A FAULT-TOLERANT COMPUTER ARCHITECTURE

FOR SPACE VEHICLE APPLICATIONS

by

Jennifer Susan Hane

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Electrical Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 2012

APPROVAL

of a thesis submitted by

Jennifer Susan Hane

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citation, bibliographic style, and consistency and is ready for submission to The Graduate School.

Dr. Brock J. LaMeres

Approved for the Department of Electrical and Computer Engineering

Dr. Robert C. Maher

Approved for The Graduate School

Dr. Carl A. Fox

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Jennifer Susan Hane

April 2012

ACKNOWLEDGEMENTS

I wish to express my gratitude to my advisor and the principal investigator for this project, Dr. Brock LaMeres. Also deserving of thanks are the additional members of my thesis committee, Dr. Ross Snider and Mr. Randy Larimer, as well as Dr. Todd Kaiser, who with Dr. Snider was a supervising faculty member on the project. I also want to thank Leigh Smith with NASA's Marshall Space Flight Center and Robert Ray Jr. for their supervision and advice throughout the project. Fellow students Jylissa Whisenhunt, Eric Gowens, Todd Buerkle, and Raymond Weber designed a number of system components and peripherals and assisted me with the integration process; I thank them for their valuable input. Finally, I would like to thank my parents (Michael and Patti Hane), the members and leaders of the Seven Lakes Baptist youth group, and everyone else who supported and inspired me along the way. Funding for the work presented here was provided by the Montana Space Grant Consortium and the NASA Experimental Program to Stimulate Competitive Research (EPSCOR).

TABLE OF CONTENTS

TABLE OF CONTENTS - CONTINUED

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

LIST OF FIGURES - CONTINUED

Figure Page

LIST OF FIGURES - CONTINUED

Figure Page

ABSTRACT

The discovery of new methods to protect electronics from harsh radiation environments outside earth's atmosphere is important to the future of space exploration. Reconfigurable, SRAM-based Field Programmable Gate Arrays (FPGAs) are especially promising candidates for future spacecraft computing platforms; however, their susceptibility to radiation-induced faults in their configuration memory makes their use a challenge. This thesis presents the design and testing of a redundant fault-tolerant architecture targeted at the Xilinx Virtex-6 FPGA. The architecture is based on a combination of triple modulo redundancy (TMR), numerous spare units, repair (scrubbing), and environmental awareness. By using the spares and the partial reconfiguration capabilities of the FPGA, the system can remain operational while repair of damaged modules proceeds in the background. The environmental awareness is supplied by a multi-pixel radiation sensor designed to rest above the FPGA chip, providing information about which areas of the chip have received radiation strikes. The system places these potentially damaged areas first in the queue for scrubbing. Four implementations of the architecture with different types of computing module and numbers of spares reveal its versatility and scalability. These four demonstration systems were modeled with theoretical Markov calculations, for the purpose of determining their reliability. They were also implemented on Xilinx hardware and tested by the injection of simulated faults, based on realistic orbital fault rate data from the Cosmic Ray Effects on Micro-Electronics Code (CREME96) tool. These results confirm that the systems will be highly reliable under typical earth orbit conditions. The results also demonstrate that the inclusion of numerous spares and the sensor both lead to substantial improvements in the Mean Time Before Failure, over a traditional TMR system with only three modules and scrubbing.

INTRODUCTION

The work presented here concerns the ongoing quest to bring increased computing power to the harsh environment of space. As spacecraft and planetary rovers acquire increased autonomy and more complex tasks, the demand for improved performance in their onboard computers will continue to increase. However, this performance must be coupled with reliability. Computer systems face special challenges outside earth's atmosphere, one of the most notable being higher levels of ionizing radiation. This radiation can produce various types of erroneous voltage levels, or faults, inside electronic devices, leading to incorrect outputs. Attempts to protect electronics from radiation with shields have had limited success. A shield capable of blocking all of the high-energy particles found in space would be impractically massive, and any level of shielding adds undesirable weight and volume to the system. While radiation-hardened parts are available, these are typically slower and more expensive than their standard counterparts [1]. Thus, computer systems designed for extraterrestrial use face tradeoffs between dependability, performance, and cost. This thesis presents work intended to help make these tradeoffs more favorable for space system designers by enabling the reliable use of fast, comparatively inexpensive commercial off-the-shelf (COTS) parts. Reliability is achieved through a combination of redundancy, repair, and environmental awareness.

Of late, much attention has been focused on SRAM-based Field Programmable Gate Arrays (FPGAs) as computing platforms for space vehicles. The reconfigurable nature of these devices essentially allows them to morph into different specialized

computing systems over the course of a mission, or serve as universal spares. Thus, they combine the high performance of customized hardware with the flexibility of traditional microprocessors. Since one FPGA can serve its spacecraft in multiple capacities, they have the potential to greatly reduce weight and space requirements for the mission. FPGAs also allow spacecraft designers to upload new configuration data (essentially modifying the hardware) after launch, if an error is found or the mission requirements change.

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

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

overrule it. Initially, TMR is more reliable than a simplex (single module) system; however, ultimately its reliability will fall below that of a simplex system. The probability of faults in multiple modules begins to exceed the probability of a fault in any given single module after a certain period of time, because the TMR system has more area in which to collect faults. For this reason, TMR alone is not suitable for long missions [2]. Even if TMR were able to exceed the reliability of simplex for the entire mission duration, we could ensure much better reliability by continually correcting faults as they occur. A repair technique called scrubbing fulfills this need when combined with TMR. Scrubbing is the process of continually overwriting the contents of the configuration memory with known good data (the "golden copy"), which is read from a non-volatile storage device. The golden copy should be stored in a type of memory that is highly radiation-resistant (e.g. fuse-based memory), or triplicated itself. Scrubbing prevents the accumulation of errors which would otherwise eventually doom TMR. However, since the process of reconfiguring the entire FPGA is relatively slow, TMR is still necessary to detect errors instantaneously and prevent them from propagating to the output. Scrubbing and TMR are, therefore, complementary approaches to fault tolerance.

The topic of this thesis is a radiation-tolerant system that extends the TMR-plus-scrubbing technique in two ways. First, it provides many spare modules which the TMR system may draw on as replacements if a member of the active triad suffers a fault. The presence of these spares, which are kept in good condition by the scrubber while they remain on standby, can significantly improve the reliability of the TMR system, especially if multiple faults occur in quick succession. Partial reconfiguration, which

allows the modification of a portion of the configuration memory without overwriting all of the memory, is used so that the scrubber can repair inactive parts of the device without disrupting active parts. Reconfiguration of part of an FPGA's SRAM is a comparatively slow process, and the availability of spares allows useful computations to continue while faulted parts of the FPGA are repaired in the background. Second, our system incorporates a novel radiation sensor which provides spatial awareness to the processes of scrubbing and fault recovery. This sensor is designed to be mounted above the FPGA die in such a way that high-energy particles will pass through it on their way to strike the FPGA. The sensor has 256 pixels, each of which can signal when radiation passes through it. This information can then be used to concentrate the scrubber's efforts in the areas of greatest potential damage and avoid bringing "dirty" spares online before they are scrubbed. The sensor is described in more detail in [3]. Our analysis demonstrates that the inclusion of many spares in the TMR system and the use of the radiation sensor substantially improve the reliability of the fault-tolerant systems.

The remainder of this paper is organized as follows. Chapter 2 gives additional background on some of the design methods used for this project, and summarizes previous work in this field. Chapter 3 gives a more extensive description of and justification for our approach to fault tolerance. Chapter 4 describes the basic architecture that was used for all of our fault tolerant systems, while Chapter 5 discusses each system in detail. Chapter 6 contains a theoretical analysis of the systems' reliability, based on Markov models. Chapter 7 presents the results of various experiments that were performed to measure the dependability, overhead, etc. of the systems. Finally, Chapter 8

concludes the paper with a summary of the project results and expectations for future work.

BACKGROUND AND RELATED WORK

Radiation Effects on Electronic Hardware

Ionizing radiation can affect electronics in a variety of damaging ways. Total Ionizing Dose, or TID, refers to the cumulative permanent damage done to an electronic device by ionizing radiation. It takes the form of charge carriers that are injected into the device's insulators by radiation strikes and subsequently trapped there, where they alter the electrical characteristics of the integrated circuits. TID causes a device to degrade slowly and inevitably over time; for this reason, space hardware is rated for the amount of TID it can withstand, and is simply replaced after the specified dose has been exceeded.

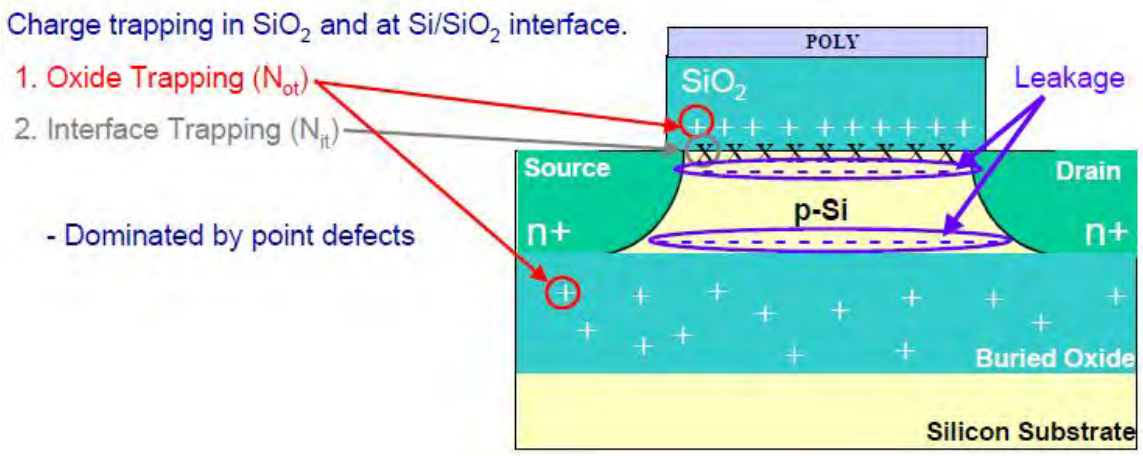

Figure 2.1 The effects of TID on the electrical characteristics of a MOS device [4].

Radiation-induced transient faults make up the other major category of negative effects. Unlike TID, these faults do not permanently damage the device, but they may cause undesirable outputs while they are active. All transient effects result from the production of extra charge carriers when a high-energy particle strikes or passes through

part of the device, ionizing some of the silicon atoms. If enough charge is concentrated in the area to reverse the state of a digital logic line in the system, the event is called a single event transient (SET). A SET is likely to result in a brief "glitch" before the excess charge dissipates, unless the new state of the line is captured and retained by a storage device (e.g. a flip-flop). A SET that is captured in this way is identified as a single event upset (SEU) and can have a more enduring effect on the output of the system. However, an SEU can generally be corrected with a quick system reset that restores all flip-flops to a known state. SEUs that cannot be dislodged by a simple reset are known as single event functional interrupts (SEFIs).

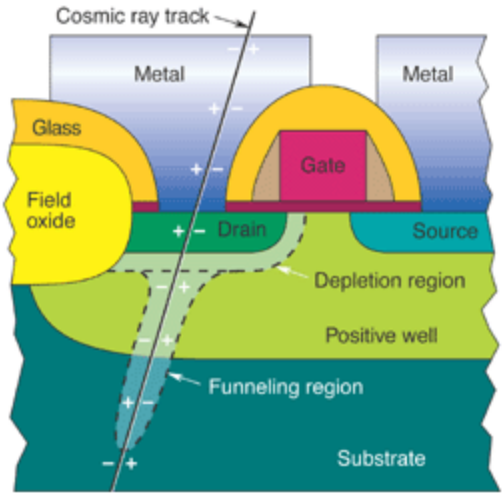

Figure 2.2: The transient charge created by a heavy ion passing through a MOS device [5].

SEUs can also lead to another type of effect called Single Event Latchup (SEL), which occurs when parasitic transistors in the silicon substrate are turned on by ionizing radiation. For these transistors, the "on" condition is stable, and they may continue conducting current until the device is reset. In some cases, latchup-induced currents can

cause destructive overheating and burn out parts of the device. It is possible to manufacture chips in such a way that they are essentially immune to latchup, at least below certain energy thresholds. Xilinx FPGAs follow the appropriate manufacturing rules to avoid latchup concerns [6], so we will not discuss this phenomenon further.

Yet another radiation effect is bulk/displacement damage, which is created when a high-energy particle knocks silicon atoms out of their places in the crystal lattice as it passes through the device. These displaced atoms result in crystal defects (vacancies and interstitials) which can alter the electrical characteristics of the device. Bulk damage is semi-permanent; the vacancies and interstitials persist in the crystal lattice, but they can migrate and will annihilate each other if they meet. Heat treatment increases the rate of movement and annihilation of defects; thus, bulk damage can be annealed away. Fortunately, this effect is fairly minor in the sort of electronic hardware we are considering, since it has a minimal effect on MOS devices [7].

FPGA-Specific Consequences of Radiation

This thesis focuses on mitigation of the radiation issues most critical for FPGAs, namely SEUs and SEFIs. In some ways, FPGAs are uniquely susceptible to radiation-induced problems. A typical FPGA stores data that represents its current configuration in banks of SRAM inside the device. Protecting this data is crucial, since it dictates which FPGA parts have connections to each other and therefore determines the nature of the circuitry. Radiation-induced faults within this SRAM can actually change the function of the device. Such faults cannot be corrected by a simple reset; rather, the damaged

configuration data must be overwritten. Hence, any fault in the configuration SRAM is a SEFI, making SEFIs far more common in FPGAs than in traditional integrated circuits. FPGAs are also vulnerable to SEUs in their logic fabric, and can be weakened over time by TID, just like other integrated circuits.

Antifuse and flash-based FPGAs do not suffer from SEFIs the way SRAM-based FPGAs do. However, the antifuse FPGAs are not reconfigurable; they can only be programmed once. They also tend to be smaller (i.e. have fewer logic elements) than SRAM-based FPGAs [1]. Flash-based FPGAs do not yet support partial reconfiguration; to make any changes in the FPGA, one must configure the entire FPGA at once. This property reduces their versatility. Due to the lack of suitable alternatives, a few radiation-tolerant SRAM-based FPGAs have been produced. Different manufacturing techniques make these chips more resistant to faults. However, the number of radiation-tolerant FPGA models available is quite limited, they are still not completely reliable, and they are more expensive than their commercial counterparts. Thus, technologies that would enable the use of SRAM-based, COTS FPGAs are still important and sought after by the aerospace community.

Previous Attempts to Improve Reliability

Triple Modulo Redundancy and Scrubbing

Numerous fault-tolerant FPGA systems featuring TMR and scrubbing have been designed. In fact, this combination of techniques is officially recommended and supported by Xilinx [8], one of the major manufacturers of FPGAs. A few examples

should be sufficient to reveal the state of the art in this field. First of all, a sample system which uses scrubbing and TMR to protect block RAMs was created by Xilinx, as part of an application note for their FPGAs [9]. TMR and scrubbing techniques were deployed on-orbit in the Cibola satellite, where their effectiveness at mitigating SEUs was tested [10]. The Space Cube design proposed in [11] votes on the outputs of four active modules (QMR, not TMR), but in its basic concept, it is essentially similar to the other designs showcased here. The four computation modules are scrubbed to prevent accumulation of errors, and the voter and system controller are housed in an external radiation-hardened part. An FPGA-based system featuring TMR with one spare was developed at the University of Tsukuba [12]. While this system did not continuously scrub itself, faulted computation modules could be repaired by manually triggered reconfiguration. Further examples of systems that use TMR and scrubbing may be found in [13, 14].

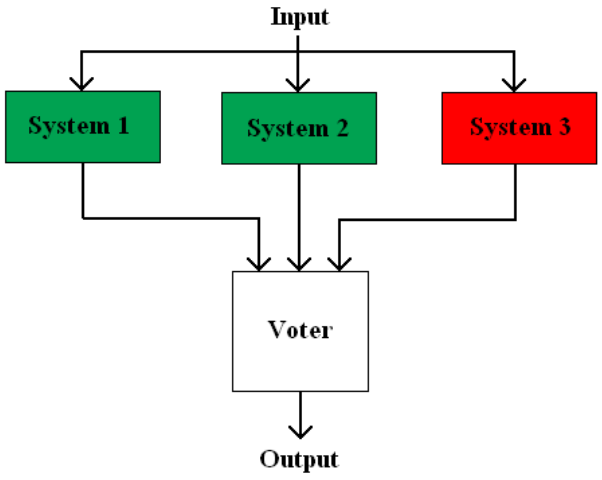

Figure 2.3: Diagram of a simple TMR setup with two functioning modules and one faulted module. Since the outputs of System 1 and System 2 agree, the majority voter will overrule System 3 and maintain a correct final output in this scenario.

Previous work at Montana State University has also included TMR systems. One of MSU's first experiments with reconfigurable computing was an FPGA system that could be optimized for one of three modes: high performance, low power, or radiation tolerance. The radiation tolerant version featured three soft processors that performed identical computations, connected to a TMR voter; the low power version had only one processor; and the parallel processing version incorporated three processors, each with its own task. Full reconfiguration of the FPGA was used to switch between modes [15]. The second set of systems built at MSU was based on TMR with spares, and could use partial reconfiguration to repair damaged modules. One of the systems in this set featured a total of sixteen PicoBlaze processor tiles (three active tiles plus thirteen spares), while the other incorporated four MicroBlaze processor tiles (three active tiles plus one spare). Both of these systems targeted the Virtex-5 FPGA from Xilinx [16].

Other Related Approaches

It may also be instructive to examine other types of fault-tolerant systems which, while they do not employ the TMR/scrubbing combination, have some relevance to the work presented here. In [17], a reconfigurable, many-tile processor system is presented. Like our systems, this concept has many SEU-susceptible tiles for performing computations, with critical control logic located in a radiation-hardened part. TMR is applied to the critical circuits, but the computational tiles can be configured for a variety of fault-tolerance levels, including triple redundancy, double redundancy, and independent parallel processing. However, this system consists of traditional processors, and lacks the customizability of a completely reconfigurable FPGA. Another example of

a fault-tolerant multi-processor architecture may be found in [18]. It is similar to the preceding one in [17], but appears to have no radiation-hardened central controller, relying instead on a "switch processor" in each tile. Thus, the routing controls themselves are protected by redundancy.

A very fine-grained approach to reconfigurable fault tolerance is presented in [19]. That paper envisions a many-tile system, in which each tile contains some spare resources, and multiple configurations (with the same function but different placement and routing) are provided for the tile. Thus, a permanently faulted tile could be reconfigured with a design that would avoid using faulty combinational logic blocks. The Triple Modular Redundancy with Standby model [20] features a similar concept. Creating multiple routing configurations for each tile is unnecessary for dealing with SEUs and SEFIs, but would improve the system's resilience against TID, and could be complementary to our approach. Advanced scrubber-like mechanisms that can detect latent TID failures are also available [21], and would, again, be complementary to our work.

Unique Contributions of Our Approach

Our new systems are distinguished from our previous work in several ways. 1) They include larger numbers of tiles (our newest PicoBlaze system has 36 tiles, and the newest MicroBlaze system, 16). 2) They incorporate fully automatic scrubbing routines to keep the spare tiles in good repair. 3) They are designed to interface with our multi-pixel radiation sensor, and use this information to improve their fault tolerance. 4) They

were designed using the newest Xilinx partial reconfiguration tool flow, and targeted at a more advanced FPGA, the Virtex-6. Points 1) and 3) also set our new systems apart from the TMR-plus-scrubbing examples listed in the previous paragraph.

We are not aware of any previous attempts to create a fault-tolerant FPGA computing system with the large number of spare modules and high level of spatial radiation awareness that ours possesses. The sensor gives the scrubber knowledge of ionizing radiation passing through the FPGA circuits, allowing it to jump to areas with potential damage and "clean" them quickly. Standard scrubbers move through an FPGA's configuration memory sequentially, correcting errors as they find them, so the amount of time a SEFI remains in the memory depends on the relative location of the scrubber when the fault occurs. The ability to locate faults as they happen removes this disadvantage. It also reduces the average time needed to bring a spare tile online after one of the active tiles experiences a fault, since the system knows the location of all "dirty" spares and does not have to test multiple tiles before finding a functional tile to activate.

