [5] where two imaging spectrometers would record data measuring visible to short wave infrared (VSWIR) and thermal infrared (TIR). These two instruments have the capability to produce data at ≈930 Mb/s with an available peak downlink capacity of ≈800 Mb/s. The questions remains: how to provide high performance fault tolerant processing to realize the global misssion real-time collection, compression, and download goals of the HypsIRI mission?

This work is further motivated by the SpaceCube developed and constructed by NASA's Goddard Space Flight Center. The SpaceCube is targeted towards on-board processing and in version 1.0 includes two Xilinx Virtex4 FX60 [6] commercial FPGAs. These FPGAs each include two PowerPC 405 32-bit RISC processors. The PowerPC presents an attractive computational component for space-based systems because of its computing power. Compared to radiation hardened processors, like the RAD750 (266 MIPS) [7], the PowerPC 405 can compute nearly 3.5 times (900 MIPS) the number of computations per second. However, to use the non-radiation tolerant PowerPC 405 in a space-based system, it must be capable of identifying and recovering from a Single Event Upset (SEU). Without a recovery mechanism, an SEU within the processor could send it into an unknown or unrecoverable state.

Using these PowerPC 405 processors, we set out to explore the capability to integrate and evaluate the effectiveness of our RHBSW techniques on the predictor function of the FL compression algorithm. This is in contrast to conventional approaches which would utilize the four processors in some form of double, triple, or quad modular redundancy. Instead, our approach aims to demonstrate performance gains by using the multiple processors for computation and relying on the RHBSW infrastructure for detecting and recovering from

978-1-4577-0557-1/12/$26.00 ©2012 IEEE.
[1] IEEEAC Paper #1771, Version 7, Updated 01/18/2012.
[2] This work was supported by NASA grant #NNX09AF16G.

faults. This work will not focus on the detection or correction of silent data corruptions (SDC). Instead we assume SDCs can be corrected for on earth as part of the post processing.

To evaluate the proof of concept design several experiments are performed to collect run-time performance and reliability, along with the overhead of the RHBSW techniques on program execution and resource utilization overhead. These experiments include the use of our Memory Sentinel Injection System (MSIS) [3], a hardware based fault injector, to inject faults into the PowerPC's registers, data and instruction caches along with the RHBSW fault detection and recovery mechanisms.

Overall, our results are show that such an approach is feasible with low performance overhead (4–8%). Moreover, using multiple processors for computation instead of modular redundancy greatly increases the computational throughput and yields nearly linear speedup, 1.97×. The RHBSW techniques also eliminate processor hangs and improve the computed good data by 5–10%. Finally, the use of MSIS provides us with a better understanding of how a commercial platform FPGA might behave in harsh environments, such as Low Earth Orbit, where single event upsets (SEU) are more common place.

The rest of the paper is organized as follows. In Section 2 background information and related work describing the predictor function of the Fast Lossless compression algorithm, radiation hardening by software, and fault injections is presented. Section 3 presents our fault tolerant approach to implementing and evaluating the predictor function of the Fast Lossless compression algorithm on PowerPC 405 processors on an FPGA. The experimental setup and results of which follow in Section 4. Finally, we conclude within Section 5 along with a description of our future work.

2. BACKGROUND AND RELATED WORK

This section aims to provide background and related work discussion regarding the Fast Lossless compression algorithm, as well as, current Radiation Hardening by Software and embedded processor fault injection techniques. Furthermore, previous work published by the authors in each area will also be described. This paper reports on the first system wide integration of the three individual components, which includes experiences and lessons learned. The results of the integration can be found in Section 4.

Fast Lossless Compression Algorithm

The application being evaluated in this investigation is the Fast Lossless compression algorithm which has been designed for multispectral and hyperspectral imagery data [4]. While this primary investigation is principally concerned with the predictor function of this algorithm (for reasons described in greater detail in Section 3), this subsection will provide a brief overview of the algorithm. Refer to [4] for a more thorough description of the algorithm. The compressor has been proposed for standardization by the Multispectral and Hyperspectral Data Compression (MHDC) working group of the Consultative Committee for Space Data Systems (CCSDS) and, as of May 2011, has been approved for publication by the Management Council for CCSDS [8].

The FL compression algorithm is a predictive technique using adaptive filtering to achieve low complexity and high compression, beyond current state-of-the-art implementations. The predictive technique uses the sign algorithm [9] which is a relative of the least mean square (LMS) algorithm [10, 11]. Compression operates on chunks of data that consist of three-dimensional (3-D) arrays of data samples. The sign algorithm is applied to differences between sample values and values that we call preliminary estimates or local means. The general principle used in the mean subtraction is to adjust each sample in the prediction neighborhood by the preliminary estimate in the same band as the sample but at the spatial location of the sample being predicted. Since this is done as part of a predictive compression algorithm, the difference is encoded in the compressed bitstream.

The Fast Lossless algorithm has been shown to have effective compression on the order of 4× [4]. JPLs tests with uncalibrated AVIRIS data sets demonstrate compression results of about 40% lower bit rate than state-of-the-art 2D approaches (ICER), with an approximately 4:1 compression ratio.

Radiation Hardening by Software

As opposed to conventional fault tolerant approaches, which utilize radiation hardened devices, this work focuses on techniques that can be applied to commodity processors. Towards this goal several fault tolerant techniques have been developed in software to create a more resilient system. These efforts are generally categorized as Radiation Hardening by Software (RHBSW) which use a small amount of resources to increase computational throughput instead of using a form of modular redundancy [12]. Our previous work in the area [1, 2] demonstrate that the RHBSW approach is feasible and is able to obtain an overall performance speedup by using the resources to perform distributed computation.

Our approach has been to provide low overhead fault tolerance techniques to detect and correct flow control and other errors that would otherwise crash the processor. This is accomplished by leveraging existing hardware already present within the space-based system, namely a small radiation hardened controller (such as the AeroFlex as part of the SpaceCube). In addition to the traditional tasks of configuring the FPGA and providing bitstream scrubbing capabilities, the controller would be used to monitor tasks running on the FPGAs. The controller would also be responsible for managing the pool of connected processors and distributing tasks as they become available. While this work does not focus on silent data corruptions (SDC) which may produce incorrect results, additional hardware could be added to mitigate these errors.

