NOVEL APPROACH

TO FAULT TOLERANCE IN SPACE COMPUTERS

LEVERAGING THE RISC-V ARCHITECTURE

by

Chris Michel Major

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

of

Doctor of Philosophy

in

Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

May 2023

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES – CONTINUED

Figure Page

NOMENCLATURE

ALU	Arithmetic Logic Unit
$BISER$	Built-In Soft Error Resilience
$BRAM$	Block Random Access Memory
BSS	Block Starting Symbol
$CMOS$	Complementary Metal–Oxide–Semiconductor
CNN	Convolutional Neural Network
$COTS$	Commercial Off-The-Shelf
CPU	Central Processing Unit
ECC	Error Correction Codes
$FPGA$	Field Programmable Gate Array
FR	Full Reconfiguration
$FRAM$	Ferroelectric Random Access Memory
GP	Global Pointer
HDL	Hardware Description Language
IC	Integrated Circuit
$ICAP$	Internal Configuration Access Port
ILA	Integrated Logic Analyzer
ISA	Instruction Set Architecture
LUT	Look-Up Table
$MRAM$	Magnetic Random Access Memory
MMU	Memory Management Unit
$MTTF$	Mean Time To Failure
PC	Program Counter
$PCRAM$	Phase Change Random Access Memory
PR	Partial Reconfiguration
$ReRAM$	Resistive Random Access Memory
$RHBD$	Radiation-Hardened By Design
$RHBP$	Radiation-Hardened By Process
RNN	Recurrent Neural Network
RTL	Real-Time Logic
SEE	Single Event Effect
$SEFI$	Single Event Functional Interrupt
SEM	Soft Error Mitigation
SET	Single Event Transient
SEU	Single Event Upset
SP	Stack Pointer
TID	Total Ionizing Dose
TMR	Triple Modular Redundancy
$4MR$	Four Modular Redundant

ABSTRACT

As the aerospace industry continues to accelerate in growth and mission frequency, the demand for high-performance computers that can withstand radiation environments has become a critical need within the field. Traditional space computing systems rely on specialized and complex means of radiation resistance, but more modern systems seek to implement redundant components to mitigate radiation effects. This dissertation presents a novel approach to radiation-tolerant space computing, based on Montana State University's RadPC program, by developing a resilient architecture leveraging the open-source RISC-V processor. The architecture discussed is designed to withstand radiation environments and engage repairs for damaged sections using commercial, off-the-shelf components - without requiring radiation-hardened fabrication processes or specialized manufacturing. This dissertation discusses the design and performance of the components required to ensure radiation resilience in the system, reconfigure compromised processors in the event of damage, and provides a characterization of the system's overall performance in various space environments.

1

INTRODUCTION

The Need For Space Computers

Space exploration, as an industry, has grown rapidly over the past decade due to increasing interest and investment from the private sector [39]. Companies such as SpaceX, Blue Origin, Firefly Aerospace, etc. have formed to develop new spacecraft [60, 62, 63] and companies such as Boeing, Lockheed Martin, and Northrop Grumman Innovation Systems have accelerated research and development of new aerospace technologies [58, 59, 61]. NASA continues to launch the Artemis program, a series of missions designed to cement a human presence on the moon [42], and has set its sights on future Mars missions [44]. With increased demand for space exploration comes increased demand for aerospace hardware that can safely carry humans and equipment deep into space, and these companies and organizations seek to meet these requirements.

Despite their advancements in Earth-based computing technology over the last few decades, the reliability and performance of currently implemented space computers has stalled. Consider the Mars-deployed Opportunity Rover, which suffered a nearly fatal systems failure in 2009. Charged particles from a radiation strike had caused its motor controller board to crash [65]. The incident left NASA engineers and technicians with no choice but to let the controller stop the rover, wait for the system to resume normal operation, and hope that no further strikes would interfere. Fortunately, Opportunity made a successful recovery and resumed its mission long past the expected finish date.