OUR APPROACH: DESIGN COMPONENTS AND JUSTIFICATION

Triple Modulo Redundancy

The radiation-tolerant systems presented in this paper rely heavily on redundancy to maintain their reliability in the harsh environment of space. Specifically, they employ triple modular redundancy (TMR) with majority voting. TMR is a proven technique with a long history of use in spacecraft computers and other critical systems [22]. TMR requires more than three times as much hardware as a non-fault-tolerant simplex system, since the computing hardware must be triplicated and a voter added; such an expense is troublesome for designers of space systems, who are always concerned with minimizing the weight and power requirements of their electronics. Nonetheless, TMR proves its worth by frequently demonstrating superior reliability when compared to other forms of fault tolerance, such as self-checking pairs and error correction codes [23, 24]. However, this improved reliability only applies for relatively short missions, because TMR can endure a fault in only one of its three modules before failing. Since the larger amount of hardware used for TMR creates a greater chip area in which faults can occur, a TMR system can actually become less reliable than a simplex system if the TMR is improperly applied [25] or if the system is exposed to radiation for too long without repairs [2].

Spares can be added to a TMR system to improve its resilience against faults. Allowing the TMR system to replace a faulted triad member with a spare (assumed to be fault-free) can significantly increase the amount of time TMR can run without seeing its reliability degrade below that of a simplex system [26].

Scrubbing

Repair of the computational modules via scrubbing can assist TMR, by restoring any member of the triad that becomes damaged [2]. Like the inclusion of extra spares, scrubbing allows the TMR system to tolerate multiple single-event faults before failure. Continuous scrubbing of dormant spares also helps to ensure that the fault-free-when-activated assumption holds true for every spare. If the rate of scrubbing is rapid enough compared to the rate at which faults arrive, the system could (theoretically) remain functional for a near-indefinite amount of time, until TID-related damage begins to wear out the hardware.

It is important to note that, just as TMR without scrubbing is insufficient for long missions, scrubbing alone has its disadvantages. In a system that employs scrubbing but lacks redundancy, a single fault can arrive and have an adverse effect on the system's output before the scrubber has time to correct it [27]. Thus, while scrubbing helps to prevent faults from accumulating in a TMR system, a TMR system helps to compensate for the inherent slowness of the scrubbing process. A scrubber is useless for protecting the system from transient faults that only cause brief glitches on the output, rather than affecting the memory; the majority voting of TMR will cover those errors, however. TMR with spares is even better; should a rapid burst of faults arrive, the availability of many clean spares will help keep the system operational while the scrubber "catches up." Block RAMs and other types of user memory have ever-changing contents that cannot be compared to a golden copy, so scrubbing them is impossible unless error detection and correction codes (ECC) are employed. Putting TMR or another form of redundancy in

the system, in addition to the scrubber, helps to solve this problem. Redundancy also mitigates the effects of TID (something scrubbing alone cannot do). A device affected by high TID levels may not fail all at once, and the ability to replace a computational module with a spare in another part of the FPGA helps one avoid localized permanent errors due to TID. Therefore, TMR, scrubbing, and the inclusion of inactive spares should be seen as complementary techniques. By integrating all three of them in one system, we attempted to design a computational platform that is robust against three of the major radiation fault types that afflict FPGAs: SEUs, SEFIs, and TID.

Environmental Awareness/Radiation Sensing

The final ingredient of our fault-tolerance strategy is environmental awareness, which is provided by our custom-made radiation sensor. The speed with which the scrubber can find and remove faults is important to the overall dependability of the system; the faster a module can be repaired, the lower the probability of a second module becoming damaged before the first one is made operable again. The sensor provides the system with information about where and when radiation is striking the FPGA, allowing it to pinpoint the location of faults almost as soon as they happen, even in dormant spares. This feature reduces the latency between the occurrence of a SEFI and its removal by the scrubber.

Protection of Single Points of Failure and Other Remarks

In summary, an SRAM-based FPGA was chosen as the platform for our designs because of its inherent flexibility and attractiveness to space systems designers. Our approach incorporates TMR, scrubbing, localized radiation strike detection, and large numbers of spare modules (at least thirteen), to provide a high level of reliability. One point remains to be addressed, and that is the issue of protecting crucial control circuitry and single points of failure in the system. Crucial circuitry in our systems includes the TMR voter, the control and switching logic for swapping spares in and out of the active triad, and the circuits that implement the scrubber. Faults in this circuitry could be mitigated by 1) triplicating and voting on all of the crucial hardware as well, as described in [28], or 2) implementing all of the crucial hardware in a slower radiation-tolerant part, external to the FPGA. If the latter approach were followed, the system could still possess an advantage in cost-effectiveness and performance over a completely rad-hard / rad-tolerant system. For example, implementing the control circuitry alone in a rad-tolerant part could allow the designers to purchase a smaller rad-tolerant part than would be required to implement all of the desired computation modules, and would allow the computation modules to operate at speeds above the capabilities of the rad-tolerant part. Placing the control and voting circuitry of a redundant system in a radiation-tolerant unit is contemplated in [11]. For the present work, we have chosen to assume that the control circuitry is secure, and have only considered the impact of faults in the computational tiles. The work is still conceptual and has not yet progressed to a phase in which we

would consider purchasing an actual radiation-hardened part to house the control circuitry.

Despite its incorporation of multiple fault-tolerant techniques which compensate for one another's weaknesses, the system may still require a complete reset/reconfiguration due to more drastic types of faults. Multiple Bit Upsets (MBUs) have the potential to damage two tiles in the active triad at once, preventing the TMR voter from determining the correct output. Two SEUs or SEFIs which occur in quick succession, with a separation smaller than the time needed to replace a faulted tile with a spare, could have the same effect as an MBU. Such weaknesses are common to all TMR systems.

Thanks to the SRAM-based FPGA's ability to be dynamically partially reconfigured, the fault-tolerant techniques described above can be made highly flexible, and much potential exists for future modifications of our design. For example, an FPGA could reconfigure its voting circuits and routing on the fly as it entered areas of increased or decreased radiation in an orbit. Though the FPGA system might regard TMR as its standard configuration, it could switch to duplex or simplex operation to conserve power in safe regions, or employ an even higher level of redundancy (NMR) to guard against multiple simultaneous faults in more dangerous areas. (An example of such an adjustable system is given in [1].) It could increase or decrease the frequency of scrubbing as necessary, to balance fault tolerance against power consumption. If one FPGA were performing multiple functions (one essential, others non-essential), and the essential circuit exhausted all of its spares, other portions of the FPGA could be quickly converted

into more spares for the high-priority system. Although we have left the implementation of such possible techniques to future work, they illustrate the utility of a many-tile FPGA system for fault-tolerant computing. In particular, the ability of an FPGA to convert one specialized circuit into a fresh spare for a different circuit gives it an edge over, for instance, an array of fixed processors with reconfigurable interconnect.

PROOF-OF-CONCEPT DESIGNS: GENERAL STRUCTURE

In order to illustrate, analyze, and test our unique approach to radiation fault tolerance, we have designed and implemented four prototype computing systems. Although each design performs a different task and they feature different numbers of spare tiles, they hold their fault tolerance techniques and control architectures in common. This chapter discusses that basic system design. All four designs were targeted at the Xilinx ML605 demonstration board, which features a Virtex-6 FPGA and many convenient peripherals. The Virtex-6 was chosen as a platform for this project because, at the time the project was initiated, it was the largest commercially available FPGA that supported partial reconfiguration. The designs were created in the VHDL hardware description language, then compiled and prepared for download to the FPGA with the Xilinx ISE Design Suite of software tools. The final versions of most of the systems were developed in ISE version 13.2, but parts of one system were finished in 13.1.

Many-Core System with TMR and Partial Reconfiguration Capability

The basic fault-tolerant design produced for this project consists of a number of small hardware units, or "tiles," that perform the computations required by the system. (The required computations depend on which specific design variant is being considered; variants will be discussed in Chapter 5.) Each tile takes up a rectangular portion of the FPGA chip; the FPGA hardware within this region is configured to create the circuits required by the tile. Depending on the specific system (see Chapter 5), each tile may be a

soft processor or a more specialized circuit. At any given time, three tiles are active and connected to a majority voter, providing Triple Modulo Redundancy (TMR). The remaining tiles function as spares, and are held in reset to conserve power. If one of the three active tiles suffers a serious fault, the majority voter will mask its incorrect output, since the outputs of the two good tiles will overrule that of the bad one. The voting circuitry is also able to detect the disagreement of one tile with the other two, and declare the tile that disagrees "damaged." A simple state machine will then handle the process of deactivating the faulty tile, bringing a spare tile online, and re-initializing all three active tiles to a common state (e.g. the closest checkpoint in a processor's code) before resuming computations. The complexity of the re-initialization, and the time needed, depends on the module type. See Figures 5.2, 5.6, 5.10, and 5.14 for flow diagrams depicting the spare swap/recovery process for each of the four systems.

Each computational tile circuit includes an input that can be used to force a fault on the output (e.g. by driving the output to zero). This feature can be used to simulate transient faults that alter the output without committing any changes to the configuration memory (i.e. SEUs), for the purpose of testing the voting and recovery circuitry.

Each tile occupies a Partially Reconfigurable Region (PRR) within the FPGA. Thus, if it is necessary to overwrite the portion of configuration memory that is specific to the tile, that can be done without disturbing the operation of the other tiles or the control circuitry. Multiple circuit designs can be generated for the same PRR, as long as the interfaces to the FPGA hardware that lies outside the PRR (referred to as the static region) are consistent across designs. Hence, partial reconfiguration can be used either to

repair a PRR by overwriting its memory with clean data for the circuit that is currently present in the PRR, or to change the PRR's functionality by overwriting its memory with data for a different circuit. With that in mind, we created two different circuits that would fit inside each PRR, the first being the actual computational tile circuit (a counter, a processor, etc.). The other is a "fake" circuit which has the same input and output ports as the desired computational module, but gives a faulty, useless output. These fake tiles represent severe corruption of the original computational modules (e.g. by SEFIs). They allow us to use partial reconfiguration to perform coarse fault injection, in order to test the system's ability to respond to and repair damage to the configuration memory.

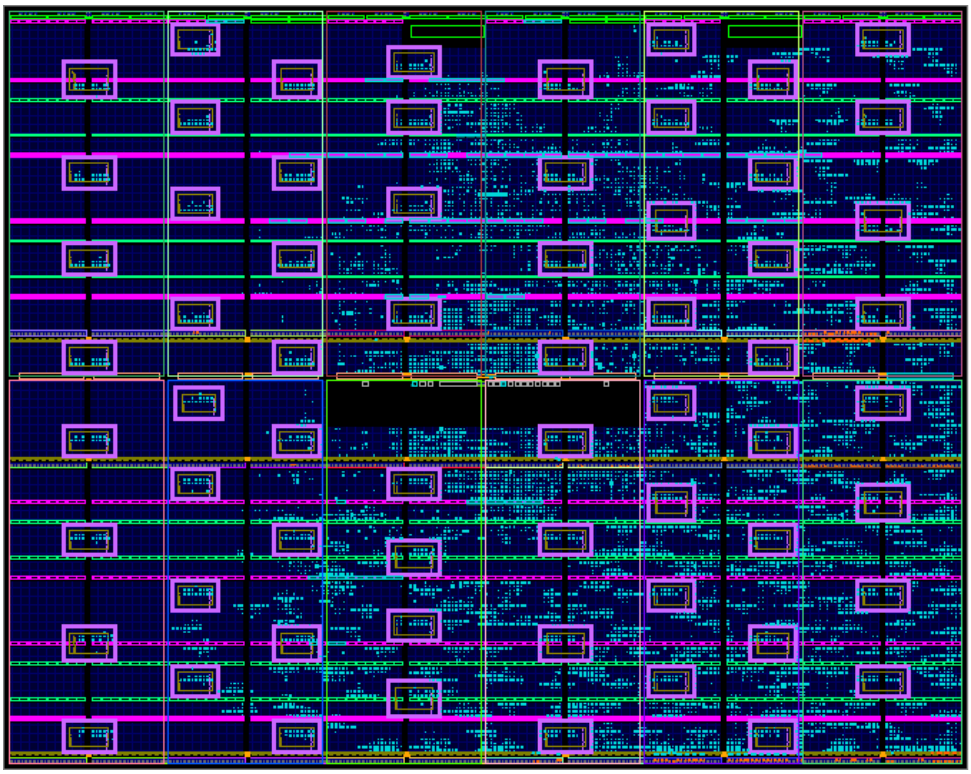

Figure 4.1: A diagram of the FPGA floorplan and implementation for one of the demonstration systems, captured from Xilinx's PlanAhead tool. Each of the sixty-four pink rectangles surrounds a PRR. The cyan dots pinpoint FPGA resources used by the design.

Once chosen, the PRRs cannot change shape or be moved to new locations within the FPGA at runtime (relocation is possible, but requires additional reverse-engineering effort; see [29]). Therefore, while the contents of the tiles in our systems may be dynamically reconfigured, the layout of the tiles themselves is fixed. Each PRR contains more combinational logic blocks than are actually necessary to create the circuits it may contain; these extra resources help the system meet timing closure by providing more possible routing paths for the Place and Route tool to use.

Scrubbers

The basic design also includes the necessary hardware and software to perform both blind and readback-compare scrubbing. The blind scrubber simply overwrites the contents of each reconfigurable tile that it scrubs, correcting errors without detecting them. The readback scrubber, however, examines each frame of the configuration memory, and only triggers a partial reconfiguration if it finds errors. It can also report the number of faults found. The scrubbers are controlled by a master MicroBlaze soft processor on the FPGA. The MicroBlaze is a standard processor core supplied to the public by Xilinx. Both scrubbers read their "golden" copy of the configuration data from the compact Flash card provided with the ML605 board, and use the Virtex-6 FPGA's Internal Configuration Access Port (ICAP) to scrub the configuration memory. The HWICAP, a standard peripheral for the MicroBlaze, is used to communicate with the ICAP port.

Notably, in systems whose tiles contain processors or other elements that make use of the FPGA's BRAMs and LUT RAMs, some frames cannot be scrubbed. Since the data located in RAM blocks changes over time, it cannot be reliably compared with the golden copy data. The Xilinx tools can create "mask files" which pinpoint the location of bits/frames that should not be scrubbed. These masks are stored on the Flash card with the partial bit files, and the readback scrubber consults them to avoid detecting false errors in bits that are off-limits to scrubbing. Since our spare processor tiles refresh their RAM after being brought into active service, and the TMR system will detect any faults in the RAM of the active tiles, scrubbing the BRAM blocks is not essential for system reliability.

Radiation Sensor Interface

The radiation sensor has sixteen conducting channels on its upper side and sixteen more on the underside, running in perpendicular directions. Simultaneous current pulses on a top channel and a bottom channel indicate a radiation strike at the pixel which corresponds to the intersection of those two channels. The sensor is connected to a custom circuit board which conditions and amplifies its signals to match the FPGA's logical voltage levels. After amplification, they are sent to the FPGA through thirty-two parallel general-purpose IO channels. Because the pulses are short when compared to the period of the system clock, it was necessary to design a high-speed event detector to capture them. The detector passes each channel's signal through a chain of flip-flops. A pulse on the channel will appear as a binary '1' traveling through the chain. The contents

of the flip-flop chain are periodically assigned to a register. If a '1' is present anywhere in a channel's register, that channel is considered active; thus, the register makes the short pulses persist in the system for a much longer time than their original width would allow. Also present on the FPGA is a binary counter for each sensor pixel. If the top and bottom sensor channels corresponding to a pixel are active at the same time, the enable line for that pixel's counter is triggered. Essentially, the system keeps count of how many high-energy particles have struck each sensor pixel. The sensor itself is designed to be mounted above the FPGA, so that any radiation which strikes the FPGA must pass through it (except for the small percentage that enters through one of the sides of the FPGA package). Since the tile PRRs reside at fixed locations within the FPGA, a correspondence between each tile and the sensor pixels above it can be established. Refer to [3] and [30] for more information about the radiation sensor, the amplifier board, and the hardware interface within the FPGA (including the event detector and counters).

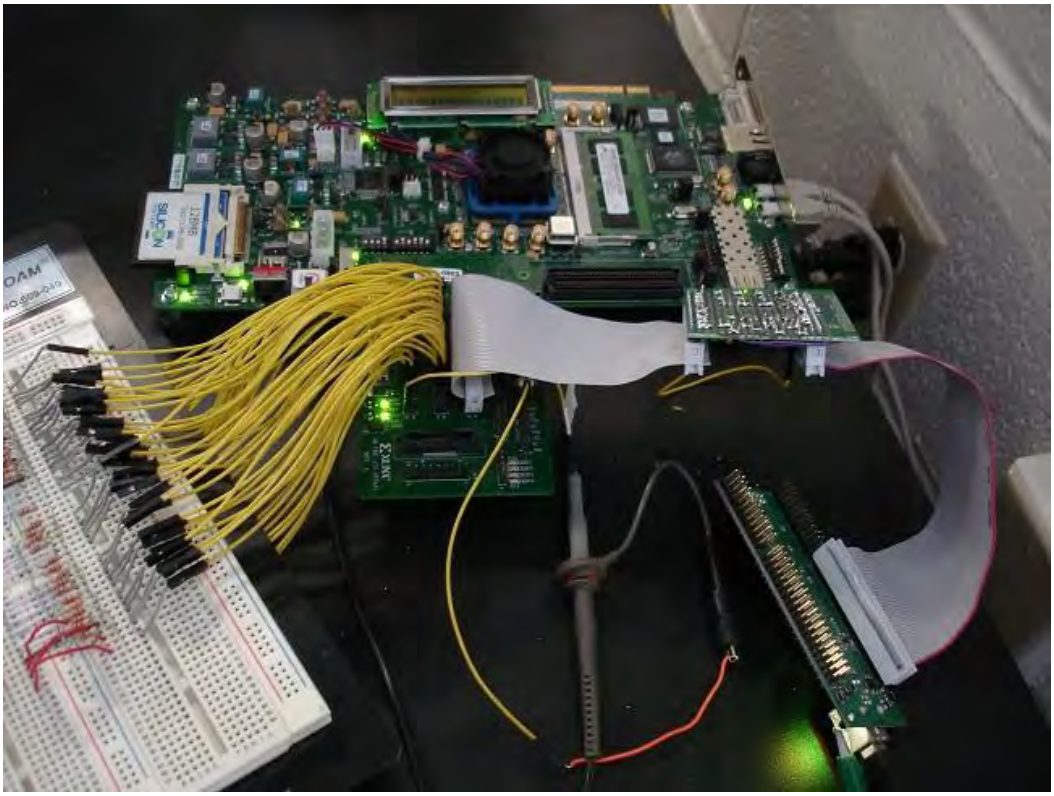

Figure 4.2: The ML605 board on which all of the demonstration designs were implemented, along with its peripherals. The Morph-IC II appears at the lower right; a temporary voltage divider which connects the FPGA to the sensor's amplifier board appears at left. An oscilloscope probe lies in the lower center.

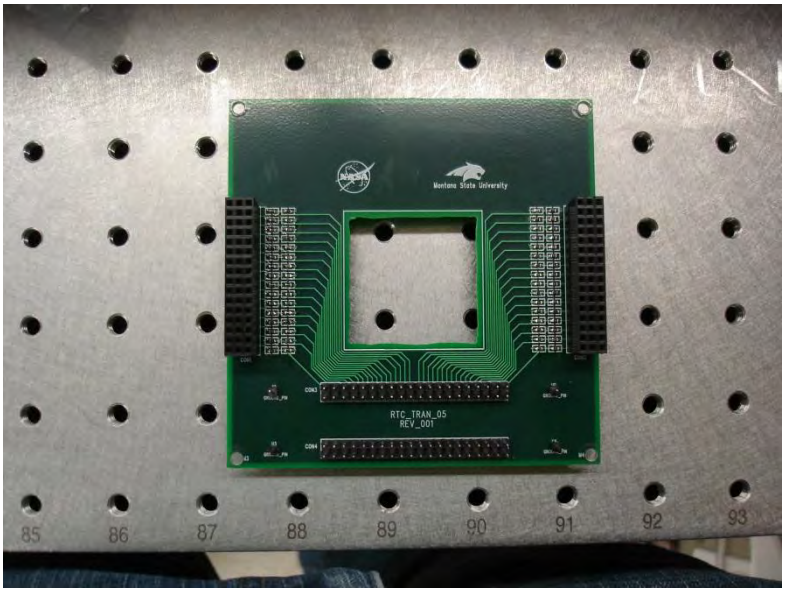

Figure 4.3: The final version of the voltage divider.