The techniques developed so far focus on the inclusion of a heartbeat monitor, watchdog timers, and control flow assertions in the user application. In addition a user-level checkpoint and rollback library has been developed to aide in the processor's ability to recover once a failure is detected. Our previously published research has focused on the use of these tools and techniques on a single processor [1, 2], whereas, this work is investigating how the techniques scale not only to larger sized applications, but also to multiple processors.

The use of checkpoint and rollback is a result of its wide adoption in High Performance Computing community. In many HPC applications a snapshot of the current state is taken periodically throughout application's execution. In the event of an error, the application rolls back to the most recent checkpoint rather than restarting the whole computation. Berkeley Lab's Checkpoint/Restart (BLCR) [13] is perhaps the most well known library which provides this

functionality. Accelerators also are beginning to incorporate this capability as well, CheCUDA [14] is a tool designed to checkpoint CUDA applications that use GPU as accelerators and is an add-on package of BLCR due to limitations on CUDA content existing during the BLCR checkpoint. For FPGAs several tools are being developed to provide checkpoint/rollback support at both the processor an reconfigurable fabric level [15, 16].

Hardware Based Fault Injection

Injecting faults into the embedded PowerPC 405 processors on the Virtex4 FX60 FPGA requires a different approach than conventional FPGA fault injectors [17]. Whereas most fault injectors focus on injecting into the data or configuration planes of the FPGA fabric, this work injects faults into the hard IP core of the PowerPC. To facilitate this, we have developed the Memory Sentinel Injection System (MSIS) [3]. The goal of MSIS is to provide a design team a low-cost means for rapidly injecting faults to approximate the processor's behavior in environments with cosmic radiation before actually performing beam testing. MSIS is not intended to be a replacement for conventional testing; however, a design can go through several iterations of hardware based injections to identify and correct bugs prior to running a beam test.

The MSIS tools consist of both hardware cores implemented in the FPGA fabric and software programs in the form of interrupt service routines (ISR) running on the device-under-test's (DUT) processor. To date MSIS has been targeted towards the PowerPC 405s processors within the Virtex4 FX FPGA, although ongoing work is underway to assemble a more generic MSIS infrastructure for PowerPC 440 within the Xilinx Virtex5 FX FPGA, Microblaze, and other soft core processors. The hardware cores are responsible for triggering an ISR and providing a random seed to determine which bit within the processor's context will be flipped.

Once the hardware MSIS core has triggered the MSIS software ISR the processor performs the calculated bit flip, emulating a single event upset (SEU), logs the event with a separate controller for future evaluation, and then restores the processor to its pre-interrupt state. More specifically, when the ISR is triggered, the processor stores all of the general purpose, a limited number of the special purpose, and all of the nonvolatile registers onto the vector stack before calling the MSIS software ISR. At this point the ISR has access to the software stack and can manipulate any of the registers prior to returning, in which case the processor's registers are loaded back to their general purpose, special purpose and nonvolatile register locations.

In addition to these registers, MSIS also can inject faults into the instruction and data caches of the PowerPC 405. Within the same ISR, if the fault injection is determined to be within either instruction or data cache, the MSIS hardware core performs the bit flip as the data is read from off-chip memory into the cache. To accomplish this the cacheline is invalidated within the ISR causing the cacheline to be re-read from memory. Injections can occur anywhere within the cacheline, which includes the valid, dirty, tag and data bits.

Table 1 depicts an estimation of the PowerPC 405 processor's sensitive bits which are used to characterize the sensitivity to single event upsets. Although the precise number of bits in the Xilinx implementation of the PowerPC are proprietary, these estimates are based on available documentation as well as our own experiences implementing RISC processors [18, 19]. This work does not imply that any particular bit is more sensitive to upsets than any other; however, we use this data to determine the area of the PowerPC that is vulnerable to upsets. From this data we can create a focused injection campaign that has a more uniform distribution across these sensitive bits. Section 3 will cover how MSIS is inserted into a specific design and Section 4 presents the results of using MSIS with the Fast Lossless predictor function.

3. DESIGN

The application of the RHBSW techniques on the Fast Lossless compression prediction has been heavily motivated by the NASA SpaceCube project, as well as, the proposed NASA/JPL HyspIRI mission [5]. As a result, our design focuses on key characteristics of the SpaceCube hardware, namely embedded processors running on an FPGA, and the capability to quickly and reliability process hyperspectral imagery from instruments capable of producing an estimated 47.2 Tbits of data per day.

The SpaceCube utilizes two Virtex4 FX60 FPGAs along with one radiation hardened AeroFlex anti-fuse gate array. To emulate the SpaceCube this work utilizes one Xilinx ML410 development board, which contains the same Virtex4 FX60 FPGA. This development board includes a diverse set of peripherals, such as off-chip memory (DDR/DDR2), non-volatile storage in the form of CompactFlash, and a host of external interfaces like serial ports, I2C, and Gigabit Ethernet. The FPGA also includes two PowerPC 405 processors embedded into the fabric. In this work these processors have been combined with two system-on-chips (SoC) that are common to embedded system designs. These SoCs include off-chip memory controllers for DDR2, CompactFlash controllers to read in large data sets used by the Fast Lossless application and to store the results, and UART controllers to interface with a host PC for testing and debugging. Figure 1 depicts the basic block diagram of the constructed system running on a single FPGA.

The rest of this section is divided into three subsections. First, the development and implementation of the Fast Lossless (FL) compression algorithm's predictor function on both a single and dual processor system executing on the Virtex4 FX60 FPGA is described. Next, the fault tolerance techniques developed as part of the Radiation Hardening by Software (RHBSW) running on the processors performing the FL predictor are presented. Finally, the design is augmented to incorporate the Memory Sentinel Injection System (MSIS) which is used to evaluate the fault tolerance techniques on the FL predictor function. The results of this investigation can be found in Section 4.