The risks of radiation strikes have left their mark on NASA, so much so that engineers considered the 2013 Curiosity rover computer failure to be a result of a radiation strike [57]. While Perseverance, deployed on the surface of Mars in 2021, has yet to experience

any crippling radiation damage, it still uses the exact same dual-computer strategy of its sibling rovers Curiosity, Opportunity, and even Spirit. NASA's Jet Propulsion Lab has documented the Perseverance rover's recovery from radiation effects during its mission, citing its predecessors as examples of similar events. One would assume NASA's next rovers would undergo extensive computer redesign to avoid meeting the same fate, but in fact, the exact same computer system has been used on all the Mars missions going back to 2005 (ie: the BAE RAD750 space computing system). Even before such recent experiments, the aerospace industry has documented extensive cases of radiation-induced failures in commercial spacecraft and classified radiation anomalies in various mission environments [3, 13].

To this day, the development of spacecraft relies on the principle of "flight heritage," a concept that determines the effectiveness of spaceflight hardware by evaluating how many successful space exploration missions in which it has played a part. Current space computing platforms have extensive flight heritage, despite their susceptibilities, and have provided NASA a reliable, yet costly, means of launching and completing space exploration missions.

Types of Radiation Faults

Space computing systems are susceptible to cosmic radiation and charged particles outside of the Earth's atmosphere and magnetic field, requiring some form of error mitigation to continue functioning [6]. Two types of failures can occur, classified as total ionizing doses (TID) and single event effects (SEE) [10]. Figure 1.1, demonstrates both types within a cross-sectional view of a silicon device.

TIDs cause material degradation within a computer's silicon through charged particles but SEEs can deposit charges that cause errors in digital signals and data latching. Older computer hardware demonstrated greater susceptibility to TIDs due to larger process nodes ($>$ 65 nm) used to manufacture such chips. As modern digital technology continues to

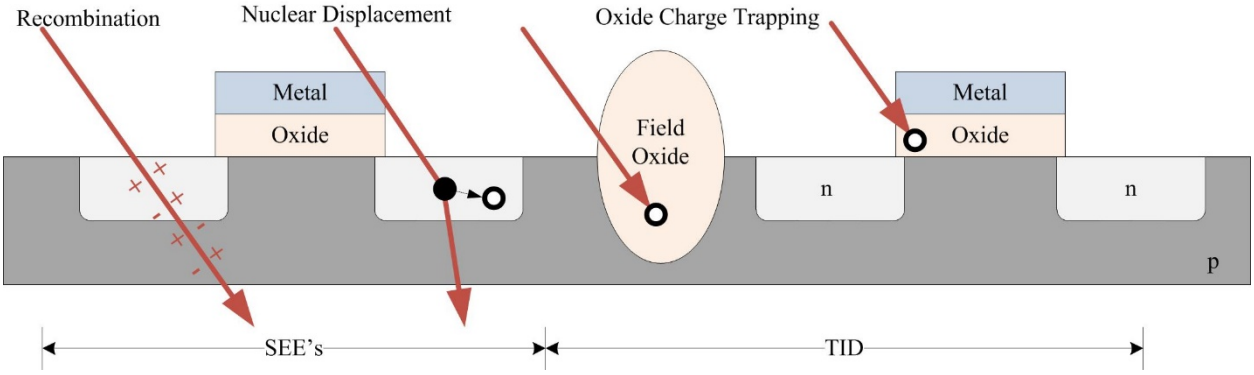

Figure 1.1: Cross-sectional view of SEE and TID effects on a silicon device

decrease rapidly in size, TID errors have become less of a concern due to small feature sizes in fabrication [4]. For instance, complementary metal–oxide–semiconductor (CMOS) integrated circuits (ICs) manufactured in a 65nm process node demonstrate TID tolerance levels just above 500krad, and CMOS ICs in a 28nm process node can achieve up to 600krad [81, 82]. As such, commercial off-the-shelf (COTS) components are considered to be acceptable rad-hard solutions in regards to TIDs [21].

Conversely, SEEs have become the prevalent concern in modern space hardware, a concern that can be divided into three subcategories [23, 64]. A single event transient (SET) occurs when high-energy particles strike a data line within a circuit and cause a logic-level change. A single event upsets (SEU) is similar in concept but only occurs when high-energy particles strike the surface of a CMOS device, depositing enough charge to prompt an unintended logic-level shift. These logic shifts can then interfere with the functionality of a device, requiring counteracting measures such as system resets or power cycles. At this point, a single event functional interrupt (SEFI) has occurred and must be addressed before eventual system failure is reached.