It was necessary to integrate the internal hardware interface with the radiation-tolerant many-core system, so that the sensor information could be put to use. To that end, the same state machine which takes faulty tiles offline and replaces them is also responsible for monitoring the sensor counters while in its "idle" state. If any counter corresponding to one of the pixels registers one or more counts, the tile that the pixel covers is declared damaged. If it is an active tile, it is proactively taken offline and a spare is brought online to replace it, even if its output has not yet been found faulty. Every 250 ms, the value of each pixel counter is sent across the USB interface (see User Interfaces, below), and all counters are reset.

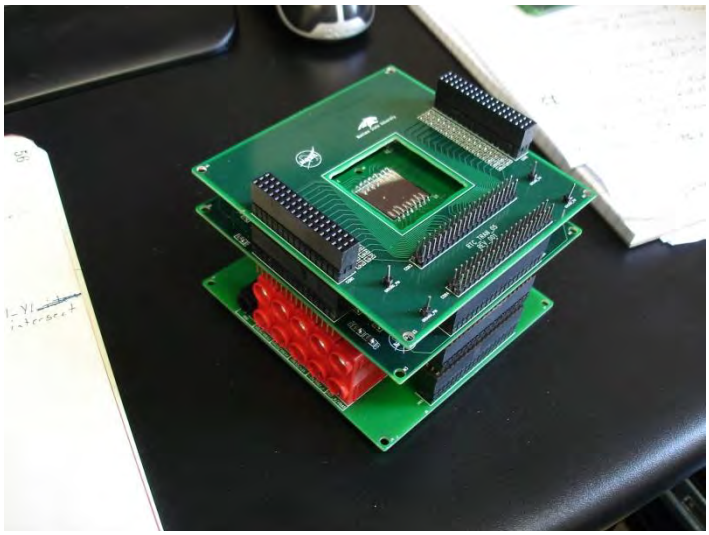

Figure 4.4: The radiation sensor and its stack of interface and power boards.

User Interfaces

Two communication interfaces, based on serial UART and USB technology, respectively, were included in each system, so that users could send commands to the system and display its internal state. The USB interface was included for its high

maximum data rate, the UART for its ease of setup and use. Two Graphical User Interfaces (GUIs) were also designed to present information from the system in a convenient fashion.

UART and USB Communication Interfaces

The master MicroBlaze which governs the scrubbing process was also provided with a UART peripheral. This UART could be connected to a virtual COM port on any computer via the USB-UART bridge provided on the ML605 demonstration board. Since the UART was simple and easy to set up, it was the first communication interface completed. It was initially used for simple text-based debugging of various system features, including the USB interface. Later, it was employed to communicate with our MATLAB-based GUI, because this application did not require a high data rate, and it was more straightforward to design the MATLAB program to use a virtual COM port rather than a USB port. (See "Orbital Environment GUI," below.)

A true USB interface was also included; although more complicated to interface with computer programs than the UART is, it offers a greater maximum data rate. Therefore, it was used to connect the FPGA board to the System Status GUI (which needs to receive a large volume of radiation strike data from the sensor). Unfortunately, the USB port on the ML605 board is difficult to use; we were unable to find any drivers that a computer on the other end of the USB cable could use to communicate with it. Instead, we chose to purchase a small USB interface board that came with drivers. Our choice was the Morph-IC-II, which includes an Altera Cyclone II FPGA and many general-purpose IO pins. Some of these pins were connected to a corresponding set of

GPIO pins on the ML605 board, and the small FPGA on the Morph-IC-II board was programmed to perform the needed conversions between parallel data and serial USB packets.

System Status and Control GUI

The purpose of this GUI is to provide an abstract visual representation of the interior of the FPGA, and allow the user to send commands to the fault-tolerant system. It was created in Visual Basic. Each computational tile is represented as a group of buttons in the GUI. Three of the buttons are color-coded to represent the status of the tile; the color green indicates a member of the active triad, red indicates a tile that has been declared damaged, and yellow indicates that the tile has been reconfigured with the "fake" version, i.e. its configuration memory has been artificially corrupted. Tiles with three beige/gray buttons are dormant, undamaged spares. The remaining button, located at the lower right of the group, displays the number of radiation strikes detected by the sensor for that tile. The radiation strike buttons begin with a white background, and gradually fade from white to red to black as the strike counts mount up. A tile whose button group is surrounded by a bright blue border is currently being scrubbed. If the readback scrubber is active and finds errors in the configuration memory of a tile, the number of corrupted memory frames will be reported in the text box at the upper right of the GUI. Some versions of the GUI have a performance graph at the lower right. This graph displays the number of processor MIPS being used for computations and the number being used for the fault recovery process. A higher fault rate forces the TMR

system to swap in spares for faulted active tiles more often, reducing the amount of time the system can spend on useful work.

A user can click the appropriate buttons in a tile's button group to force a simulated SEU fault on the tile, repair the tile through partial reconfiguration, or replace the tile with a fake through partial reconfiguration. The user can turn the blind and readback scrubbers on and off with the checkboxes at the middle right (only one scrubber can be active at a time). The GUI also includes controls that allow the user to log and/or clear the radiation sensor count information.

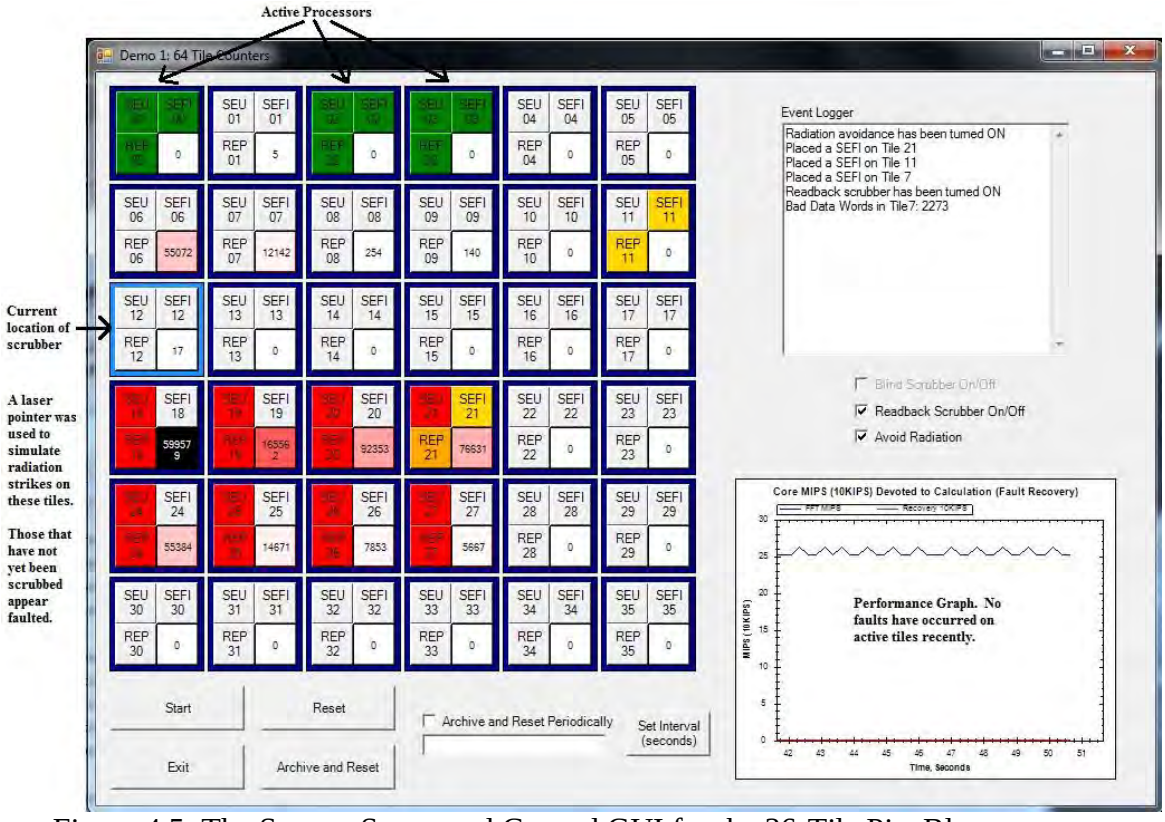

Figure 4.5: The System Status and Control GUI for the 36-Tile PicoBlaze system.

Orbital Environment Display

The Orbital Environment Display was born out of a desire to demonstrate the behavior of the systems in realistic orbital environments. It is based on a MATLAB program which runs a simulation of a spacecraft traveling on an orbit around the earth. The spacecraft is presumed to be carrying a Virtex-6 FPGA running one of our systems, and the simulation calculates and plots the fault rate of the device at each point in the orbit. The Display also includes a projection of the earth, on which the orbit and the current location of the craft are plotted. Viewers can watch the fault rate change in real time as the craft moves along the orbit. Four preset orbits are available: the International Space Station (or Zarya) orbit, a Low Earth Orbit; Molniya 1-80, a Highly Elliptical Orbit; Satcom 5, a geosynchronous orbit; and EXP-1 Prime, the orbit of Montana State University's student-built cubesat (later renamed the Hiscock Radiation Belt Explorer, or HRBE).

The Orbital Environment Display can communicate with the FPGA's UART interface through a virtual COM port on the computer. If the FPGA is connected, the Display will send it current simulation information about the fault rate the device would be experiencing on-orbit. The master MicroBlaze can accept this information and force faults on the computational tiles at the specified rate, making the FPGA and its fault-tolerant system participants in the simulation.

The Display relies on data from Vanderbilt University's CREME96 tool to create a realistic simulation. CREME96 is designed to compute the expected fault rate due to ionizing radiation for an electronic device in a given orbit. The orbit may be divided into

segments, and an average fault rate computed for each segment. Each segment is specified by a range of McIlwain-L parameters. The Display program includes a routine which uses a model of the magnetic field surrounding the earth to calculate the L-value for each point on the orbit. The fault rate may then be obtained as a function of the L-value, using an interpolated version of the imported CREME96 data. Since we could find no appropriate radiation test data for the Virtex-6 FPGA, we used data from the Virtex-5 [31, 32] as an input to CREME instead. This yields a conservative estimate, as the Virtex-5 is actually more vulnerable to radiation than the Virtex-6 [6]. Certain types of faults were trimmed from the data set in [31] and possibly in [32] as well; however, since the number of faults trimmed is several orders of magnitude lower than the number of faults remaining in the data set [31], it is not likely that they had a substantial impact on the calculations.

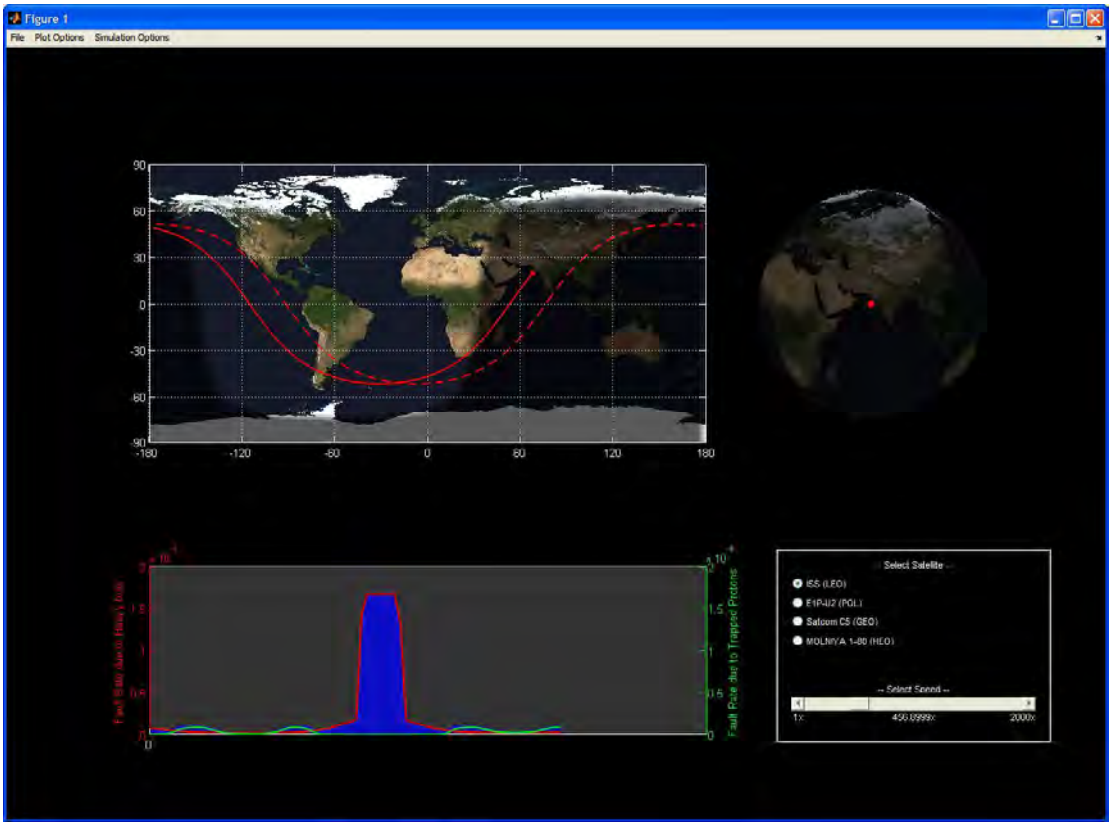

Figure 4.6: A screenshot of the Orbital Environment GUI, currently plotting an ISS orbit.

PROOF-OF-CONCEPT DESIGN: VARIATIONS

64-Tile Binary Counter System

The first reconfigurable fault-tolerant design that was prepared was the 64-tile binary counter system. Its main purpose was to test the scalability of the design, in terms of the number of tiles. Each tile in this system contained a 32-bit binary counter; we purposely chose a relatively small, simple circuit, in order to fit as many of them inside the FPGA as possible. The initial target for the size of the system was 256 counter tiles; however, this criterion proved too difficult to meet. Although the FPGA has sufficient logic resources for 256 binary counters, the complexity of the multiplexers needed to connect that many tiles to the control circuitry made the design nearly impossible to route. Since three tiles are active at any given time, the system has a total of sixty-one spares. The eight most significant bits of the final voter output are displayed on the ML605 board's user LED lights. See Figure 5.1 for a block diagram of the system, Figure 5.2 for a representation of the recovery process as a flow diagram, Figure 5.3 for the FPGA floorplan (showing the PRRs in which the counter tiles reside), and Figure 5.4 for the System Status and Control GUI.

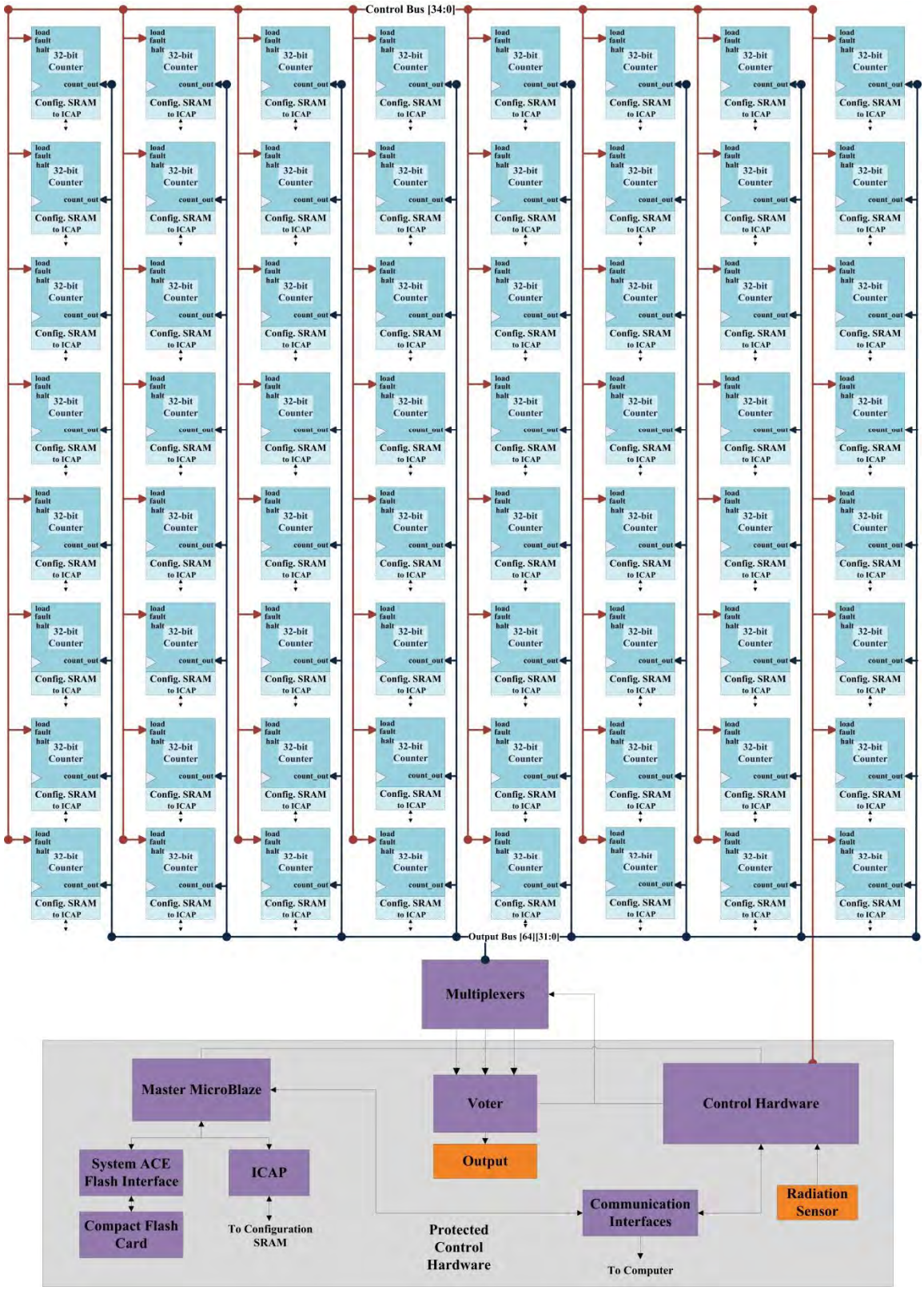

Figure 5.1: Block diagram of the 64-tile Binary Counter System.

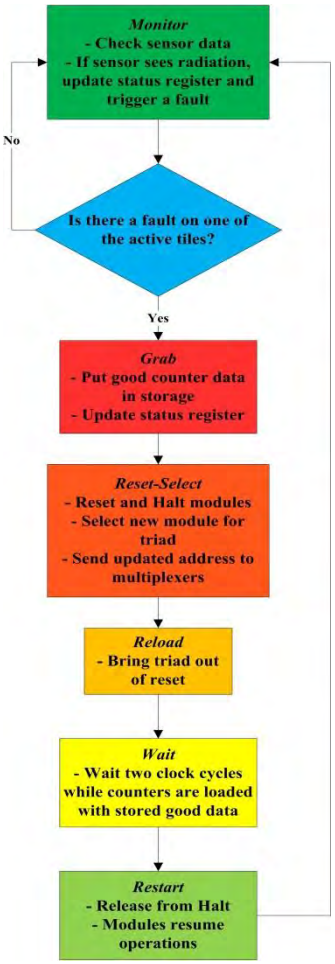

Figure 5.2: Flow chart describing the recovery process for the 64-Tile Binary Counter System (each block corresponds roughly to one state in the recovery state machine).