Fast Lossless Prediction

The Fast Lossless (FL) prediction function is one portion of the compression algorithm that has been described in Section 2. While the algorithm has been previously published [4] this is, to the author's knowledge, the first implementation of FL on an embedded processor running on a platform FPGA. Moreover, this is the first time the FL predictor has been integrated with any fault tolerance techniques, such as RHBSW.

The choice for focusing this work's implementation on the predictor function of the FL compression algorithm is largely due to the result of profiling the application on a conventional x86 processor. The profiling results indicated that ≈67.5% of the execution time was spent in the prediction function.

Table 1. PowerPC 405 Sensitive Bits

Feature	Size
Instruction Cache	16KB + 1408B tag + 64 control bits
Data Cache	16KB + 1216B tag + 64 control bits
General Purpose Register Set	32 x 32 bits
Special Purpose Register Set	32 x 32 bits
Execution Pipeline	10x32 bits
ALU/MAC	1200 bits
Timers	3x64 bits
MMU	72x68 bits
Misc	1024 bites
Total	**292,820 bits**

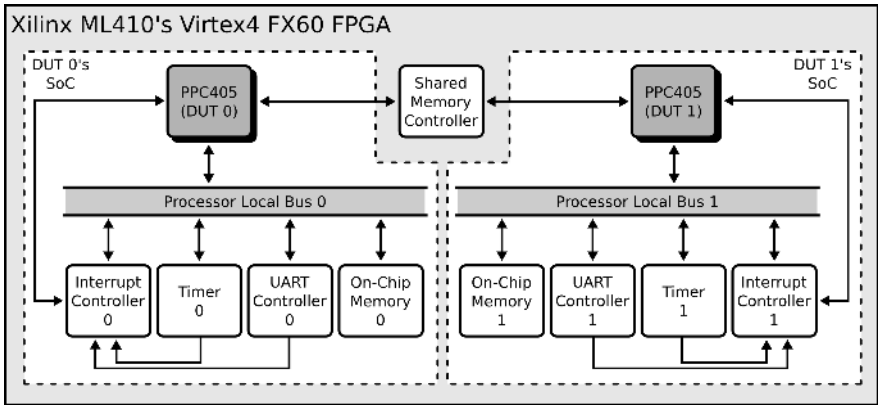

Figure 1. Block diagram of the dual PowerPC 405 systems-on-chip used in this work

Following traditional embedded development with an FPGA, the software implementation and evaluation on the PowerPC 405 is the first step towards the creation of a design running FL compression within the FPGA fabric to accelerate the application through custom hardware cores. This common approach enables rapid development of the core functionality in software first, minimizing risk. Then the software implementation's performance can be analyzed and highly parallizable functions can be implemented as hardware accelerator cores. This work will focus on the former. Our previous work [20] has focused on the latter; however, without implementing fault tolerant techniques.

For the purposes of evaluating the scalability of the FL predictor function the application was ported to both a single and dual processor implementation. The single processor implementation performs the prediction on the entire image; whereas, the dual processor system divides the image into odd and even blocks in an effort to achieve a speedup in the computation. Figure 2 illustrates the control flow graph for the application running on one or more processors in the system.

In the current implementation the image to be compressed is stored in non-volatile storage, CompactFlash, and is accessible by processor 0. The image is read into a shared memory to enable multiple processors to access the necessary image blocks for its local computation. To synchronize each processor's execution at critical sections of the application, a software barrier is used. To support multiple processors accessing image blocks of shared memory a mutex and mailbox infrastructure was added. After each processor finishes its current image block FL predictor computation a mutex must be acquired before retrieving the next block.

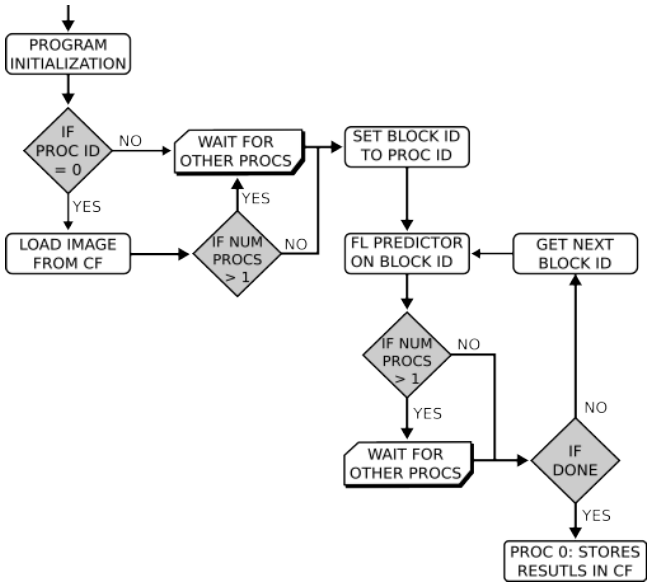

Figure 2. Control flow graph of Fast Lossless predictor running on one (or more) PowerPC 405 processors

Once the base system has been developed, the primary task for migrating the FL predictor functionality to the PowerPC 405 requires a cross-complier. Some modifications were made to support the smaller available memory resource on the embedded system and the added use of non-volatile CompactFlash to store the original image and the results of the prediction. In addition, the application was augmented to use mutexes and mailboxes to enable processors to be issued

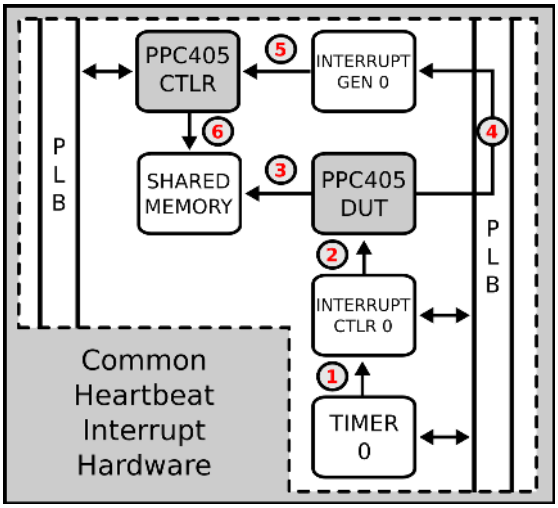

Figure 3. Section of the dual processor system-on-chip highlighting the sequence of events that occur for a heartbeat message to be sent from the DUT to the controller

the next block of the image to process. Overall, the migration to the PowerPC 405 proved to be a simple task, supporting the traditional FPGA development approach. Once the application is running on the device, the next step is in integrating the Radiation Hardening by Software techniques.