Radiation Hardening Methods

An immediate solution to mitigating radiation effects is to shield the payload with a boundary material. On Earth, this strategy is a simple and cost-effective means of protecting personnel and equipment from radiation hazards. For instance, concrete with a tungsten additive, as a physical boundary between the radiation source and the observer, has been shown to decrease radiation dosage significantly [28]. In space, however, such solutions are immediately infeasible due to the large amounts of mass and volume needed to achieve significant radiation attenuation. Transporting such mass into space drastically increases launch service costs and is infeasible for missions, especially for smaller payloads [68].

To address these concerns, space hardware engineers rely on two types of processes to develop radiation resistance within computers themselves. Chips can be radiation-hardened by process (RHBP), where the materials are altered to reduce the amount of charge deposited when struck by radiation [37]. This process does not guarantee invulnerability, only increased resistance to radiation-induced errors. Due to the specific and non-standard manufacturing processes required to create these chips, coupled with limited market demand in the aerospace sector, fabrication and design of RHBP hardware is incredibly expensive and often cost-prohibitive.

Engineers also create radiation resistance in systems using techniques in the circuit layout itself, a technique known as radiation-hardening by design (RHBD). Creating charge channels and guard rings to reroute extra charge deposits due to radiation strikes can increase a computer's ability to survive faults and prevent system damage [2, 69]. Supporting circuits are designed and implemented to handle charge deposits and current pulses in storage devices, driving them away from the transistor gates to prevent disrupted operation [49, 66]. Figure 1.2 shows two examples of these additional, charge-rerouting features in the silicon.

These techniques are often paired with RHBP for maximum resistance, but un-

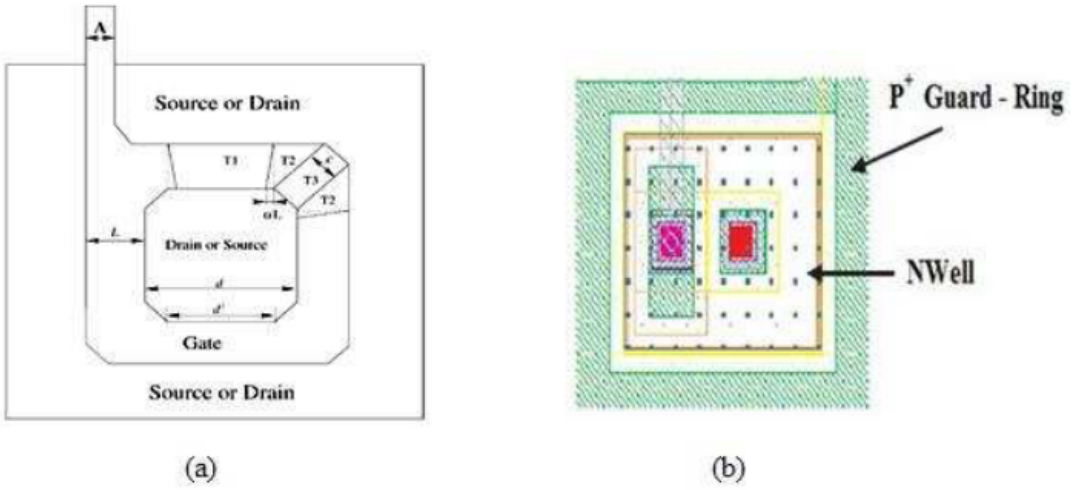

Figure 1.2: Examples of RHBD Transistors: (a) Enclosed Gate Layout, (b) nMOS transistor with charge guard ring

fortunately, cannot guarantee invulnerability. Additional bare-silicon features must be manufactured to take advantage of these hardening strategies, increasing costs, labor, and design complexity. Regardless, as demonstrated with the Curiosity Rover case study, such systems will eventually lead to a failure state given enough time, component wear, and excessive charge deposits.

While circuit hardening techniques (ie: RHBP, RHBD) take the approach of preventing faults, another strategy that has emerged focuses on making systems resilient, or able to recover from faults that have occured. Such techniques are referred to as built-in soft error resilience (BISER) and implement discrete redundancy checking to prevent error propagation [83]. These techniques implement checking mechanisms between redundant latches and flip-flops to ensure data integrity and correct errors as they occur [52]. Memory cell implementations of these resilience solutions seek to provide error mitigation and protect their contents from upsets [49]. These solutions have proven to be just as effective in some environments as RHBP and RHBD techniques but, when implemented on a transistor

level, significantly increase the design complexity and power consumption of the device. Regardless, the premise of resilience through redundancy checking demonstrates greater effectiveness than radiation hardening processes.