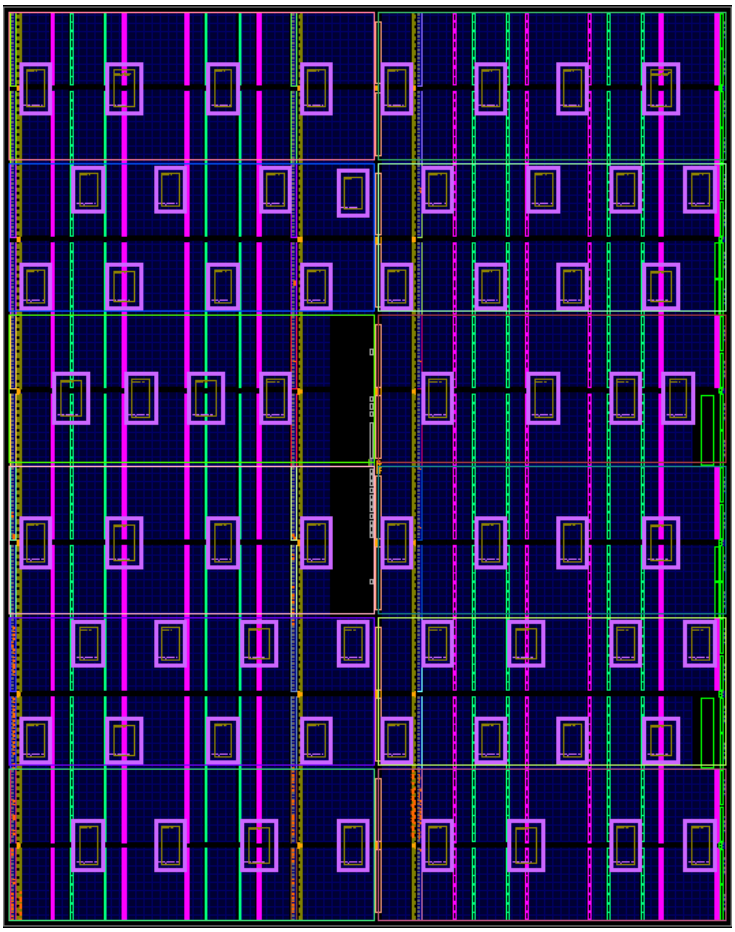

Figure 5.3: The floor plan of the 64-Tile Binary Counter System, obtained from PlanAhead. Each pink rectangle is a partially reconfigurable region, able to hold one counter module.

Figure 5.4: The 64-Tile Binary Counter System's graphical user interface.

36-Tile PicoBlaze System

The PicoBlaze is a simple soft processor which Xilinx provides to users of its FPGA products. Versions optimized for use on the Virtex-6 are available. The second proof-of-concept system uses these small processors as the basis of its computational tiles. Each PicoBlaze runs a test program which computes an eight-point FFT on hard-coded data. After passing through the voter, the outputs of the FFT computation are stored in memory and cycled across the LEDs on the ML605 board.

When an active PicoBlaze experiences a fault, the two good PicoBlazes remaining in the active triad will advance to the next checkpoint in their code, then copy the

contents of their RAM and registers to storage in the control circuitry. Once the faulted processor has been replaced with a spare and all three active processors have been reset, they will reload their RAM and registers with the values stored before reset, jump to the appropriate code checkpoint, and resume computations where they left off. Another portion of the control circuitry monitors the progress of the PicoBlazes, and reports the number of successful FFT computations they complete to the System Status and Control GUI every 250 ms. Numbers of spare swaps in the past 250 ms are also sent to the GUI. This information is used to draw the performance plot. See Figure 5.5 for a block diagram of the system, Figure 5.6 for a representation of the recovery process as a flow diagram, Figure 5.7 for the FPGA floorplan, and Figure 5.8 for the System Status and Control GUI. One peculiarity of both this system and the 16-Tile PicoBlaze + FFT Core system is the need to go through the memory offload/reload process twice. If this is not done, the new PicoBlaze module will not be in sync with the other two when the system resumes normal operation. I was never able to determine why this is the case.

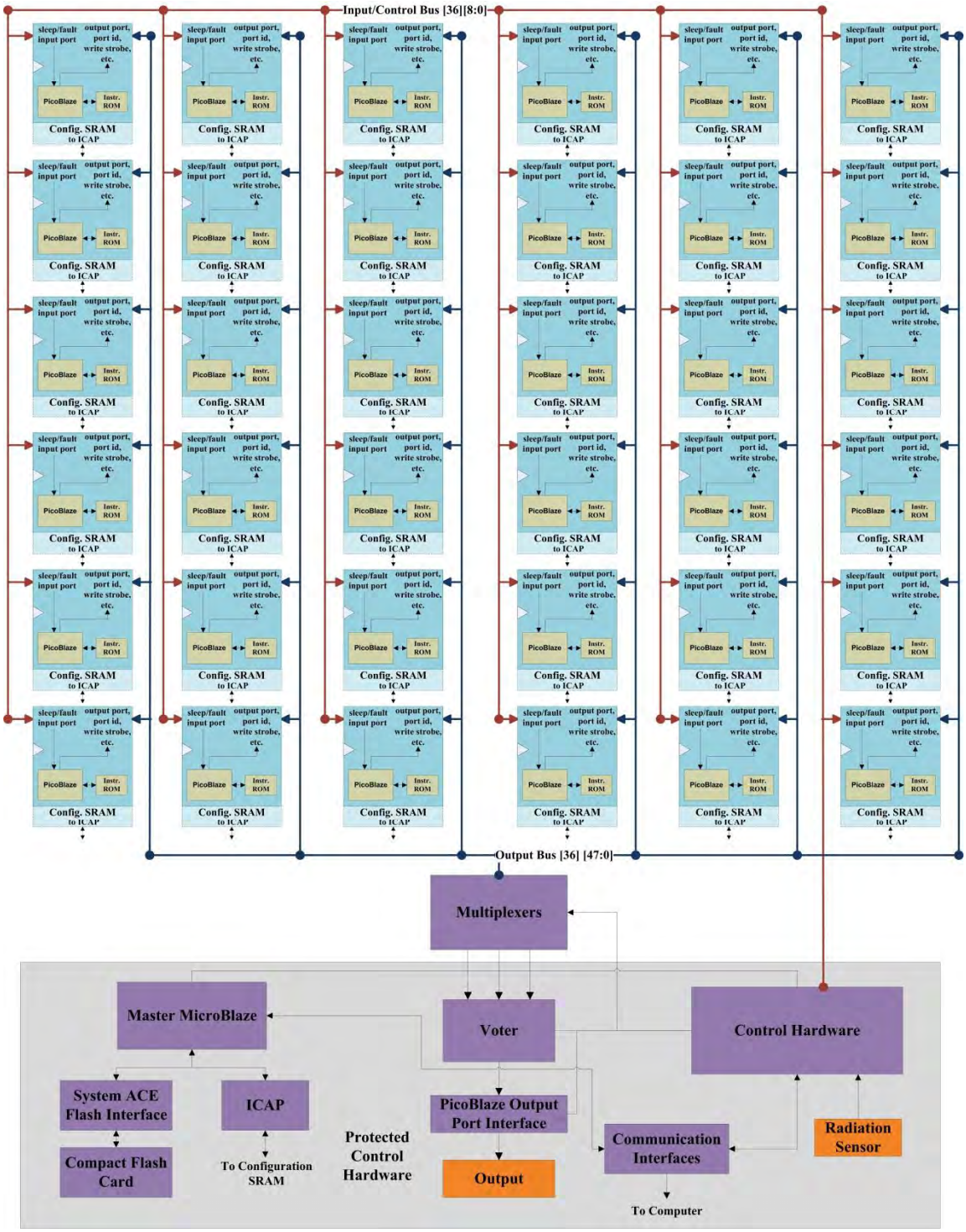

Figure 5.5: Block Diagram of the 36-Tile PicoBlaze System.

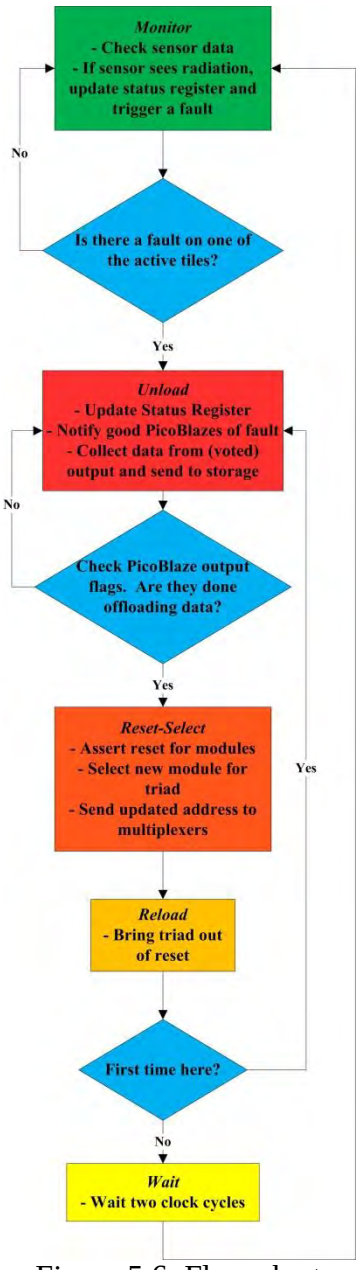

Figure 5.6: Flow chart describing the recovery process for the 36-Tile PicoBlaze System (each block corresponds roughly to one state in the recovery state machine).

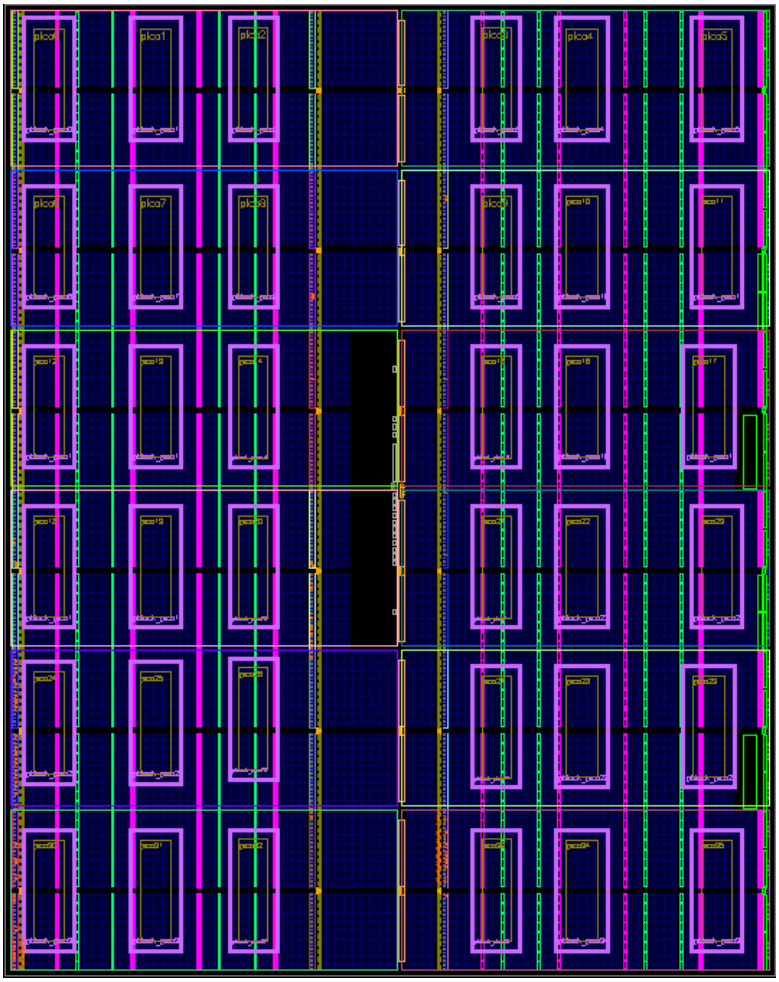

Figure 5.7: The floor plan of the 36-Tile PicoBlaze System, obtained from PlanAhead. Each pink rectangle is a partially reconfigurable region, able to hold one PicoBlaze module.

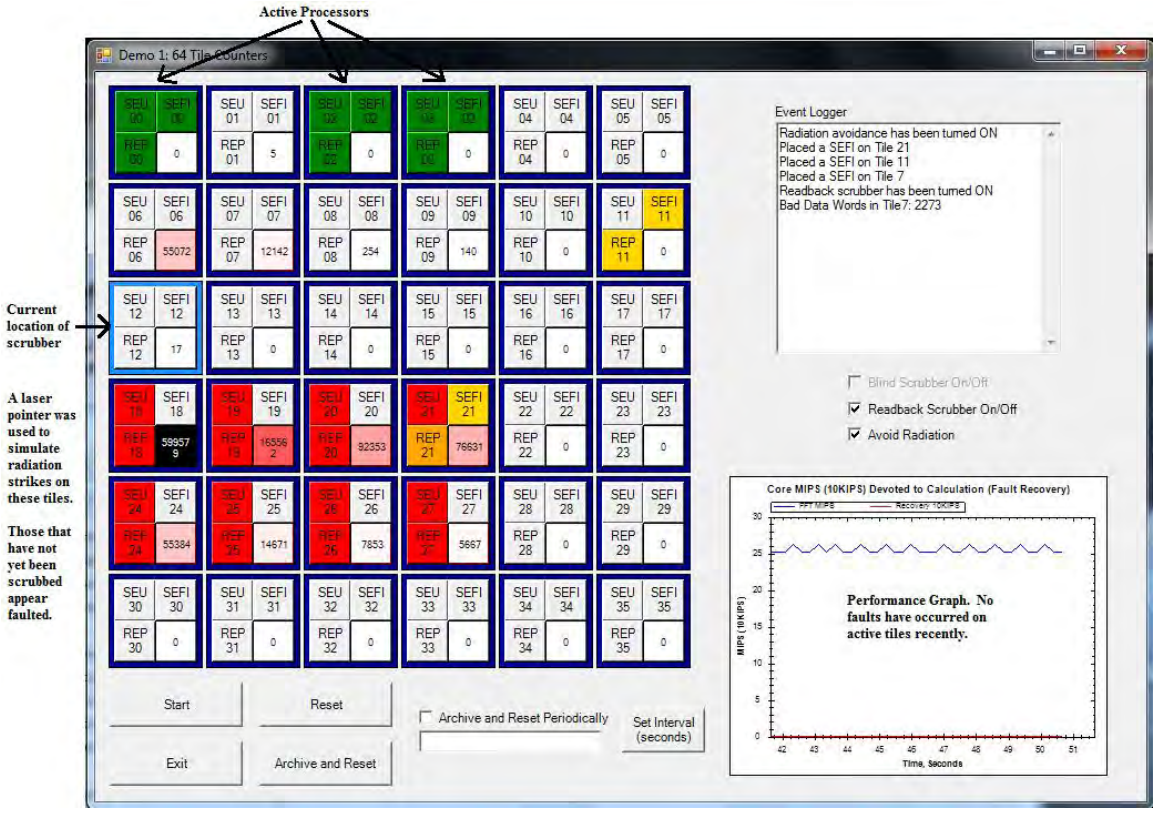

Figure 5.8: The 36-Tile PicoBlaze System's graphical user interface.

16-Tile PicoBlaze + FFT Core System

The 16-tile PicoBlaze system is much like the previous 36-tile version in some respects. Each tile contains a PicoBlaze soft processor, whose interfaces with the voter and other control circuitry are the same as in the previous case. When a tile swap is necessary, the PicoBlazes in the 16-tile system follow the same procedure of offloading and reloading their registers and RAM as in the 36-tile system. However, in the 16-tile version, each tile also includes an FFT core defined in hardware, which functions as the PicoBlaze's co-processor. The FFT core is another pre-designed Xilinx module, which was customized for this application using the Core Generator program. The superior performance of the customized hardware allows this system to perform a more practical

256 point FFT. The input data for the computation is stored on the ML605 board's platform Flash chip. Outputs are placed in memory after passing through the voter, and can be fetched and displayed in the terminal by sending a command to the master MicroBlaze via the UART interface. Much like the 36-Tile PicoBlaze System, this system also reports numbers of computations and spare swaps to the GUI so that the performance graph can be updated. See Figure 5.9 for a block diagram of the system, Figure 5.10 for a representation of the recovery process as a flow diagram, Figure 5.11 for the FPGA floorplan (showing the PRRs in which the counter tiles reside), and Figure 5.12 for the System Status and Control GUI. In Figure 5.10, you may observe the practice of offloading and reloading the memory twice, which is necessary for unknown reasons (see the 36-Tile PicoBlaze System section, above).

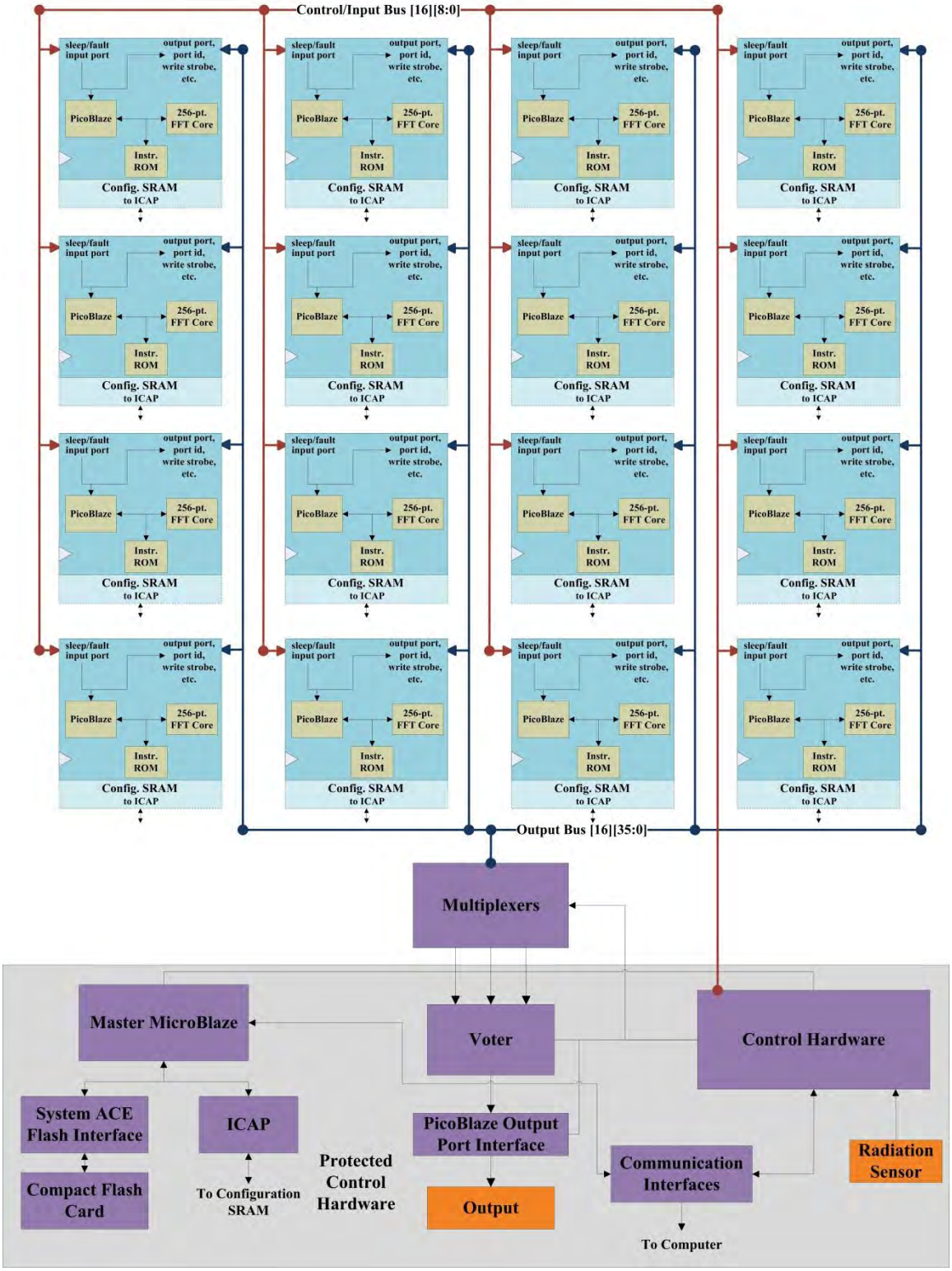

Figure 5.9: Block Diagram of the 16-Tile PicoBlaze + FFT Core System.