Radiation Hardening by Software

The process of applying RHBSW has previously been performed on a variety of small and medium scale applications, such as matrix-multiplication, Fast Fourier Transforms (FFT), synthetic aperture radar (SAR), hyperspectral atmospheric correction [1–3]; however, the FL predictor implementation represents the first application to run out of off-chip memory. The RHBSW techniques have been integrated in the FL predictor implementation, enhancing the reliability of the system. This includes internal and external monitoring of the *device-under-test* (DUT) through heartbeat generating, watchdog timers, and control flow assertions. In addition, checkpoint and rollback techniques taken from the High Performance Computing (HPC) community have been integrated into the system to reduce the amount of lost work when an upset occurs.

Heartbeat Generation and Monitoring—The first key component of the RHBSW infrastructure is the use of heartbeat messages, which are generated by the DUT to notify a separate controller of its current status. In our previous work, heartbeats were generated and shared through a common on-chip memory. A simple software barrier mechanism was used to pass the heartbeat status between the DUT and the controller. This mechanism requires the controller to employ polling to determine when a new heartbeat message has arrived. While easily implemented, the controller was limited in its capability to perform any other task, such as application execution management.

In this work heartbeats are now generated at a regular interval, through the use of a soft IP timer core connected to an interrupt controller through the Processor Local Bus (PLB), as can be seen in Figure 3. Once the timer expires (1) it triggers an interrupt to the DUT (2) which causes an interrupt service routine to package up any necessary status information (3) recorded since the last heartbeat and generates an interrupt (4)

```
...
#pragma BEGIN weight_calc
weight_calc(deltas,wd,height,bands,mu);
#pragma END weight_calc
...
```

Figure 4. Simple example of control flow assertion use around the weight_calc function in FL predictor program

```
long int weight_calc_loop;
...
weight_calc_loop=0,weight_calc_loop ^=1';
weight_calc(deltas,wd,height,bands,mu);
if (weight_calc_loop != 1)
  SendHeartbeat(ASSERTION_ERROR);
...
```

Figure 5. Example of control flow assertions inserted in place of `pragma BEGIN/END`

to the controller processor (5). Lastly, the controller retrieves the heartbeat packet (6) which may include information such as the DUT's current program counter value, number of interactions through a particular loop, or whether or not a checkpoint has been recorded.

For the FL implementation, the controller uses this status to determine if the DUT is busy processing the current image block, is ready to process the next block, or is in an invalid state and needs to be reset/rolled back. The adjustment of the heartbeat interval is dependent on the application and anticipated upset rate. In environments where relatively few upsets are expected, such as Low Earth Orbit, a heartbeat every tens of seconds may prove to be sufficient. Alternatively, applications which are in more hostile environments may benefit from shorter intervals. Due to the duration of execution of the FL predictor function, this work focuses on heartbeats arriving every second, although a designer can program the controller and/or the DUT to dynamically adjust the heartbeat when entering and exiting critical sections of the application.

Watchdog Timers—In addition to heartbeat generation, watchdog timers enable the DUT to self-monitor and to tell if it is still properly executing. In the event of an upset that crashes the processor, the watchdog timer will trigger the DUT to reset. However, unlike the heartbeats which are transferred from the DUT to the controller, watchdog timeouts result in a simple reset of the DUT. The watchdog is only responsible for restarting the DUT. Once the DUT restarts, it checks to see if there is a valid checkpoint to rollback to. In most cases, there will be at least one checkpoint available; however, in the event that no checkpoints remain or the reset occurred prior to the recording of a valid checkpoint the DUT restarts the application from the beginning.

Control Flow Assertions and Progress Monitoring—Control flow assertions are used to ensure proper program execution. If an upset causes the program to execute out of order, the control flow assertions enables the DUT to notify the controller that a fault has occurred. Furthermore, control flow assertions also provide one mechanism to determine if the DUT is making progress through the application. If the application ceases to make progress for some user-defined amount of time, the DUT can notify the controller to then be reset. Currently, control flow assertions are added through the use of `#pragma BEGIN` and `#pragma END` statement, which are manually added by the developer. To ease the

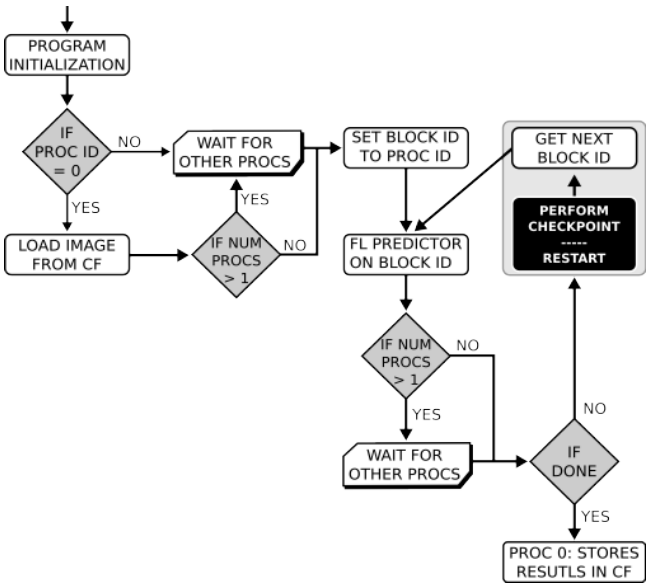

Figure 6. Control flow graph of Fast Lossless predictor running on one (or more) PowerPC 405 processors with checkpoint/rollback

integration of the actual code to perform the control flow assertions a supplemental utility was developed which replaces each `pragma` pair with the requisite assertion incriminating and checking. A drawback of this approach is that it adds execution overhead, potentially resulting in slower overall application execution. Fortunately, the developer can tune the number of assertions in the event the application experiences intolerable slowdown. An example of how control flow assertions are implemented in the FL predictor by a developer is shown in Figure 4; whereas Figure 5 shows same code after running the control flow assertion insertion utility.