Even as RHBP/RHBD devices mitigate the effects of heavy ions, the issue of radiation-induced faults still persists. Within smaller process nodes, 28nm for example, the risk of SEEs compromising the system increases, while TIDs become far less of a concern [34]. This trend is exhibited graphically in Fig. 1.3, demonstrating that as the processing node of an IC decreases in size, hardness against TIDs increases as hardness against SEEs reaches a lower limit. As such, TIDs are considered significantly less of an issue in modern CMOS devices compared to SEEs.

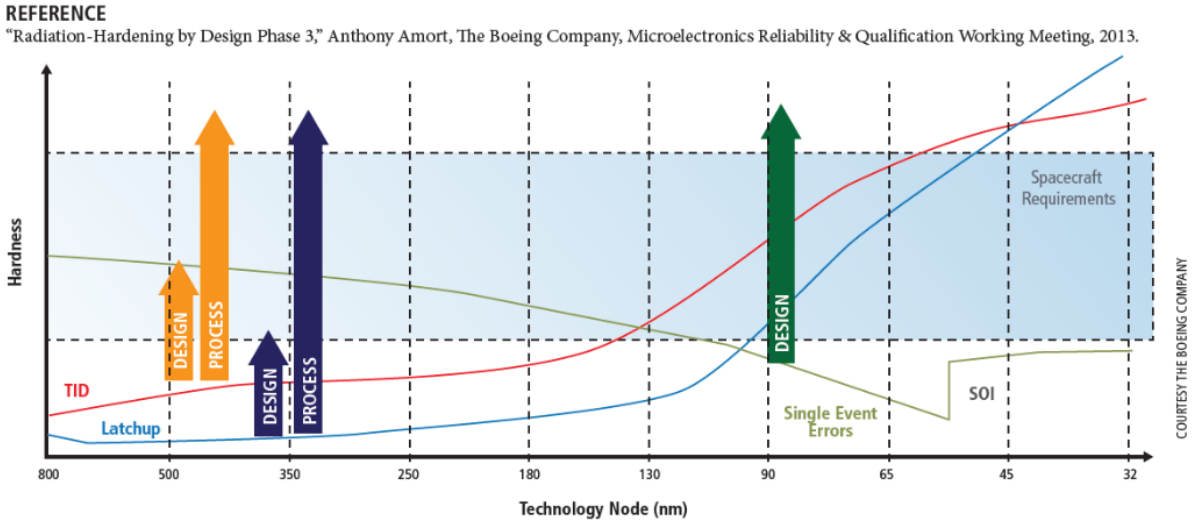

Figure 1.3: Radiation Hardness in relation to CMOS fabrication node size in nm

Thus, the research presented in this dissertation aims to advance the reliability of future space computers by implementing a novel approach to adding resilience to computing systems. Given the limitations of radiation hardening methods in protecting space computing systems, methods revolving around radiation resilience demonstrate greater potential in computational reliability. Methods to reduce the effects of SEEs in commercially-available

components are presented and evaluated in a novel architecture.

MOTIVATION

Space Computing Roadmap

Due to the limitations of radiation-hardened fabrication methods and the heavy role of flight heritage in developing these chips, the hardware speed lags current-generation computers by at least twenty years [27, 43]. Figure 2.1 demonstrates this trend for radiation-hardened processors to lag behind commercially available processors in terms of speed and performance. Coupled with the aerospace industry's preference for systems with flight heritage, the lag in performance is especially noticeable for radiation-hardened systems relative to modern, commercial units.