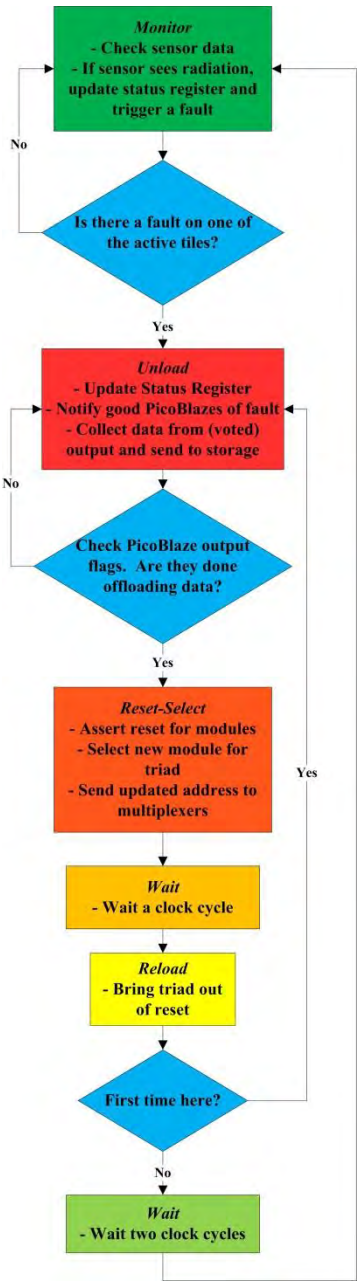

Figure 5.10: Flow chart describing the recovery process for the 16-Tile PicoBlaze + FFT Core System (each block corresponds roughly to one state in the recovery state machine).

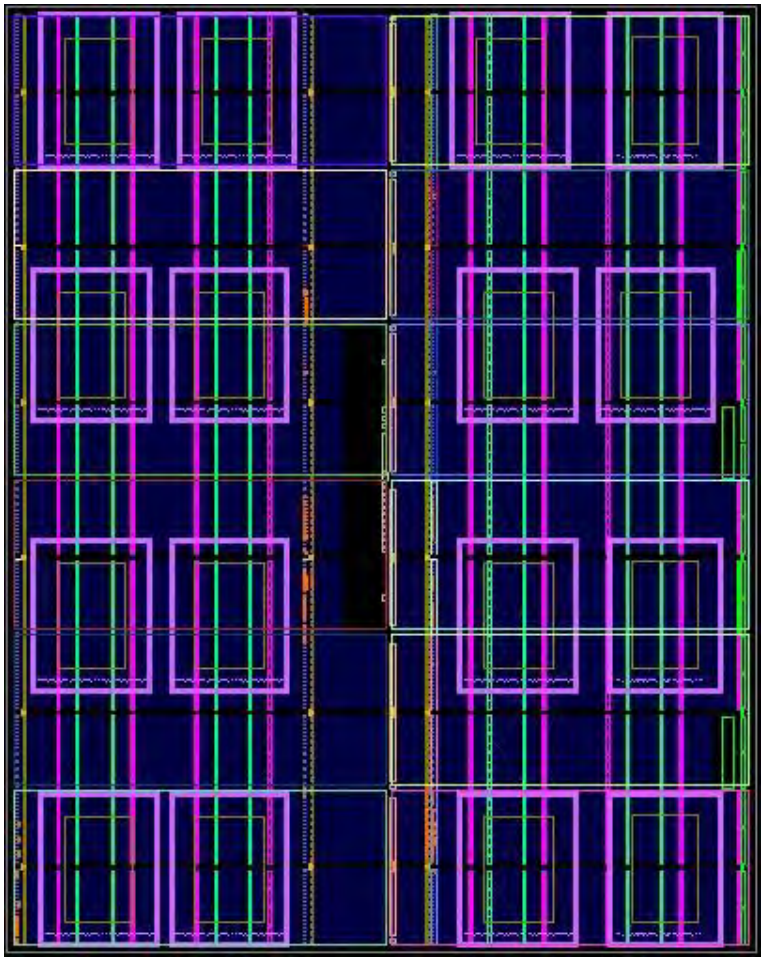

Figure 5.11: The floor plan of the 16-Tile PicoBlaze System, obtained from PlanAhead. Each pink rectangle is a partially reconfigurable region, able to hold one PicoBlaze + FFT core module.

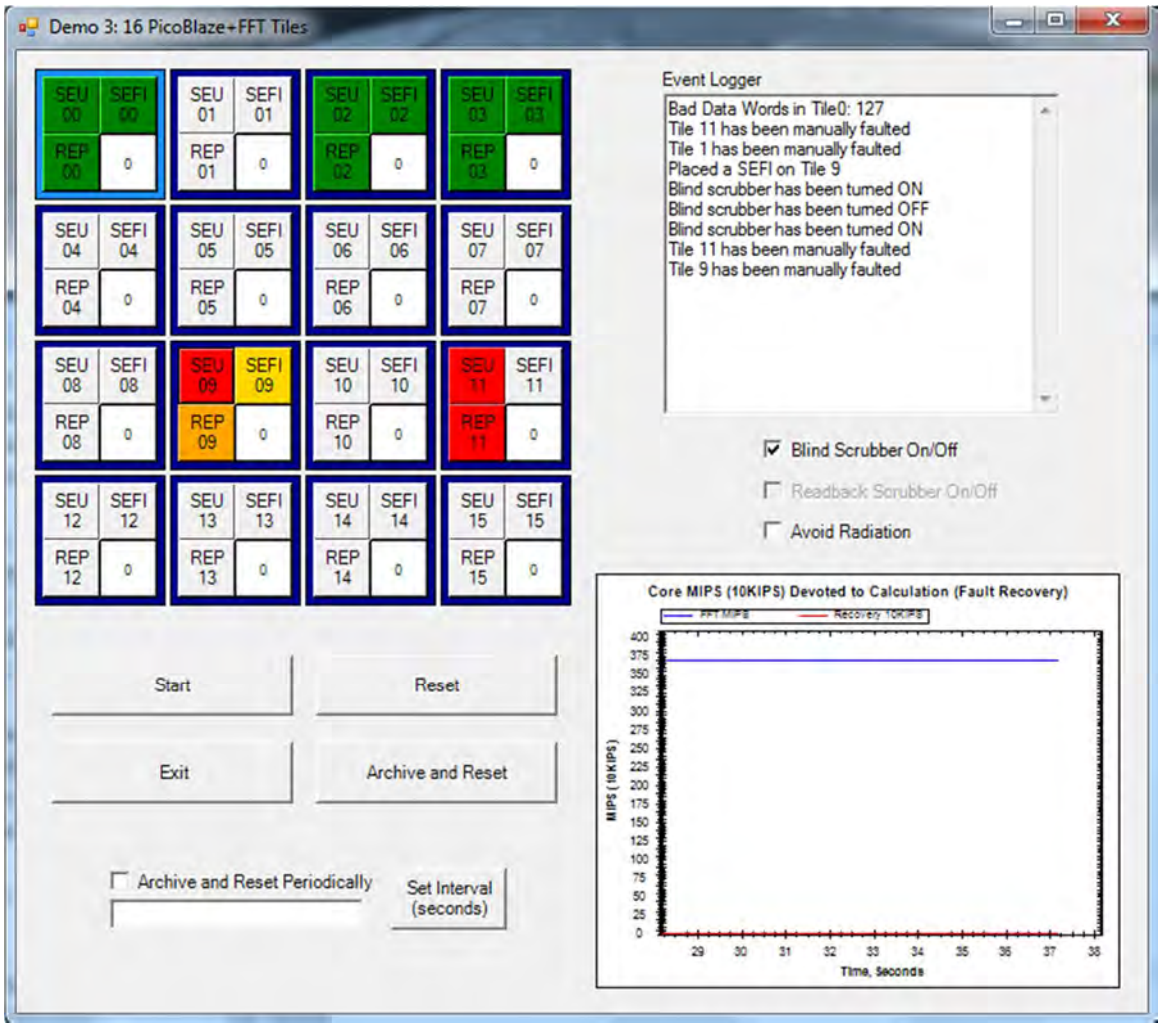

Figure 5.12: The 16-Tile PicoBlaze + FFT Core System's graphical user interface.

16-Tile MicroBlaze System

The last of the four demonstration systems is based on the MicroBlaze soft processor, another Xilinx product which is larger and more capable than the PicoBlaze. Although every one of our demonstration systems has a master MicroBlaze that functions as part of the scrubbing and control circuitry, in this system the computational tiles are also MicroBlaze processors. Each tile MicroBlaze has 32 KB of memory, drawn from the block RAMs inside the FPGA. In our demonstration system, they run a program

which calculates digits of pi. After being voted on, the resulting outputs are placed in memory and cycled across the ML605 board's LED lights. Like the PicoBlaze systems, this system reports the number of successful computation cycles and spare swaps that occur during each 250 ms period to the GUI, so that the performance graph can be updated. See Figure 5.13 for a block diagram of the system, Figure 5.14 for a representation of the recovery process as a flow diagram, Figure 5.15 for the FPGA floorplan (showing the PRRs in which the counter tiles reside), and Figure 5.16 for the System Status and Control GUI.

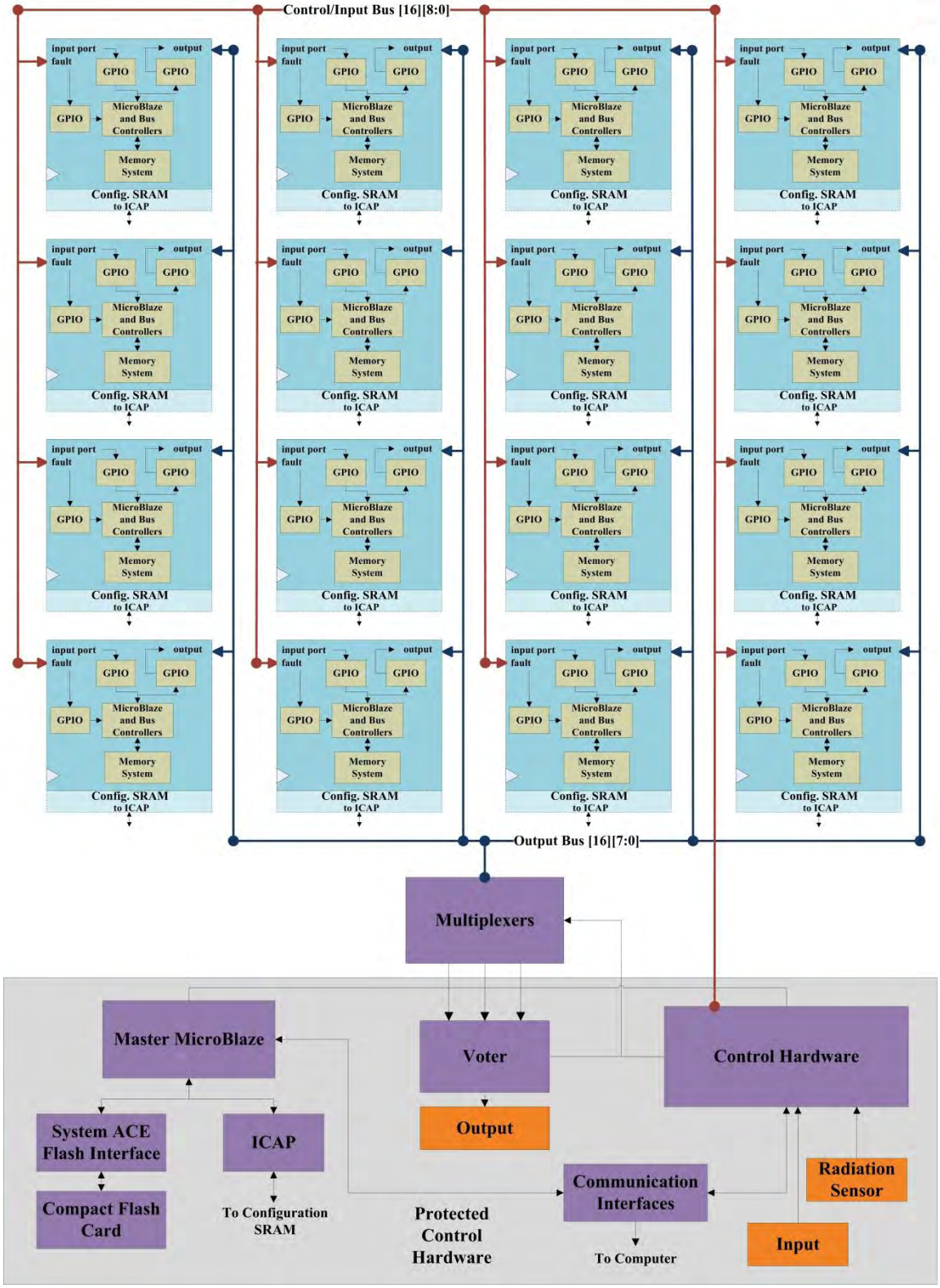

Figure 5.13: Block diagram of the MicroBlaze System.

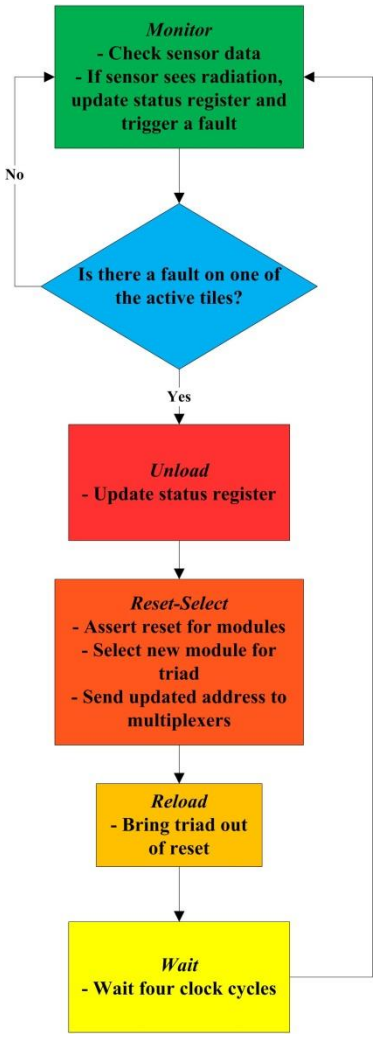

Figure 5.14: Flow chart describing the recovery process for the 16-Tile MicroBlaze System (each block corresponds roughly to one state in the recovery state machine).

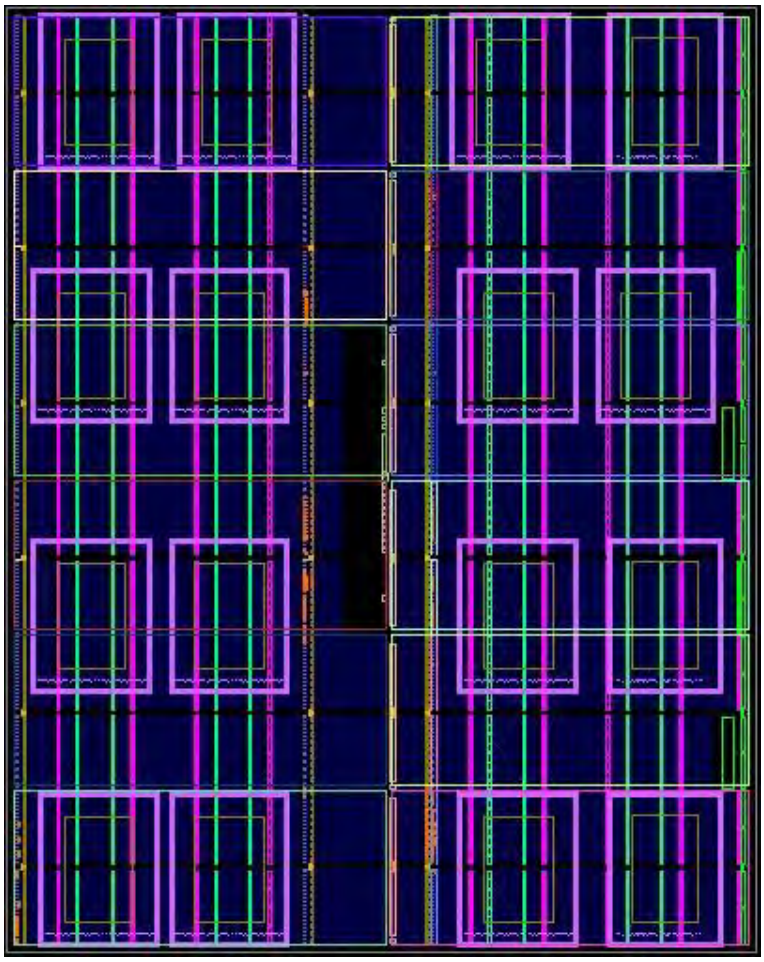

Figure 5.15: The floor plan of the 16-Tile MicroBlaze System, obtained from PlanAhead. Each pink rectangle is a partially reconfigurable region, able to hold one MicroBlaze module.

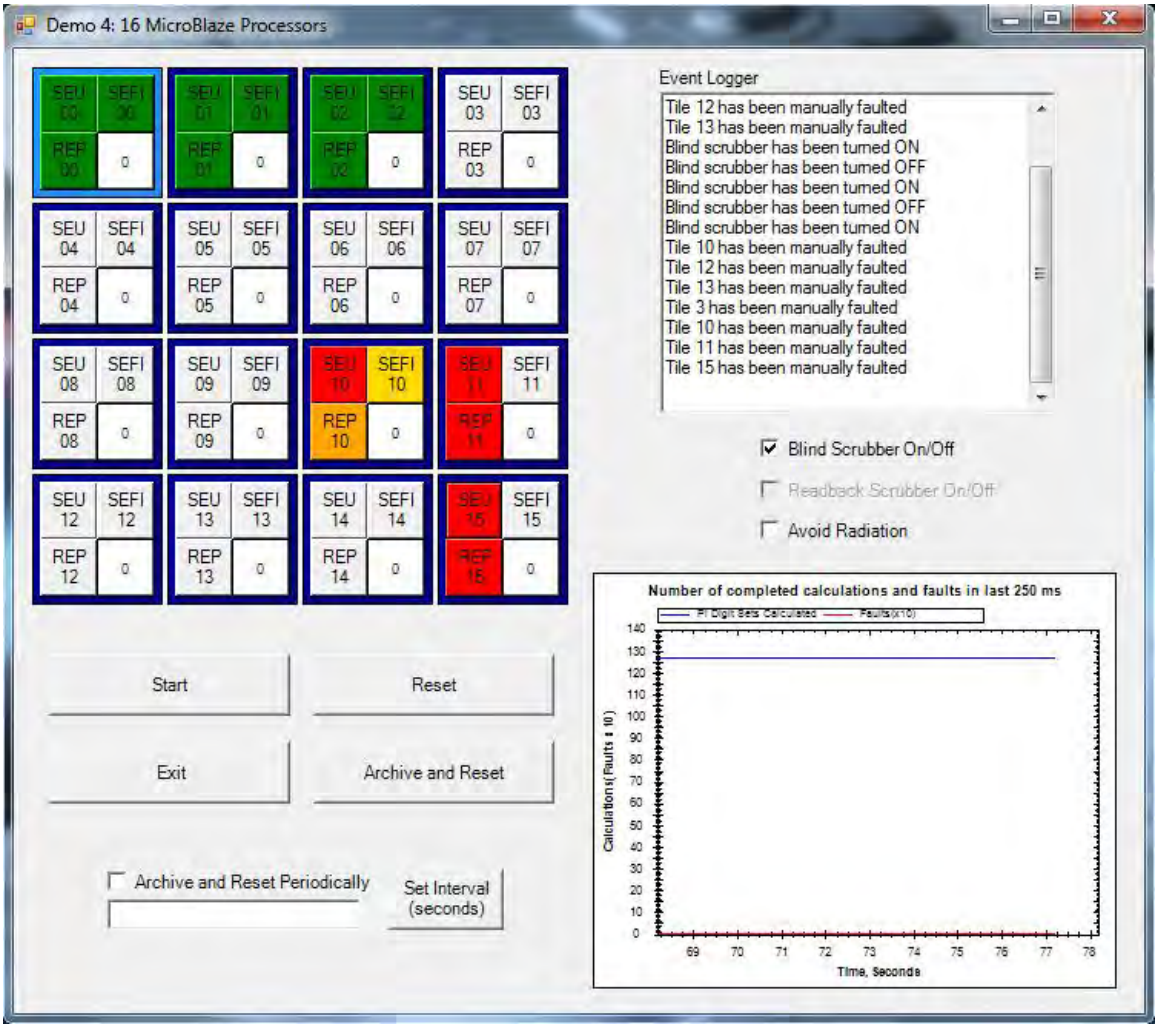

Figure 5.16: The 16-Tile MicroBlaze System's graphical user interface.