Checkpoint and Rollback—The previously described mechanisms have been more geared towards fault detection, Checkpoint and Rollback is intended to aide in the recovery of the DUT. During program execution the user inserts checkpoint subroutines in order to save the state of the application's execution. Then when a failure occurs and the DUT is restarted, it can pick up where the last checkpoint left off, reducing the amount of work that is needed to be re-performed.

For the FL predictor application, checkpoints are inserted after each successive block of the image computation. If a failure occurs in the middle of one block, the program will gracefully rollback to the beginning of the block. This work improves on our previous implementations of checkpoint and rollback by enabling an application to store multiple checkpoints. Furthermore, checkpoints are now able to be stored in off-chip memory, allowing for more checkpoints to be stored and for the checkpoint sizes to be larger. The benefit to the system is that if the checkpoint data contains invalid data that will lead to a fault when restarted the system can rollback to another, earlier, checkpoint. Figure 6 illustrates the augmented control flow graph which includes the checkpoint and rollback functionality.

Experimental Setup

In order to evaluate the fault tolerance techniques integrated into the Fast Lossless predictor function, the Memory Sentinel Injection System (MSIS) is used. MSIS provides

Table 2. FL predictor runtimes with and without cache on one and two PowerPC 405 processors on Virtex4 FX60 FPGA

Number of PowerPCs	No Cache Enabled	Cache Enabled
1	883.22 s	53.53 s
2	461.08 s	27.15 s
Speedup	1.92×	1.97×

the necessary software and hardware infrastructure to inject faults into the PowerPC 405 registers, data and instruction caches. The primary role MSIS provides for this work is a reliable mechanism to evaluate how the RHBSW techniques are on the FL predictor implementation. Previous work [3] provides significant details regarding the sensitivity of specific registers, data and instruction caches. Instead, this work is focused on a medium scale fault injection campaign to profile the fault detection and recovery mechanism.

To illustrate how MSIS is integrated into the existing hardware design, Figure 7 upgrades the original design by adding two MSIS hardware cores (one for each PowerPC 405) and a third system-on-chip for system control. Since the Xilinx Virtex4 FX60 FPGA does not have a third PowerPC 405 a soft processor has been used in its place. The Microblaze processor is responsible for controlling the injection campaigns on the processors as well as receiving heartbeat monitor signals and issuing resets and rollbacks to the two DUTs.

4. RESULTS

To evaluate the Fast Lossless compression algorithm's predictor function we have developed several experiments. These experiments are intended to identify a baseline for performance and quantify the effectiveness of radiation hardening by software techniques on the predictor function. For each experiment we use one dataset from the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS), namely the 2001 uncalibrated (raw) dataset with imagery from the Island of Hawaii, Hawaii (flight f011020t01, run p03r05). We chose a dataset that fits within our embedded development board's available off-chip memory and have verified our results exactly match those calculated on a general purpose processor. The Hawaii dataset is stored in non-volatile CompactFlash memory prior to being transferred to off-chip DDR2 memory. The size of the image is ≈134 MB with 512 lines of 614 samples and an instrument width depth of 12-bits.

Fast Lossless Predictor Profiling

To begin, the predictor function's runtime performance without any Radiation Hardening by Software techniques or fault injections are shown. The test is used to demonstrate functionality as well as to provide a baseline for comparison when RHBSW techniques are applied. Table 2 lists the results when running with and without cache. The performance comparison with and without cache indicates this is a compute bound problem. The run-time performance for the single processor is 1.32 Msamples/sec (million samples per second).

While the focus of this work is not to compare against a commodity x86 processor, in order to provide a point of reference a single core 2.66 GHz processor is able to perform the same computation in ≈12.95 seconds. The PowerPC 405

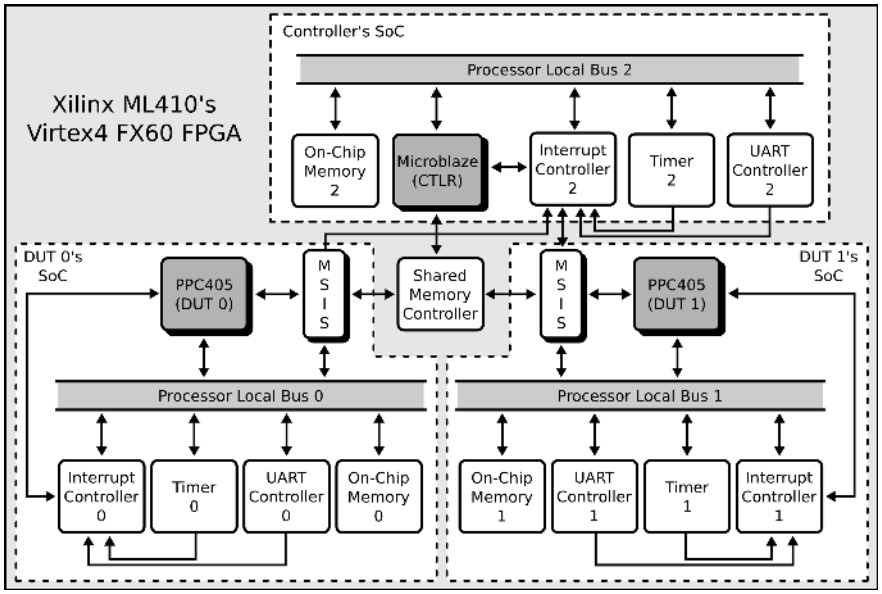

Figure 7. Block diagram of dual PowerPC 405 system-on-chips with MSIS controller infrastructure

in these experiments is clocked at 300 MHz (≈8.87× lower clock rate) yet executes only 4.13× slower than the x86 processor. The run-time performance for the x86 processor is 5.44 Msamples/sec.

The predictor function is further profiled when running on one PowerPC 405 to provide a better understanding of the execution breakdown. As seen in Figure 8, the majority of the program's execution is in performing the sign algorithm portion of the predictor function (≈78.12%). The weight calculation runs for ≈6.84 seconds, and reading the image from memory takes ≈4.87 seconds. The profiling data is useful for a variety of reasons. In a conventional embedded systems to FPGA development cycle the dominate functions would be further evaluated for acceleration in hardware. In this work we use the data to better understand where to place critical control flow assertions and checkpoints.

Table 2 also shows the performance for two PowerPCs running on a single FPGA without RHBSW techniques to show overall scalability. The ideal speedup of 2× when scaling to two processors is nearly reached, with 1.97× achieved when cache is enabled. The computed run-time performance is 2.6 Msamples/s. The speedup is primarily due to the minimal coordination needed between the two processors when computing independent blocks of the image. Furthermore, the contention for memory is largely minimized when enabling the cache since the FL predictor function is largely compute bound.

Radiation Hardening by Software Overhead Analysis

To evaluate the FL predictor function with RHBSW techniques we first implement heartbeat monitoring with a single device-under-test (DUT) at 1 second intervals. It is important to note that the heartbeats themselves do not require 1 second to execute, in fact as the results show they require significantly less time. The total execution time only increased by 0.91 seconds or ≈1.69% overhead to 54.43 seconds. When adding the Watchdog timer the time increased by an additional 0.89 seconds, ≈1.63%, to 55.32 seconds. Overall, the overhead incurred was ≈3.32%.

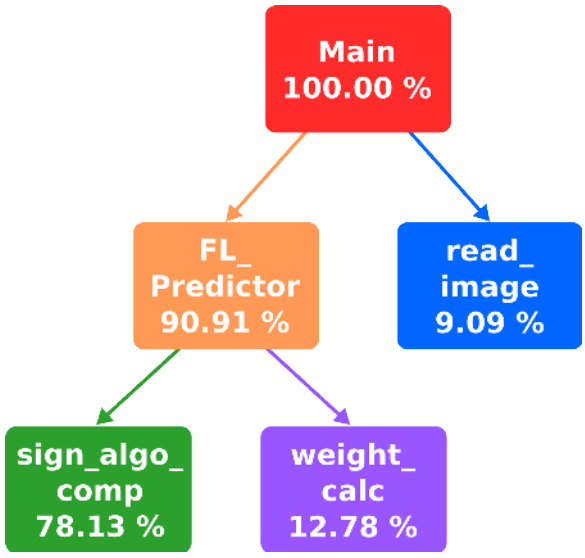

Figure 8. Runtime profiling of predictor function of Fast Lossless algorithm running on one PowerPC 405

The largest contributor to the execution overhead occurs when performing a checkpoint. For this particular image example the checkpoint size is 17.76 MB which is performed after every image block computation. The overhead incurred for a single PowerPC performing one checkpoint is 0.57 seconds, or ≈1.06%. The time for two PowerPCs to perform a checkpoint in unison is 0.59 seconds. The slight increase over one processor is due to memory contention during the write. The frequency of the checkpoints is application dependent; however, the time is commonly associated with the mean time between errors (MTBE). For an orbit comparable to the projected HypsIRI mission, we have computed some early estimates using upset characterizations for the PowerPC405 processor on the Virtex4 FX FPGA [21] of 2.56×10^{-02} upsets per device/day. These early estimates are only used to indicate the frequency of checkpointing can occur on the order of days or weeks rather than seconds or minutes, reduc-

Table 3. MSIS register injection campaign results

Result	No RHBSW	RHBSW
Good Data	79.18%	79.60%
Bad Data	15.18%	14.75%
Hang	5.64%	0.00%
Recovery via Rollback	0.00%	4.35%
Recovery via Reset	0.00%	1.30%
Total Good Data	79.18%	85.25%

ing the overhead incurred by checkpointing dramatically.

When performing a rollback the processor must retrieve the last valid checkpoint and load its contents back into the processor. This time is computed by enabling a timer prior to forcing a rollback and stopping the timer immediately after the program recommences. Furthermore, the number of rollbacks taken during a programs execution is directly related to the number of faults identified, we report only the single rollback time. For the 17.76 MB checkpoint a rollback was measured to take 0.52 seconds, or comparable to a single checkpoint time.

With all of the Radiation Hardening by Software techniques applied to the FL predictor function, a single PowerPC 405 computed the image in 55.9 seconds, or with 95.76% efficiency of the non-fault tolerant system. The real-time performance dropped only slightly to 1.26 Msamples/s. For two processors running in parallel the image was computed in 29.52 seconds, or 91.91% efficiency over the two processor non-fault tolerant system, with a real-time performance of 2.4 Msamples/s.

When comparing these RHBSW run-time performances with real-time requirements of the HyspIRI mission where data can be collected at ≈800 Mb/s (≈50 Msamples/s), we require between 20 and 40 FPGA devices (for dual and single PowerPC implementations respectively). While this number is high, the equivalent number of radiation hardened RAD750 processors [7], which are ≈3.5× slower than the PowerPC405, is ≈133 in order to fulfill the real-time performance requirement. Work is underway to leverage the FPGA's reconfigurable fabric to accelerate the compute intensive functions. Current work has demonstrated the capability in a hardware implementation [20]; however, no fault-tolerant techniques have been applied.

Hardware Based Fault Injection Reliability

The next set of experiments evaluated the systems capability to use the RHBSW techniques when faults are injected into the system. Presently, no mechanisms are used to detect when an injected fault results in a silent data corruption (SDC). For these tests two sets of a medium scale injection campaigns of 10,000 injections are performed. The first set of injections are into the general purpose and special purpose registers of a single PowerPC 405 processor. The second set of injections are into the instruction and data caches. Both sets of injections include a campaign with and without Radiation Hardening by Software techniques.