System Performance and Resource Utilization

The four systems listed above place different demands on the FPGA tools. Some require more space than others, and some have lower maximum clock rates due to more complex routing. Table 5.1 reveals the relevant statistics for each system; the reported maximum clock rate and the overall resource utilization percentage were obtained from PlanAhead's implemented design report summary. The maximum reported clock rate was not always attainable, and if implemented, could cause a failure to achieve timing

closure for the system during Place and Route. Therefore, the actual speed of the clock used to run each system is lower than the reported maximum in every case. For more detailed breakdowns of the resources used by each system, refer to Figures 5.16 – 5.19, which were also obtained from PlanAhead.

TABLE 5.1
Performance and Resource Utilization for Proof-of-Concept Systems

	64 Counters	36 PicoBlazes	16 PicoBlazes + FFT Cores	16 MicroBlazes
Reported Max. Clock Rate	71.664 MHz	63.399 MHz	52.743 MHz	78.388 MHz
Actual Clock Rate Used	66 MHz	50 MHz	30 MHz	30 MHz
Percent of Virtex-6 Resources Required	20.0%	17.0%	13.0%	28.0%

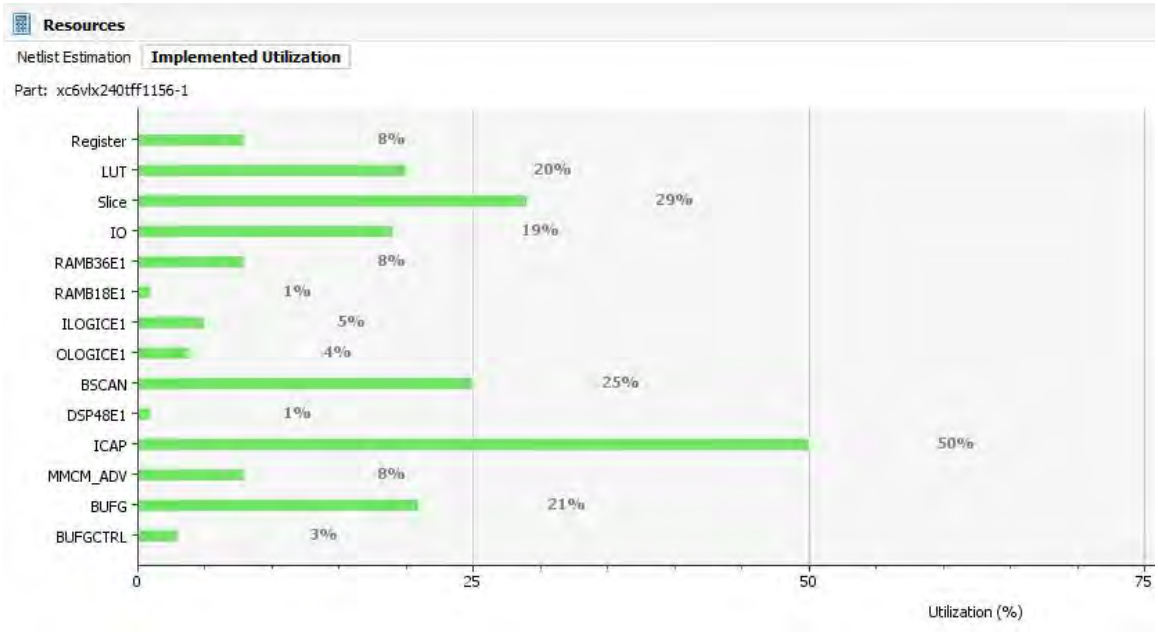

Figure 5.17: Resource utilization graph for the 64-Tile Binary Counter System.

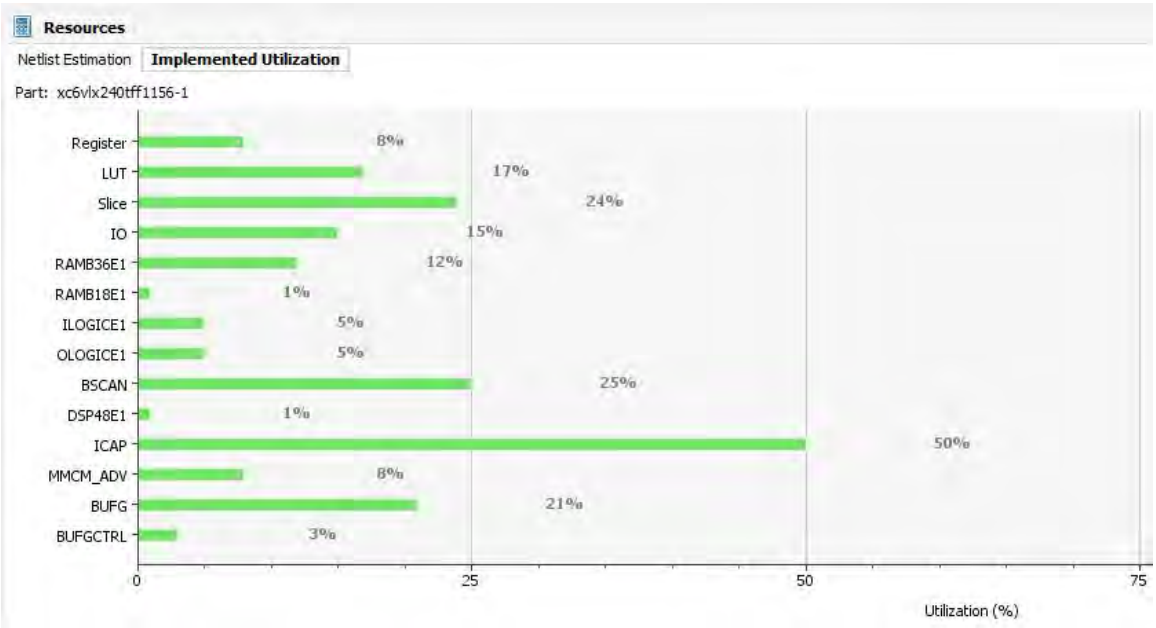

Figure 5.18: Resource utilization graph for the 36-Tile PicoBlaze System.

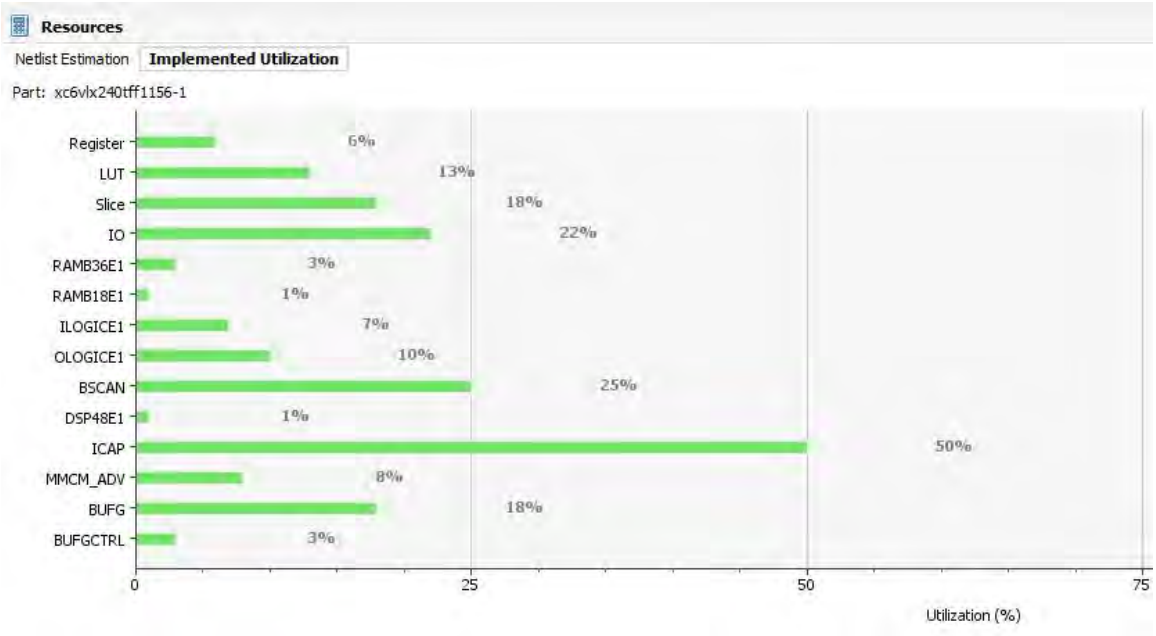

Figure 5.19: Resource utilization graph for the 16-Tile PicoBlaze + FFT Core System.

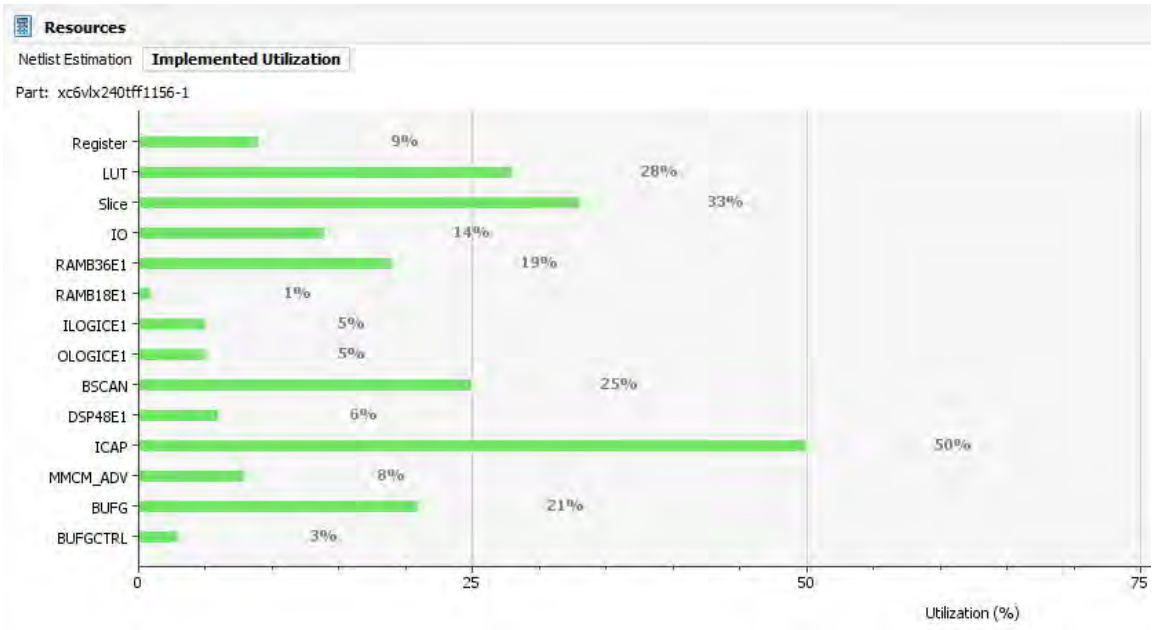

Figure 5.20: Resource utilization graph for the 16-Tile MicroBlaze System.

THEORETICAL ANALYSIS

The Markov Model

We chose to analyze our fault-tolerant systems for their reliability using Markov models. A Markov model describes a system as a combination of states and transitions between those states. Each transition is assigned a rate, depending on how often that transition happens. The mathematical representation of the model is a system of differential equations, which can be solved to determine the system's probability of reaching the failed state. From this, the overall failure rate and Mean Time Before Failure (MTBF) may be calculated.

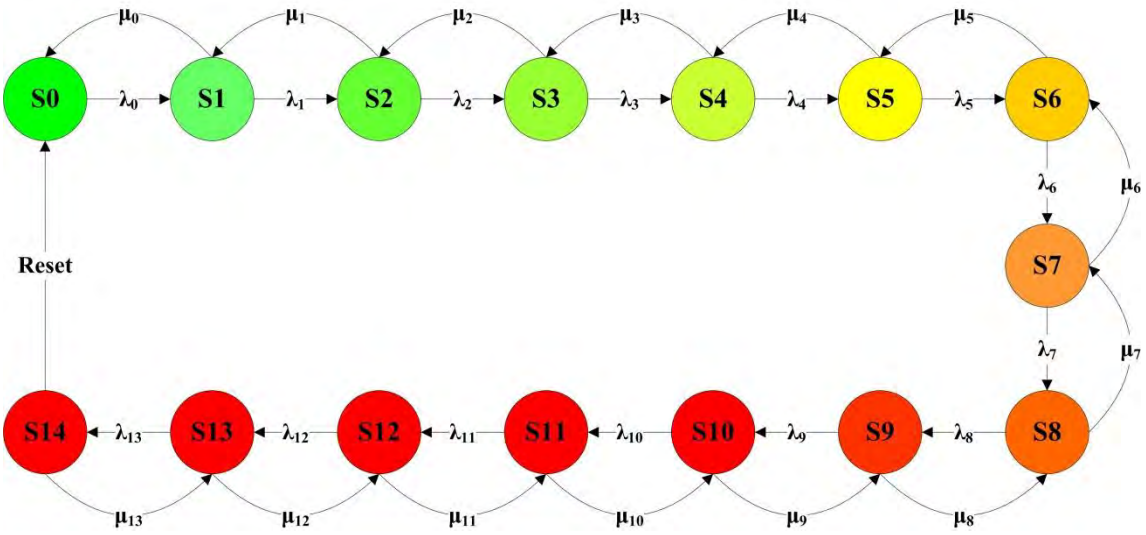

Figure 6.1: Diagrammatic representation of the Markov model for one of the sixteen-tile systems. S0 represents the initial state of the system, in which all tiles are in usable condition, and S14 represents the failure state, in which only two usable tiles remain. The system transitions to a state with a higher number at the failure rate, λ, and transitions to a state with a lower number at the repair rate, μ. Both of these may vary depending on the current state. The only essential difference between the 16-tile model and the others is the number of states.

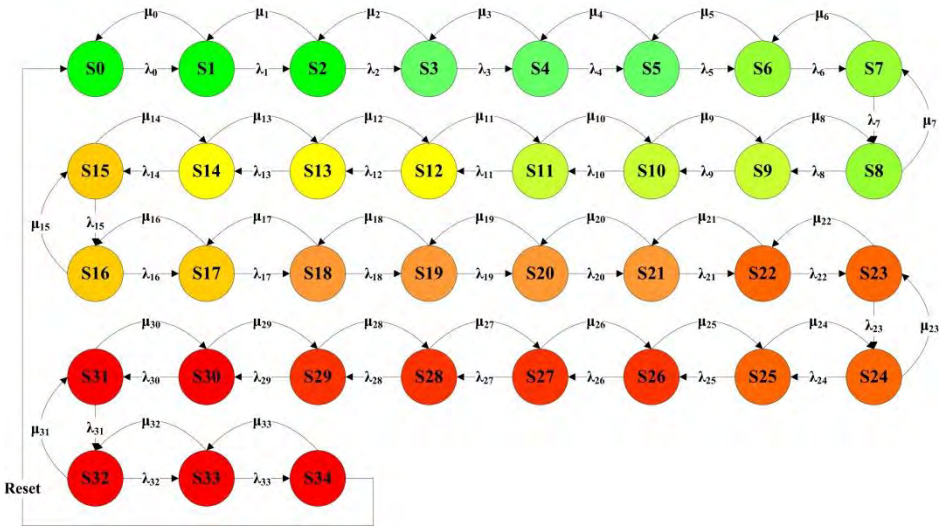

Figure 6.2: Diagram of the Markov model for the 36-Tile PicoBlaze System.

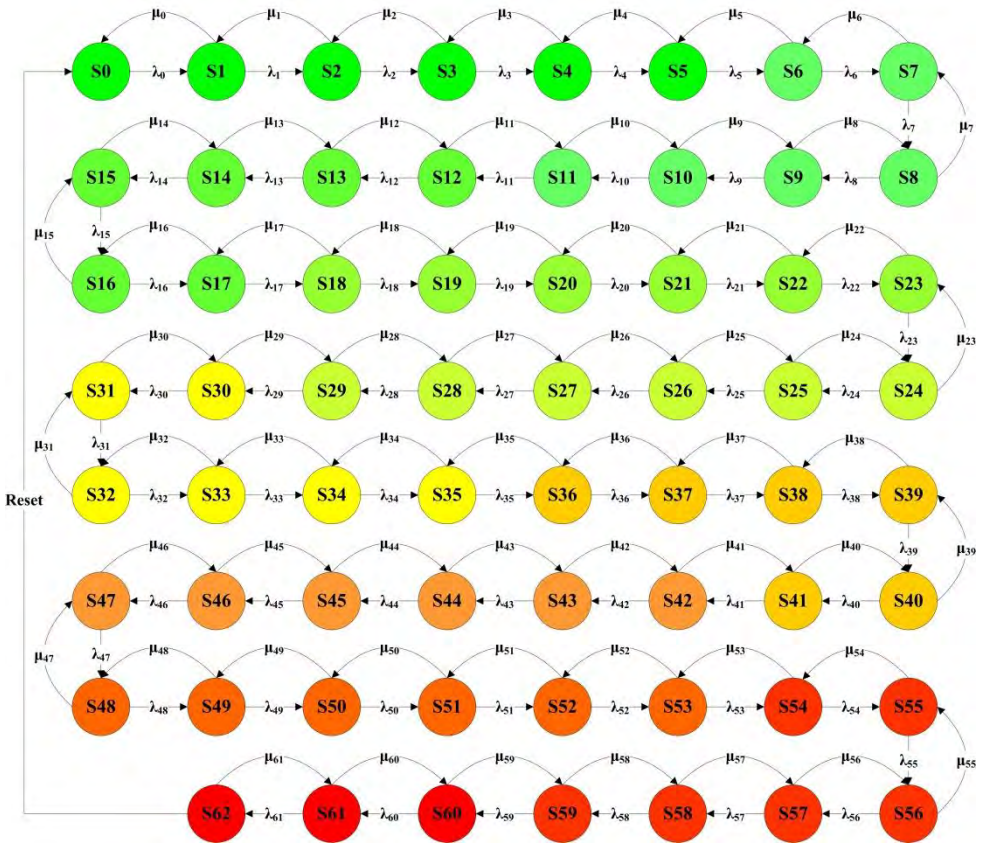

Figure 6.3: Diagram of the Markov model for the 64-Tile Binary Counter System.

In the Markov models for our systems, each state corresponds to some number of good spares. When only two undamaged tiles are left, the system is said to be in the failed state. The system may transition from its current state to a state with one less good spare, or a state with one more good spare. The latter transition occurs at the state's "repair rate," which is based on the amount of time that passes before the scrubber arrives at a bad tile and repairs it. The transition to a state with one less good spare occurs at that state's "fault rate." Our Markov calculations were based on the solution methods described in [33]; specifically, we calculated the steady-state solution using a matrix representation of the Markov equations. These calculations were implemented with a Matlab script.

Derivation of Parameters

In order to construct the Markov model, it was necessary to ensure that we had reasonable values for the fault and repair rates, corresponding to what we might expect from our physical system. Our methods of obtaining these parameters are the topic of this section.

Fault Rates (CREME96 Calculations)

Fault rates are based on realistic estimates of the faults our system's configuration SRAM would experience if it were riding on a spacecraft in various orbits, obtained from Vanderbilt University's CREME96 tool. Refer to the section above on the Orbital Environment Display for more information about CREME96 and the input data we used. Unlike the Orbital Environment Display, each Markov model uses an average fault rate

for the entire orbit, rather than attempting to break the orbit up into segments. However, the same four orbits were featured: the International Space Station (or Zarya) orbit, a Low Earth Orbit; Molniya 1-80, a Highly Elliptical Orbit; Satcom 5, a geosynchronous orbit; and EXP-1 Prime, the orbit of Montana State University's student-built cubesat, the Hiscock Radiation Belt Explorer. These will be hereinafter referred to as ISS, HEO, GEO, and HRBE, respectively.

TABLE 6.1
Orbital Fault Rates from CREME96, in Faults/Device/Second

	Average	Worst Week	Peak 5 Minutes
ISS	0.0003479	3.544	72.96
HEO	0.08788	120.2	2398
HRBE	0.003464	29.93	612.3
GEO	.0002494	149.8	3059

For the "average" environment calculations, the CREME model for solar maximum with peak protons and a stormy magnetosphere, without a flare event, was used. The "worst week" fault rate is based on a seven-day running average taken during the most intense part of a flare during solar maximum. Similarly, the "peak 5 minutes" fault rate comes from a running average taken during the most intense five minutes of a flare.