During a campaign one injection occurs per iteration of the application. First, a golden result is computed without any injections occurring. Then, after each injection, the computed results are compared with the golden results. The results are broken down into *good data*, when the computed and golden results match, *bad data*, when there is a disparity between the results, and *hang*, when the injection causes the processor to stop executing. In addition, the use of RHBSW adds two more categories of results *recovery via rollback*, when a fault is detected and the system is able to recover by rolling back to a previous checkpoint, and *recovery via reset*, when no available checkpoints exist, but the control flow assertions, watchdog timer or heartbeat monitor detected a fault.

Table 4. MSIS cache injection campaign results

Result	No RHBSW	RHBSW
Good Data	81.22%	81.21%
Bad Data	8.97%	8.98%
Hang	9.80%	0.00%
Recovery via Rollback	0.00%	5.83%
Recovery via Reset	0.00%	3.98%
Total Good Data	81.22%	91.02%

Table 5. Resource Utilization for single DUT SoC without MSIS, single DUT SoC with MSIS, dual DUT with MSIS on a Virtex4 FX60 FPGA

Design	Slices FFs	4-Input LUTs
Single DUT no MSIS	8,545 (16%)	7,530 (14%)
Single DUT MSIS	13,851 (27%)	13,460 (26%)
Dual DUT MSIS	17,176 (34%)	18,252 (36%)

The results of the injection campaign are shown in Table 3. While the difference between good and bad data is of interest, this paper is focusing on the ability to reduce the percentage of time a fault hangs the system. Most notably, the register injection campaign improves upon the 5.64% of hangs with the use of the RHBSW techniques. In Table 4 the results for the instruction and data cache injection campaign are presented. As with the register campaign the percentage of hangs is reduced to 0% with RHBSW.

Resource Utilization

Finally, we report on the resource utilization of this approach. Since this investigation spans different experiments we provide the specifics of the single PowerPC 405 SoC design, dual PowerPC 405 SoC design, and the dual PowerPC 405 SoC design plus single Microblaze design with MSIS. From Table 5, the initial design currently consumes ≈16% of the available FPGA resources on the Virtex 4 FX60. When adding MSIS, which includes a second PowerPC SoC for controlling MSIS injections, the resource utilization increases to ≈27%, adding ≈11% overhead. This is largely due to the resources needed by the second PowerPC which includes buses, memory controllers, and a second UART for debugging information. When extending the design to use both PowerPCs as devices under test the utilization increases again to ≈34%. Since MSIS is used for fault injection and evaluation, these utilization's are directly related to what a design would incur to be evaluated with MSIS. The RHBSW techniques, such at the watchdog timer and control flow assertions use the existing SoC hardware and add no additional overhead in terms of resource utilization.

5. Conclusion and Future Work

Motivated by the urgent need for fast and fault tolerant data compression of hyperspectral imagery data, this work inves-

tigates the feasibility of applying Radiation Hardening by Software techniques with the prediction function of the Fast Lossless compression algorithm. Beyond a simple proof of concept, the results show that the application can be quickly adapted to run on the multiple PowerPC 405 processors and integrated with fault tolerant techniques, such as heartbeat monitoring, watchdog timers, control flow assertions, and checkpoint/rollback with an efficiency achieved of ≈95%. Furthermore, near linear speedup is achieved when scaling the number of processors, while low overhead of ≈1.06% is incurred when performing checkpoint and rollback. In addition, we have also evaluated the RHBSW techniques through the use of our hardware based fault injection tool, MSIS, and eliminated processor hangs while improving the total good data results to above 85%.

This work has taken several significant strides towards addressing the feasibility question of such an approach. Our next step is to continue down the FPGA development cycle and migrate key functionality of the predictor function into hardware accelerator in the FPGA fabric. While some research has already investigated a hardware implementation [20], no fault tolerant techniques have been applied to the hardware approach. Even though several options exist, such as bitstream scrubbing [22], we aim to employ an alternative checkpoint/restart technique that we have been developing for the FPGA fabric [15]. This approach will allow us to directly integrate with our existing Radiation Hardening by Software approach that has been solely targeting the PowerPC 405 while providing hardware accelerated computation.

REFERENCES

[1] 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.

[2] J. Walters, R. Kost, K. Singh, J. Suh, and S. Crago, "Software-based fault tolerance for the maestro many-core processor," in *Aerospace Conference, 2011 IEEE*, march 2011, pp. 1–12.

[3] 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.

[4] M. Klimesh, "Low-complexity lossless compression of hyperspectral imagery via adaptive filtering," in *The Interplanetary Network Progress Report*. Jet Propulsion Laboratory, November 2005, pp. 1–10.

[5] NASA/JPL, "http://hyspiri.jpl.nasa.gov," last accessed January 22, 2012.

[6] Xilinx, Inc., "http://www.xilinx.com," last accessed January 22, 2012.

[7] R. Berger, D. Bayles, R. Brown, S. Doyle, A. Kazemzadeh, K. Knowles, D. Moser, J. Rodgers, B. Saari, D. Stanley, and B. Grant, "The RAD750TM-a radiation hardened PowerPC processor for high performance spaceborne applications," in *Aerospace Conference, 2001, IEEE Proceedings.*, vol. 5, 2001, pp. 2263–2272.

[8] CCSDS, "Draft recommendation for space data system standards, lossless multispectral & hyperspectral image compression," may 2011.

[9] A. Gersho, "Adaptive filtering with binary reinforcement," *Information Theory, IEEE Transactions on*, vol. 30, no. 2, pp. 191 – 199, mar 1984.

[10] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *IRE Western Electric Show and Convention Record*, August 1960, pp. 96–104.

[11] B. Widrow, J. Glover, J.R., J. McCool, J. Kaunitz, C. Williams, R. Hearn, J. Zeidler, J. Eugene Dong, and R. Goodlin, "Adaptive noise cancelling: Principles and applications," *Proceedings of the IEEE*, vol. 63, no. 12, pp. 1692 – 1716, dec. 1975.

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

[13] H. H. Paul and C. D. Jason, "Berkeley Lab Checkpoint/Restart (BLCR) for Linux Clusters," in *Proceedings of SciDAC 2006*, 2006.

[14] H. Takizawa *et al.*, "CheCUDA: A Checkpoint/Restart Tool for CUDA Applications," in *International Conference on Parallel and Distributed Computing, Applications and Technologies, 2009.*, 2009.