The fault rates obtained from CREME are reported in Table 6.1. They represent the number of faults per device per second; however, this is simply a "base" fault rate that must be modified depending on the nature of the system and the state it is currently in. For example, if we model one of our systems without the radiation sensor included,

we are only concerned about faults that occur in sensitive bits, i.e. those portions of the configuration SRAM which, if altered, would actually cause a malfunction in our design. However, if the sensor is present, it will give alarm and declare a tile bad whenever the FPGA is struck by radiation; the sensor has no way of discerning whether any particular particle struck a sensitive bit or not. Discovering how many sensitive bits are present in each of our systems would require extensive testing, so we chose to use a worst-case estimate of 35% sensitive bits. This is based on the empirically obtained sensitive bit densities for various circuits on a Virtex-4 FPGA, found in [34]. Motivated by these considerations, we reduced the fault rate by 65% in system variations without the sensor.

A further assumption made was that each radiation strike would affect one and only one tile. Since the tiles do not completely fill the FPGA in any of our systems, this assumption may be somewhat pessimistic – some high-energy particles would strike in completely empty areas of the FPGA, where they would be less likely to have any impact on the operation of the tiles. When modeling systems with only three tiles (see the Results and Discussion section), we assumed that the tiles would occupy a fraction of the FPGA equal to three divided by the maximum number of tiles for that system, and reduced the fault rate accordingly.

The fault rates require further adjustment depending on the state of the system. As the number of damaged tiles grows, the rate of new tile failures must drop, since the radiation has a chance of passing through a previously damaged region of the device. Therefore, as the system moves to progressively more damaged states, the fault rate is reduced in proportion to the number of good tiles remaining.

Repair Rates

The amount of time needed to repair a tile through partial reconfiguration was derived empirically for each system; the results of these measurements are reported in Table 6.2. In each system, we created a signal that would be driven high when the scrubber began refreshing a tile, then driven low again when scrubbing concluded. This signal was connected to an external pin on the FPGA board and monitored with an oscilloscope, and average times needed to scrub the tiles were measured. Within a system, the actual time needed to scrub a given tile varies, due to differences in tile placement within the FPGA, so the scrub times given in Table 6.2 are averages. Scrubbing time also varies by system, since some systems have larger tiles than others. In systems with very short scrub times, a lower limit on the scrub time is imposed by the timer interrupt that initiates scrubbing (hence the appearance of multiples of 250 ms in the data).

TABLE 6.2
Spare Swap and Scrubbing Times

	64 Counters	36 PicoBlazes	16 PicoBlazes + FFT Cores	16 MicroBlazes
Clock Speed	66 MHz	50 MHz	30 MHz	30 MHz
Tile Swap	106 ns	45.38 us	992 us	267 ns
Blind Scrub (one tile)	250 ms	500 ms	2.07 s	2.63 s
Readback Scrub (one tile)	500 ms /500 ms	1 s /1.5 s	4.11 s /6.13 s	2.88 s /5.30 s
Blind Scrub (all tiles)	16 s	18 s	33.12 s	42.08 s
Readback Scrub (all tiles)	32 s / 32 s	36 s /54 s	65.76 s /98.08 s	46.08 s / 84.8 s

The two numbers in the Readback Scrub rows represent the time needed to scrub undamaged/damaged tiles, respectively.

Scrubbing a tile with readback takes longer than scrubbing it blindly, and the readback scrubber takes even longer if it finds damage and must repair the tile. For this reason, minimum and maximum values are reported for the readback scrubber in Table I, corresponding to scrubs of undamaged and damaged tiles, respectively. We believe that the time required to read the “golden data” from the compact flash card is a substantial contributor to scrubbing time. Finding an alternate method of data storage for the bit files, in order to shorten the scrubbing cycle, should be a high priority for future work.

The time needed to scrub one tile is not necessarily equivalent to the actual repair time; this assumption can only be made if the scrubber is aware of which tiles are damaged and can prioritize them for immediate scrubbing. Thus, setting the repair rate equal to the single-tile scrubbing rate (for a damaged tile) is appropriate if and only if the

radiation sensor is present in the system. If the scrubber is constrained to travel sequentially through the FPGA, then the average repair rate will depend on the number of damaged tiles. A greater frequency of damaged tiles within the set of all tiles increases the probability that the scrubber will locate and repair one on its next move, raising the overall repair rate. However, unless all tiles are damaged, the average repair rate can never be as great as the single-tile scrubbing rate.

Results and Discussion

We wished to model several variations of our designs, in order to determine how the inclusion of certain features affected their reliability. The first design variation contained only three active tiles and no spares; it represents the behavior of our designs if we had implemented them as traditional TMR plus scrubbing systems. The second design variation includes the maximum number of spare tiles we were able to implement (sixty-four, thirty-six, or sixteen, depending on the system), but does not include the radiation sensor. Since the sensor is not present to alert the scrubber to potentially damaged tiles, the scrubber must travel sequentially, and a repair rate that varies with the number of damaged tiles is used. The base fault rate is also reduced by 65% to reflect the percentage of sensitive bits. The third design variation incorporates the maximum number of spare tiles and the sensor. Since we do not yet have a good estimate of the sensor's accuracy and sensitivity, we assumed for the purposes of this calculation that the sensor is able to detect all damaging radiation strikes. Therefore, we used a constant repair rate equal to the single-tile scrubbing rate in the models for this design variation. We also used the standard base fault rate, without adjusting for sensitive bits, since the

sensor will declare tiles damaged regardless of which bits are hit. Our desire to model each of our four demonstration systems on each of the four example orbits produced sixteen different Markov models for each design variation, for a grand total of forty-eight models.

TABLE 6.3
MTBF of 64-Tile Counter System (Theoretical), in Seconds

			TMR Only (Baseline)	TMR + Spares	TMR, Spares, and Sensor	Percent Improvement from Spares	Percent Improvement from Spares and Sensor
Average	Blind	ISS	5.12E+09	6.77E+231	2.20E+274	1.32E+222%	4.29E+264
		HEO	8.03E+04	1.28E+85	4.93E+125	1.59E+80%	6.14E+120
		HRBE	5.16E+07	1.07E+170	2.92E+212	2.07E+162%	5.67E+204
		GEO	9.95E+09	6.06E+240	1.98E+283	6.09E+230%	1.99E+273
	RB	ISS	2.56E+09	3.00E+213	9.55E+255	1.17E+204%	3.73E+246
		HEO	4.02E+04	5.19E+68	4.30E+107	1.29E+64%	1.07E+103
		HRBE	2.58E+07	5.76E+151	1.30E+194	2.23E+144%	5.05E+186
		GEO	4.98E+09	2.67E+222	8.59E+264	5.36E+212%	1.72E+255
Worst Week	Blind	ISS	5.15E+01	4.98E+17	5.03E+35	9.67E+15%	9.77E+33
		HEO	1.06E-01	4.62E+01	1.26E+02	4.35E+02%	1.19E+03
		HRBE	9.46E-01	4.91E+04	1.92E+08	5.19E+04%	2.03E+08
		GEO	7.83E-02	2.41E+01	4.02E+01	3.07E+02%	5.12E+02
	RB	ISS	2.68E+01	1.46E+12	1.58E+24	5.45E+10%	5.88E+22
		HEO	8.49E-02	1.54E+01	1.26E+01	1.80E+02%	1.47E+02
		HRBE	6.00E-01	1.45E+03	5.19E+04	2.42E+03%	8.64E+04
		GEO	6.46E-02	9.85E+00	6.55E+00	1.51E+02%	1.00E+02
Peak 5 Minutes	Blind	ISS	2.21E-01	2.92E+02	4.20E+03	1.32E+03%	1.90E+04
		HEO	3.14E-03	2.77E-01	1.03E-01	8.72E+01%	3.17E+01
		HRBE	1.40E-02	1.51E+00	6.90E-01	1.07E+02%	4.83E+01
		GEO	2.56E-03	2.12E-01	7.75E-02	8.18E+01%	2.93E+01
	RB	ISS	1.63E-01	5.19E+01	9.12E+01	3.17E+02%	5.59E+02
		HEO	3.14E-03	2.62E-01	9.42E-02	8.24E+01%	2.90E+01
		HRBE	1.31E-02	1.21E+00	4.77E-01	9.14E+01%	3.54E+01
		GEO	2.52E-03	2.03E-01	7.25E-02	7.96E+01%	2.78E+01

TABLE 6.4
MTBF of 36-Tile PicoBlaze System (Theoretical), in Seconds

			TMR Only (Baseline)	TMR + Spares	TMR, Spares, and Sensor	Percent Improvement from Spares	Percent Improvement from Spares and Sensor
Average	Blind	ISS	1.44E+09	1.65E+123	1.49E+139	1.15E+114%	1.03E+130
		HEO	2.26E+04	9.61E+42	6.81E+57	4.25E+38%	3.01E+53
		HRBE	1.45E+07	2.16E+89	1.77E+105	1.49E+82%	1.22E+98
		GEO	2.80E+09	1.33E+128	1.21E+144	4.75E+118%	4.33E+134
	RB	ISS	5.40E+08	2.44E+112	1.74E+129	4.52E+103%	3.22E+120
		HEO	8.54E+03	3.21E+33	1.72E+48	3.76E+29%	2.01E+44
		HRBE	5.45E+06	3.68E+78	2.13E+95	6.75E+71%	3.90E+88
		GEO	1.05E+09	1.97E+117	1.41E+134	1.88E+108%	1.35E+125
Worst Week	Blind	ISS	1.60E+01	8.06E+07	2.27E+12	5.04E+06%	1.42E+11
		HEO	7.54E-02	4.09E+00	1.83E+00	5.32E+01%	2.33E+01
		HRBE	4.49E-01	9.10E+01	1.37E+02	2.02E+02%	3.05E+02
		GEO	5.86E-02	2.93E+00	1.24E+00	4.90E+01%	2.01E+01
	RB	ISS	7.35E+00	1.45E+05	7.60E+07	1.97E+04%	1.03E+07
		HEO	6.80E-02	2.95E+00	1.19E+00	4.24E+01%	1.65E+01
		HRBE	3.27E-01	2.55E+01	1.88E+01	7.70E+01%	5.66E+01
		GEO	5.37E-02	2.25E+00	8.82E-01	4.09E+01%	1.54E+01
Peak 5 Minutes	Blind	ISS	1.37E-01	9.82E+00	5.48E+00	7.07E+01%	3.90E+01
		HEO	3.14E-03	1.18E-01	4.17E-02	3.66E+01%	1.23E+01
		HRBE	1.29E-02	5.03E-01	1.83E-01	3.80E+01%	1.32E+01
		GEO	2.51E-03	9.20E-02	3.24E-02	3.57E+01%	1.19E+01
	RB	ISS	1.17E-01	5.72E+00	2.59E+00	4.79E+01%	2.11E+01
		HEO	3.14E-03	1.16E-01	4.09E-02	3.59E+01%	1.20E+01
		HRBE	1.26E-02	4.72E-01	1.70E-01	3.65E+01%	1.25E+01
		GEO	2.50E-03	9.09E-02	3.19E-02	3.54E+01%	1.18E+01

TABLE 6.5
MTBF of 16-Tile PicoBlaze + FFT Core System (Theoretical), in Seconds

			TMR Only (Baseline)	TMR + Spares	TMR, Spares, and Sensor	Percent Improvement from Spares	Percent Improvement from Spares and Sensor
Average	Blind	ISS	1.55E+08	3.88E+45	1.42E+48	2.50E+37%	9.19E+39
		HEO	2.51E+03	1.48E+13	1.36E+15	5.90E+09%	5.43E+11
		HRBE	1.56E+06	4.61E+31	1.59E+34	2.96E+25%	1.02E+28
		GEO	3.01E+08	4.06E+47	1.50E+50	1.35E+39%	4.97E+41
	RB	ISS	5.86E+07	1.40E+41	1.92E+44	2.39E+33%	3.28E+36
		HEO	1.01E+03	6.51E+09	6.43E+11	6.45E+06%	6.36E+08
		HRBE	5.93E+05	1.90E+27	2.27E+30	3.20E+21%	3.82E+24
		GEO	1.14E+08	1.47E+43	2.02E+46	1.29E+35%	1.77E+38
Worst Week	Blind	ISS	3.64E+00	1.28E+02	6.59E+01	3.42E+01%	1.71E+01
		HEO	6.46E-02	7.52E-01	2.65E-01	1.06E+01%	3.09E+00
		HRBE	2.75E-01	3.51E+00	1.26E+00	1.18E+01%	3.58E+00
		GEO	5.17E-02	5.97E-01	2.10E-01	1.05E+01%	3.06E+00
	RB	ISS	2.71E+00	5.06E+01	2.31E+01	1.77E+01%	7.54E+00
		HEO	6.40E-02	7.31E-01	2.57E-01	1.04E+01%	3.02E+00
		HRBE	2.63E-01	3.13E+00	1.12E+00	1.09E+01%	3.28E+00
		GEO	5.11E-02	5.84E-01	2.05E-01	1.04E+01%	3.02E+00
Peak 5 Minutes	Blind	ISS	1.08E-01	1.28E+00	4.52E-01	1.09E+01%	3.18E+00
		HEO	3.14E-03	3.60E-02	1.26E-02	1.05E+01%	3.01E+00
		HRBE	1.26E-02	1.42E-01	4.97E-02	1.03E+01%	2.94E+00
		GEO	2.49E-03	2.83E-02	9.90E-03	1.04E+01%	2.98E+00
	RB	ISS	1.06E-01	1.22E+00	4.32E-01	1.05E+01%	3.07E+00
		HEO	3.14E-03	3.60E-02	1.26E-02	1.05E+01%	3.01E+00
		HRBE	1.26E-02	1.41E-01	4.94E-02	1.02E+01%	2.92E+00
		GEO	2.49E-03	2.80E-02	9.80E-03	1.02E+01%	2.94E+00

TABLE 6.6
MTBF of 16-Tile MicroBlaze System (Theoretical), in Seconds

			TMR Only (Baseline)	TMR + Spares	TMR, Spares, and Sensor	Percent Improvement from Spares	Percent Improvement from Spares and Sensor
Average	Blind	ISS	1.22E+08	1.73E+44	6.35E+46	1.42E+36%	5.20E+38
		HEO	1.99E+03	1.26E+12	8.67E+13	6.33E+08%	4.36E+10
		HRBE	1.23E+06	2.12E+30	7.20E+32	1.72E+24%	5.86E+26
		GEO	2.37E+08	1.81E+46	6.66E+48	7.64E+37%	2.81E+40
	RB	ISS	7.12E+07	6.03E+42	1.95E+46	8.47E+34%	2.74E+38
		HEO	1.20E+03	8.92E+10	3.12E+13	7.43E+07%	2.60E+10
		HRBE	7.20E+05	3.50E+28	2.23E+32	4.86E+22%	3.10E+26
		GEO	1.38E+08	6.31E+44	2.05E+48	4.57E+36%	1.48E+40
Worst Week	Blind	ISS	3.32E+00	9.07E+01	4.20E+01	2.63E+01%	1.17E+01
		HEO	6.43E-02	7.44E-01	2.61E-01	1.06E+01%	3.07E+00
		HRBE	2.71E-01	3.36E+00	1.20E+00	1.14E+01%	3.42E+00
		GEO	5.14E-02	5.92E-01	2.08E-01	1.05E+01%	3.04E+00
	RB	ISS	2.84E+00	6.31E+01	3.64E+01	2.12E+01%	1.18E+01
		HEO	6.40E-02	7.35E-01	2.60E-01	1.05E+01%	3.07E+00
		HRBE	2.64E-01	3.22E+00	1.18E+00	1.12E+01%	3.47E+00
		GEO	5.11E-02	5.87E-01	2.07E-01	1.05E+01%	3.06E+00
Peak 5 Minutes	Blind	ISS	1.07E-01	1.26E+00	4.43E-01	1.08E+01%	3.14E+00
		HEO	3.14E-03	3.60E-02	1.26E-02	1.05E+01%	3.01E+00
		HRBE	1.26E-02	1.41E-01	4.96E-02	1.02E+01%	2.94E+00
		GEO	2.49E-03	2.83E-02	9.90E-03	1.04E+01%	2.98E+00
	RB	ISS	1.06E-01	1.23E+00	4.40E-01	1.06E+01%	3.15E+00
		HEO	3.14E-03	3.60E-02	1.26E-02	1.05E+01%	3.01E+00
		HRBE	1.26E-02	1.41E-01	4.95E-02	1.02E+01%	2.93E+00
		GEO	2.49E-03	2.80E-02	9.90E-03	1.02E+01%	2.98E+00

Each model was used to calculate the Mean Time Before Failure (MTBF) for the corresponding system and orbit. The results appear in Tables 6.3 – 6.6. Table 6.3 contains the results for the 64-Tile Counter System, Table 6.4 those for the 36-Tile PicoBlaze system, Table 6.5 those for the 16-Tile PicoBlaze-plus-FFT system, and Table 6.6 those for the 16-Tile MicroBlaze system. In each table, the fourth data column gives the percent change in MTBF that occurs when the maximum number of spares is added to the system without spares. The fifth (rightmost) data column displays the percent change that takes place when both the maximum number of spares and the sensor are added to the system without spares.

The results of these theoretical calculations reveal that the inclusion of many spares in the TMR system yields dramatic improvements in its reliability under standard orbital conditions, increasing the MTBF by many orders of magnitude in some cases. The sensor yields more modest, but still considerable, improvements in reliability under standard orbital conditions, as well as Worst Week conditions in some of the more protected orbits. A positive, but very small improvement is obtained from addition of the spares-sensor combination when the system is under severe stress. This is not unexpected, since the system failure rate is still significantly greater than the scrubbing rate under these conditions, even if the improved scrubbing rate occasioned by the addition of the sensor is used.

Under standard space weather conditions, many of the systems (including all of the systems with numerous spares) could theoretically operate for decades without experiencing a complete failure. However, like many extant spacecraft systems, they

would need to be turned off temporarily during periods of intense solar activity, which could cause failures in as little as a fraction of a second.

EXPERIMENTS

Basic Verification

The four demonstration systems were downloaded to the ML605 board and tested to ensure proper operation. The output of each system was viewed on the ML605 board's LEDs and/or fetched and printed in a terminal program by the master MicroBlaze, and checked for correctness based on the inputs. Since the System Status and Control GUI displays all active, damaged, and dormant tiles, it was used to verify that the TMR system could correctly identify damaged tiles and swap in spares as needed. It was also used to observe the operation of the scrubber and determine that it was correctly walking through the tiles, detecting errors (in the case of the readback scrubber) and repairing damage. In addition to these methods, the Chipscope program was used to monitor a number of crucial signals inside the FPGA and ensure that they maintained the correct values. The fault injection commands made available by the System Status and Control GUI were used to introduce simulated SEUs and SEFIs in a controlled fashion, so that the system's response to them could be verified.

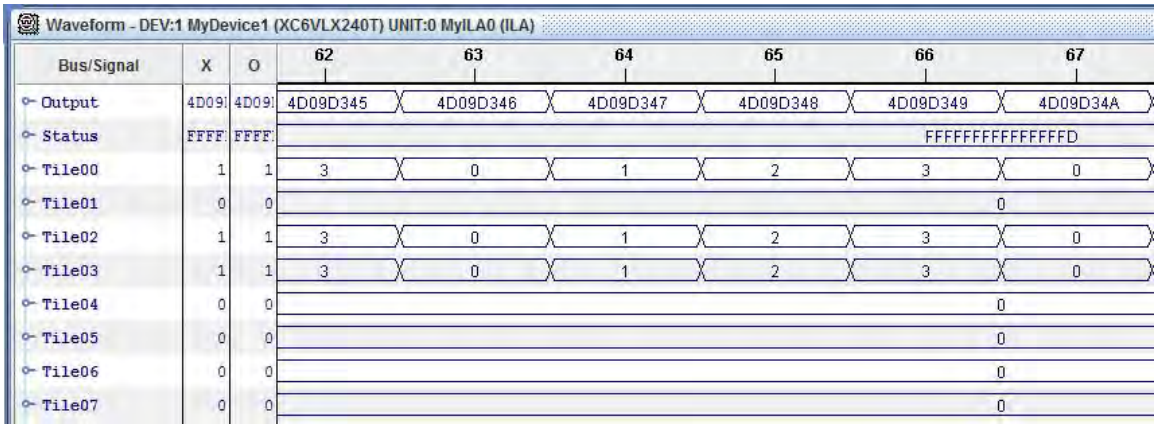

Figure 7.1: A portion of the Chipscope window, monitoring the 64-Tile Binary Counter System. The complete 32-bit voter output and the status register appear at the top; below, the two least significant bits of the output of each tile are shown. Tile 01 has been subjected to a forced SEU fault.