[15] A. G. Schmidt, B. Huang, and R. Sass, "Checkpoint/restart and beyond: Resilient high performance computing with FPGAs," in *Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines*. IEEE Computer Society, May 2011, pp. 162–169.

[16] D. Koch, C. Haubelt, and J. Teich, "Efficient hardware checkpointing: concepts, overhead analysis, and implementation," in *Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays (FPGA'07)*, 2007, pp. 188–196.

[17] M. Wirthlin, E. Johnson, N. Rollins, M. Caffrey, and P. Graham, "The reliability of fpga circuit designs in the presence of radiation induced configuration upsets," in *Field-Programmable Custom Computing Machines, 2003. FCCM 2003. 11th Annual IEEE Symposium on*, april 2003, pp. 133 – 142.

[18] J. Granacki, "MONARCH: Next generation SoC (supercomputer on a chip)," in *Workshop on High Performance Em- bedded Computing (HPEC)*, 2004.

[19] J. Draper, J. Chame, M. Hall, C. Steele, T. Barrett, J. LaCoss, J. Granacki, J. Shin, C. Chen, C. W. Kang, I. Kim, and G. Daglikoca, "The architecture of the DIVA processing-in-memory chip," in *In Proceedings of the 16th ACM International Conference on Supercomputing*, 2002, pp. 26–37.

[20] N. Aranki, D. Keymeulen, A. Bakhshi, and M. Klimesh, "Hardware implementation of lossless adaptive and scalable hyperspectral data compression for space," in *Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on*, August 2009, pp. 315 –322.

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

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

BIOGRAPHY

Andrew G. Schmidt is a computer scientist at the University of Southern California's Information Sciences Institute. He received his Ph.D. in electrical engineering from the University of North Carolina at Charlotte in 2011 investigating efficient utilization of heterogeneous resources through static analysis and runtime performance monitoring. He receive his M.S. and B.S. in computer engineering from the University of Kansas in 2007 and 2005. His research interests include reconfigurable computing, computer architecture, high performance computing, and embedded systems. Andrew is a member of IEEE and ACM.

John Paul Walters is a computer scientist at the University of Southern California's Information Sciences Institute, located in Arlington, VA. He received his Ph.D. in computer science from Wayne State University in 2007, and his BA in computer science from Albion College in 2002. His research interests include fault tolerance, high performance computing, cloud computing, parallel processing, and many-core architectures.

Kenneth M. Zick is a Computer Scientist at the University of Southern California's Information Sciences Institute with expertise in space-based computing and adaptive systems. He earned a Ph.D. in Computer Science & Engineering from the University of Michigan in 2010. Previously, he spent 12 years in advanced chip design for IBM, Cyrix Corp., and Motorola, and was a four-year NASA Graduate Student Fellow.

Matthew French is a Project Leader at the University of Southern California's Information Sciences Institute where he leads the Fine-grained Computing Group in research pertaining to FPGA fault tolerance, trust, and cognitive applications. He was Principal Investigator of the highly successful NASA funded Reconfigurable Hardware in Orbit (RHinO) project, which first looked at many of the radiation and power issues of homogeneous SRAM-based FPGAs. He has over 20 papers and 3 patents in the areas of embedded processing and signal processing. He is a Senior Member of IEEE. He has a B.S. and M.E. in Electrical Engineering from Cornell University.

Didier Keymeulen received the BSEE, MSEE and Ph.D. in Electrical Engineering and Computer Science from the Free University of Brussels, Belgium in 1994. In 1996 he joined the computer science division of the Japanese National Electrotechnical Laboratory as senior researcher. Currently he is principal member of the technical staff of JPL in the Bio-Inspired Technologies Group. At JPL, he is responsible for DoD and NASA applications on evolvable hardware for adaptive computing that leads to the development of fault-tolerant electronics and autonomous and adaptive sensor technology. He participated also as test electronics lead, to Tunable Laser Spectrum instrument on Mars Science Laboratory. He served as the chair, co-chair, and program-chair of the NASA/ESA Conference on Adaptive Hardware. Didier is a member of the IEEE.

Nazeeh Aranki received the BSEE, MSEE and Ph.D. in Electrical Engineering from Caltech and USC. Nazeeh has 28 years of experience in design and implementation of digital and FPGA based systems. Since he joined JPL in 1994, his research interests have included reconfigurable hardware, digital signal and image processing, data compression, parallel processing, evolvable hardware, and neural networks. Nazeeh developed algorithms for compression of hyperspectral data and their implementations on reconfigurable platforms. He was awarded a patent and the NASA Space Act award for his contribution in the development of an FPGA-based neuroprocessor for automotive applications in control and diagnostics. He also served as the principal investigator and task Manager on a number of NASA, DARPA and AFRL projects related to data compression and power aware computing and communications.

Matthew Klimesh received B.S.E., M.S.E., and Ph.D. degrees, all in electrical engineering from the University of Michigan in Ann Arbor, in 1989, 1990, and 1995, respectively. He spent one year as a research fellow (postdoc) at Michigan. Since 1996 he has been with the Information Processing Group at Caltech's Jet Propulsion Laboratory, working primarily on research and development of data compression algorithms for space applications. His research interests include source coding, data compression, network coding, rate-distortion theory, channel coding, probability, and discrete mathematics.

Aaron Kiely received the BS, MS, MSE, and PhD degrees from Virginia Tech, the University of Southern California, the University of Michigan, and the University of Michigan, respectively. Since 1993, he has worked in the Information Processing Group of the Communications Architectures and Research Section at JPL, where he conducts research in data compression and error-correcting codes. He wrote the emerging CCSDS 123.0 standard for Lossless Multispectral & Hyperspectral Image Compression and is the chair of the Multispectral and Hyperspectral Data Compression working group of CCSDS. Aaron led the development of the ICER image compression algorithm being used by the Mars Exploration Rovers, and he has also provided data compression consulting for several other deep space missions and instruments. Aaron has served on the faculty of Caltech, where he has periodically taught graduate-level courses on data compression and error- correcting codes.