Verifying the sensor interface presented some special challenges, since our research team is not yet equipped to bombard the sensor with actual ionizing radiation. A special version of the amplifier board was fabricated, with its signal gains set in such a way that the sensor's responses to light from a laser pointer would register as radiation strikes. This amplifier board was used to connect the sensor to the ML605 board for testing. Using standard red and infrared laser pointers, we were able to determine that the fault-tolerant architecture on the FPGA responded appropriately to radiation strike reports from the sensor.

Eventually, the sensor and its interface with the FPGA architecture were tested at Texas A&M University's Radiation Effects Testing Facility. Since characterizing the sensor's response was the primary goal of the facility visit, the sensor interface was connected to simple FPGA circuitry that sent the data to a computer for display, rather than the fault-tolerant architecture. Nonetheless, the tests demonstrated the functionality

of the sensor; coupled with the previous laser pointer tests, they show that the system can detect radiation strikes and respond appropriately. The test consisted of striking the sensor with pulses from a 25 MeV beam of Krypton ions. The numbers of strikes registered by the pixel counters were sent through the USB interface to a computer and logged. The sensor was tested with an amplifier board whose gains varied by pixel, granting information on the optimal gain level for detecting heavy ions. So long as the correct gains were used, the sensor was definitely able to detect the heavy ions and report their strike locations to the FPGA.

Measurement of the MTBF

In order to obtain an empirical plot of the relationship between fault rate and mean time before failure (MTBF), we modified the 64-tile counter system to include a timer that would cause a simulated SEU fault in a random tile after each tick. The addition of the timer necessitated a reduction in the system's clock speed from 66 MHz to 40 MHz, so these experimental results are somewhat more pessimistic than the theoretical Markov model results for this system. The system was set up with a given fault rate and allowed to run until failure, with the blind scrubber active, and the time taken to reach the failure state was measured. Since the time to failure was variable, sixteen trials were performed for each fault rate and the results were averaged. The scrubber was required to travel sequentially through the tiles, i.e. the sensor was not included in the system for this measurement. A plot of the MTBF vs. fault rate appears in Figure 7.2.

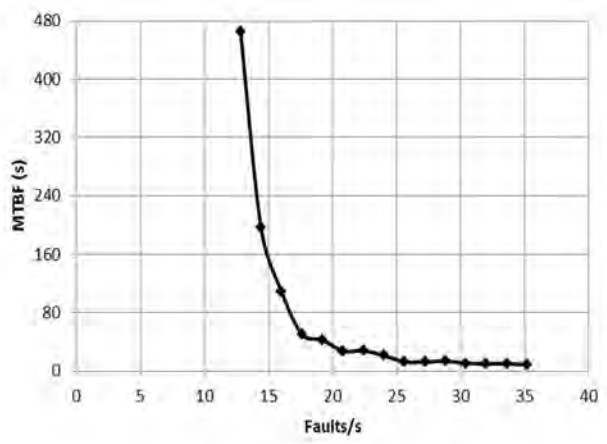

Figure 7.2: The measured MTBF of the 64-tile Binary Counter System with the blind scrubber, running at 40 MHz, vs. fault rate.

Overhead vs. Number of Spares

We wished to know how the addition of numerous spares to the system might impact resource utilization and maximum clock speed. We devised an experiment to determine this, but only performed it on the 64-tile counter system due to time constraints. First, an optimal size for the partially reconfigurable regions (PRRs), based on what PRR size would yield the best clock speed for the 64-tile system, was chosen by trial-and-error. Certain features that were included for demonstration purposes only (e.g. the USB interface for communicating with the GUI) were removed from the system to reduce its complexity and ensure that only essential features were being tested. Then the system was recreated with only four tiles. Additional spare tiles were added one at a time, and after each tile addition, the system was re-synthesized and put through all steps of the development process, up to and including Place and Route in PlanAhead. The addition of each new tile required the creation of a new partially reconfigurable region in the floor plan; the initial four were clustered near the center of the FPGA, and new ones

were added around them, spiraling outward from the center. See Figure 7.3 for several example floor plans featuring different numbers of tiles. After the PAR tool completed its work on each system, the maximum clock rate and resource utilization numbers reported by PlanAhead were collected. Since the MAP and PAR tools are deterministic (i.e. multiple runs on the same system and floor plan yield the same results), only one test was performed for each number of tiles.

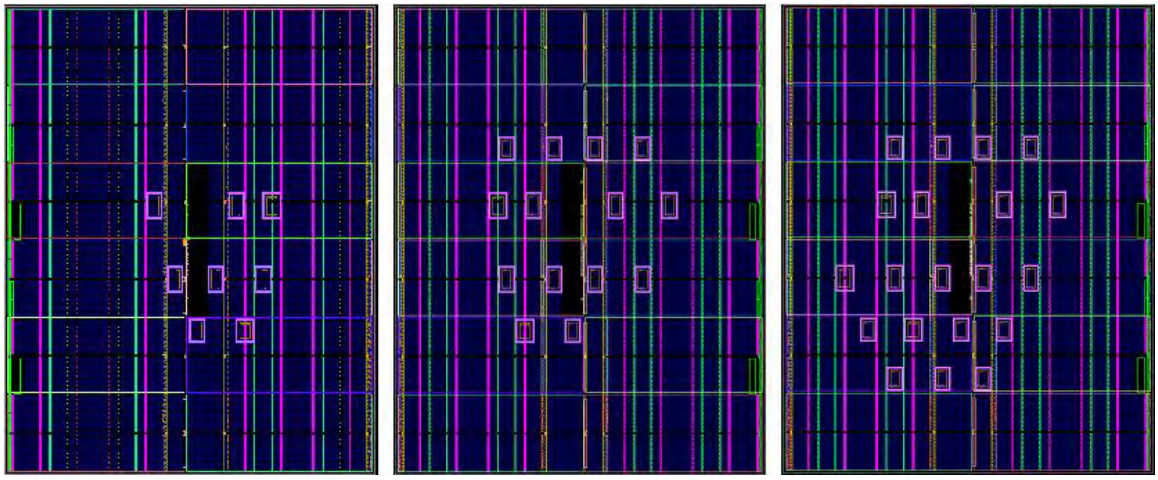

Figure 7.3: Several floor plans from the overhead test Binary Counter System, showing the gradually increasing number of spares.

Refer to figures 7.4 and 7.5 for the results of the overhead experiment. Figure 7.4 demonstrates that adding spares to the system increases the percentage of FPGA resources consumed, a fairly obvious result. Since the tiles in this system are simple and small, the percentage of resources needed goes up slowly with the addition of spares; the rate of increase would no doubt be different for systems with larger individual tiles. No clear relationship between the number of spares and the maximum clock rate is revealed by the results in Figure 7.5, although the exact routing (and thus the clock rate) changes a great deal as new tiles are added to the system.

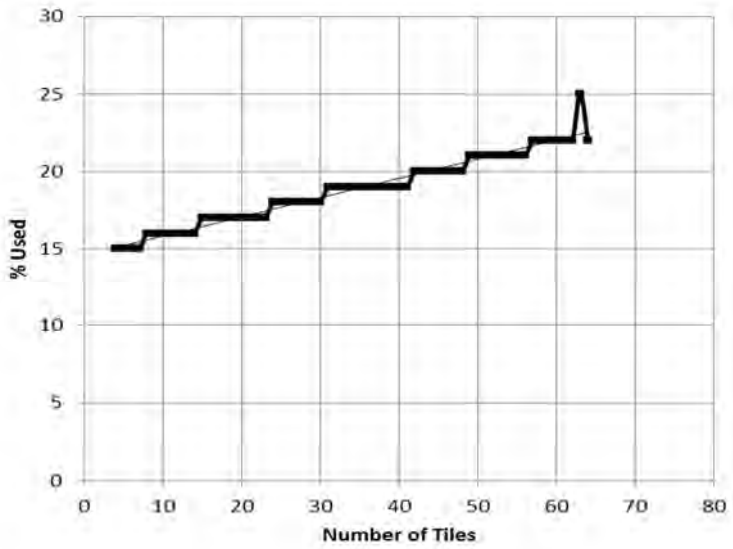

Figure 7.4: Percentage of FPGA resources used by variations of the Binary Counter System with different numbers of tiles. PlanAhead reports whole number percentages only, hence the steps that appear in the plot.

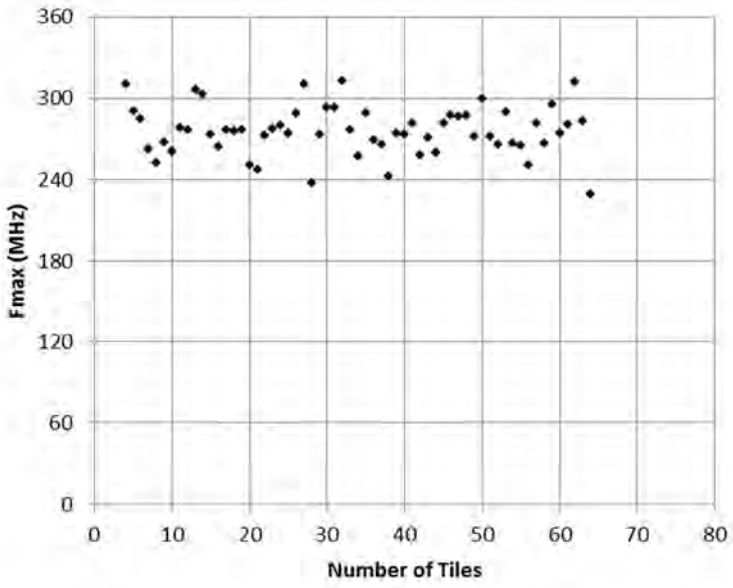

Figure 7.5: Maximum possible clock rate reported by the PAR tool in PlanAhead, vs. number of tiles, for the Binary Counter System.

CONCLUSIONS AND FUTURE WORK

Over the course of this thesis project, a novel approach to fault-tolerance in SRAM-based FPGAs was designed, implemented, and tested. Although based on the time-honored combination of TMR and scrubbing techniques, this approach incorporated two new features. The TMR scheme was modified to include numerous dormant spares (also known as tiles) which could be substituted for any member of the active triad that suffered a fault, thereby extending the lifetime of the system. A multi-pixel radiation sensor was also integrated with the system, providing the scrubber with information about which areas of the FPGA were most likely to be damaged, and thereby improving the system's overall repair rate. Together, the techniques utilized by the system are able to offer robustness against the three major types of radiation damage: SEUs, SEFIs, and TID.

Four demonstration systems were created to illustrate and test this approach. Each system featured a different number of spare tiles, and a different type of computational unit inside each tile. Each system was tested on the Xilinx ML605 board and verified to work as intended. The successful implementation of this variety of systems, created with different requirements and goals, indicates that the presented approach is scalable and adaptable to an assortment of different applications.

Theoretical modeling of the demonstration systems revealed that all four designs would be extremely reliable in representative earth orbits under average conditions. The models also showed that the addition of the two novel features (the numerous spares and the radiation sensor) could greatly increase the robustness of the systems, depending on

the fault rate conditions to which they were subjected. Additional experiments conducted on one of the systems illuminated the relationships between fault rate and MTBF, and between the number of spares included in the system and two different measures of overhead (FPGA resource utilization and maximum achievable clock rate).

Many avenues for further research remain. Future work will include porting the four demonstration systems to custom FPGA boards which can be used for *in situ* testing (e.g. on a high-altitude balloon platform or sounding rocket). The planned boards will include a separate controller FPGA to house the master microprocessor and scrubber, as was originally envisioned for the final design. Future work will also include the design of a system with multi-purpose tiles and additional voters, so that fault tolerance and the full flexibility of a partially reconfigurable system can be realized in the same device. Both of these projects are in progress as of this writing.

An additional problem that needs to be resolved concerns our ability to make the reconfigurable tiles within the FPGA truly independent of one another. Our investigations up to this point indicate that flipping a random configuration bit that pertains to one tile can affect another, because routes that serve one PRR may be sent through another, unrelated PRR. (Reconfiguring a tile with a “fake” does not cause this problem, since the tools preserve all of the PRR-crossing routes in the “fake” tile as well.) Exactly how this behavior might be prevented is an important topic for future study.

REFERENCES CITED

[1] A. Jacobs et al., "Reconfigurable Fault Tolerance: A Framework for Environmentally Adaptive Fault Mitigation in Space," in *International Conference on Field Programmable Logic and Applications*, Prague, 2009, pp. 199-204.

[2] P.S. Ostler et al., "SRAM FPGA Reliability Analysis for Harsh Radiation Environments," *IEEE Trans. Nucl. Sci.*, vol. 56, pp. 3519-3526, December 2009.

[3] T. Buerkle et al., "Ionizing Radiation Detector for Environmental Awareness in FPGA-Based Flight Computers," IEEE Sensors J., to be published.

[4] NASA Radiation Effects Group. "Space Radiation Effects on Microelectronics" [Online]. Available: http://parts.jpl.nasa.gov/docs/Radcrs_Final.pdf

[5] J. Scarpulla and A. Yarbrough, "What Could Go Wrong? The Effects of Ionizing Radiation on Space Electronics," *Crosslink*, vol. 4, no. 2, summer 2003.

[6] A. Lesea and P. Alfke, "Xilinx FPGAs Overcome the Side Effects of Sub-40nm Technology," Xilinx Inc., San Jose, CA, WP256, Oct. 11, 2011.

[7] A. Holmes-Siedle and L. Adams, "Metal-oxide-semiconductor (MOS) devices," in *Handbook of Radiation Effects*, 2nd ed. Oxford: Oxford University Press, 2002, p. 163.

[8] Xilinx Incorporated. *Radiation Effects and Mitigation Overview* [Online]. Available: http://www.xilinx.com/esp/mil_aero/collateral/ presentations/radiation_effects.pdf

[9] Xilinx Incorporated. *Single-Event Upset Mitigation for FPGA Block Memories (XAPP 962, v. 1.1)* [Online]. Available: http:// www.xilinx.com/support/documentation/application_notes/xapp962.pdf

[10] M. Caffrey et al., "On-Orbit Flight Results from the Reconfigurable Cibola Flight Experiment Satellite (CFESat)," in *17th IEEE Symposium on Field Programmable Custom Computing Machines*, Napa, CA, 2009, pp. 3-10.

[11] G. Seagrave. *SpaceCube: A Reconfigurable Processing Platform for Space* [Online]. Available: https://nepp.nasa.gov/mapld_2008/ presentations/i/08%20-%20Godfrey_John_mapld08_pres_1.pdf

[12] H. Kawai, "A Tile-Based Fault-Tolerant Approach for a Long-Life VLSI System," M.S. Thesis, Grad. School of Syst. and Inform. Eng., Univ. of Tsukuba, Tsukuba, Japan, 2007.

[13] D. Kang et al., "Design and Implementation of a Radiation Tolerant On-Board Computer for Science Technology Satellite-3," in *NASA/ESA Conference on Adaptive Hardware and Systems*, Anaheim, CA, 2010, pp. 17-23.

[14] M. Fras et al., "Use of Triple Modular Redundancy (TMR) technology in FPGAs for the reduction of faults due to radiation in the readout of the ATLAS Monitored Drift Tube (MDT) chambers," in *Topical Workshop on Electronics for Particle Physics*, Aachen, Germany, 2010 © IOP Publishing Ltd. and SISSA. doi: 10.1088/1748-0221/5/11/C11009.

[15] B. J. LaMeres and Clint Gauer, "Dynamic Reconfigurable Computing Architecture for Aerospace Applications," in *IEEE Aerospace Conference*, Big Sky, MT, 2009, pp. 1-6.

[16] C. Gauer, "Radiation Tolerant Many-Core Computing System for Aerospace Applications," M.S. Thesis, Dept. of Elect. and Comput. Eng., Montana State Univ., Bozeman, 2010.

[17] F. Kraja and G. Acher, "Using Many-Core Processors to Improve the Performance of Space Computing Platforms," in *IEEE Aerospace Conference*, Big Sky, MT, 2011, pp. 1-17.

[18] K. Singh et al. *An Evaluation of Software Fault Tolerance Techniques on a Tiled Architecture*. [Online]. Available: http://www.east.isi.edu/~mfrench/216_singh_pp.pdf

[19] J. Lach et al., "Low Overhead Fault-Tolerant FPGA Systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, pp. 212-221, June 1998.

[20] K. Zhang et al., "Triple Modular Redundancy with Standby (TMRSB) Supporting Dynamic Resource Reconfiguration," in *IEEE Autotestcon*, Anaheim, CA, 2006, pp. 690-696.

[21] M. Abramovici et al., "Roving STARs: An Integrated Approach to On-Line Testing, Diagnosis, and Fault Tolerance for FPGAs in Adaptive Computing Systems," in *The Third NASA/DoD Workshop on Evolvable Hardware, Proc.*, Long Beach, CA, 2001, pp. 73-92.

[22] M. Garvie and A. Thompson, "Scrubbing away transients and Jiggling around the permanent: Long survival of FPGA systems through evolutionary self-repair," in *10th IEEE International On-Line Testing Symposium, Proc.*, 2004, pp. 155-160.

[23] N. Rollins et al., "A Comparison of Fault-Tolerant Memories in SRAM-Based FPGAs," in *IEEE Aerospace Conference*, Big Sky, MT, 2010, pp. 1-12.

[24] N. Rollins et al., "Software Fault-Tolerant Techniques for Softcore Processors in Commercial SRAM-Based FPGAs," in 24th International Conference on Architecture of Computing Systems, Como, Italy, 2010.

[25] H. Quinn and P. Graham. *Terrestrial-Based Radiation Upsets: A Cautionary Tale.* [Online]. Available: http://www.cosmiacpubs.org/ pubs/08_quinn_neutron.pdf

[26] F. P. Mather, "Reliability Analysis and Modeling of a Dynamic TMR System Using Standby Spares," NASA Jet Propulsion Lab., California Inst. of Tech., Pasadena, CA, Tech. Rep. 32-1467, Nov. 1, 1969

[27] M. G. Gericota et al., "A Framework for Fault-Tolerant Real Time Systems Based on Reconfigurable FPGAs," *IEEE Conference on Emerging Technologies and Factory Automation*, Prague, 2006, pp. 131-138.

[28] J. F. Wakerly, "Microcomputer Reliability Improvement Using Triple-Modular Redundancy," *Proc. Of the IEEE*, vol. 64, pp. 889-895, June 1976.

[29] A. Flynn et al., "Bitstream Relocation with Local Clock Domains for Partially Reconfigurable FPGAs," in *Design, Automation & Test in Europe Conference and Exhibition*, Nice, 2009, pp. 300-303.

[30] E. Gowens, "Two Dimensional Radiation Sensor Development for Use in Space Bound Reconfigurable Computers," M.S. Thesis, Dept. of Elect. and Comput. Eng., Montana State Univ., Bozeman, Montana, 2011.

[31] H. Quinn et al., "Static Proton and Heavy Ion Testing of the Xilinx Virtex-5 Device," *IEEE Radiation Effects Data Workshop*, Honolulu, HI, 2007, pp. 177-184.

[32] M. Berg, *Xilinx Virtex 5 Proton Accelerated Radiation Test.* [Online]. Available: radhome.gsfc.nasa.gov/radhome/papers/ D062209_XCVLX30T.pdf

[33] MathPages. *Markov Modeling for Reliability Part 2: Markov Model Fundamentals* [Online]. Available: http://www.mathpages.com/home/kmath232/part2/part2.htm

[34] J. S. Monson et al., “A Fault Injection Analysis of Linux Operating on an FPGA-Embedded Platform,” *Int. J. of Reconfigurable Computing*, vol. 2012, Article ID 850487, pp. 1-11, Dec. 2011.