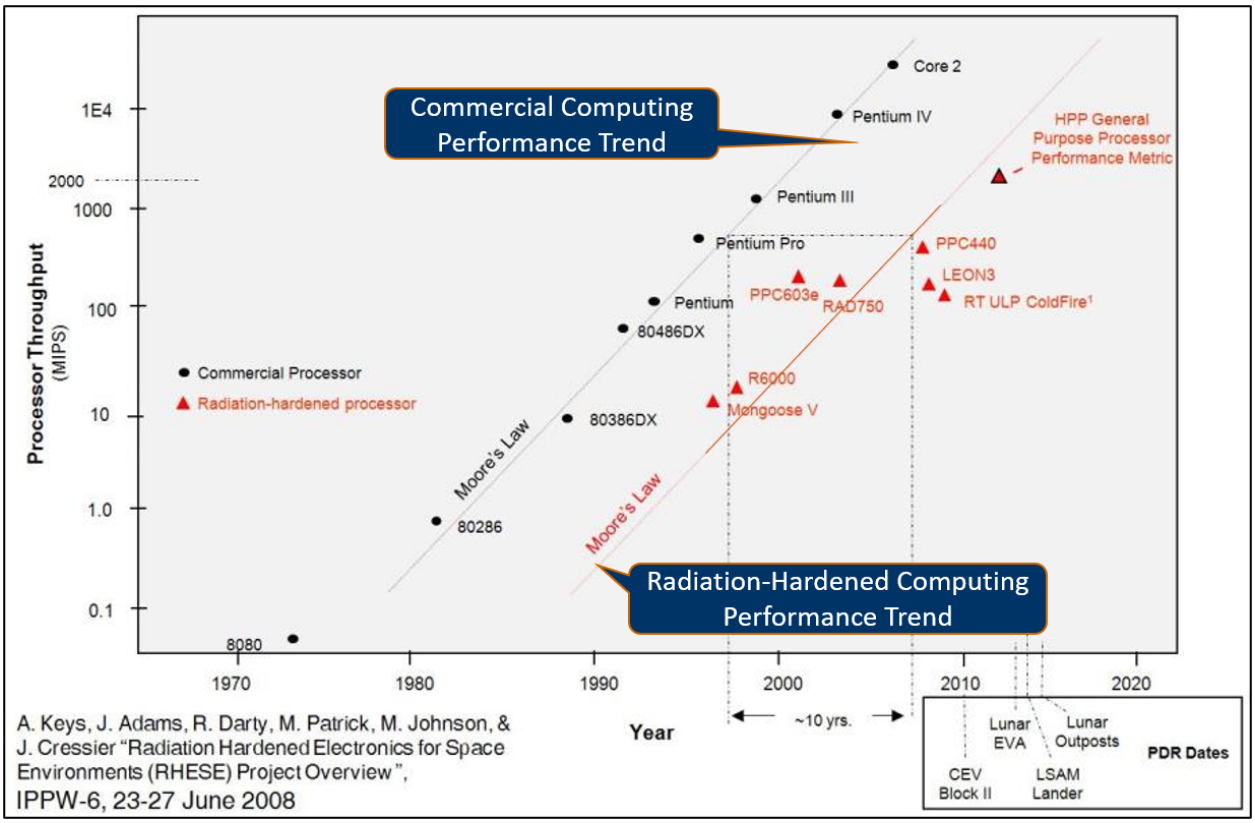

Figure 2.1: Performance of radiation-hardened processors in comparison to commercially-available counterparts

As outlined in the 2020 NASA Technology Taxonomy (TX02), most current, state-of-the-art flight computers are custom, radiation-hardened single processor systems [41]. Generally, such computers demonstrate performances of 35-400 MIPS (Mega-Instructions Per Second), 10-200 MFLOPS (Mega-Floating Point Operations Per Second), consume power within the 20-30W range, and show power efficiency of around 20 MIPS/W. With upcoming missions increasing in number and complexity of tasks, there is a growing need for flight computers to exhibit greater performance.

Within TX02, the need for computers to surpass performances of 1000+ MIPS and MFLOPS, consume less than 10W of power, and achieve power efficiencies of around 20 MIPS/W are present and in high demand. All of these requirements must be met, however, with radiation resilience and the ability to mitigate faults induced by SEEs.

Prior Work On Fault Mitigation In Space Computers

NASA technology roadmaps demonstrate the need for reliable space computing systems that can survive in harsh radiation environments and support high-speed data processing [40]. Increasing space exploration missions demand computing systems that can meet these requirements – with even greater importance given the roles of astronauts on manned missions. A fatal radiation strike on an unmanned mission, though devastating, is often limited to remotely operated hardware. A radiation strike on a mission with a human crew, however, could be the determining factor between survival and disaster. Thus, a technology is required that can provide the resilience and performance future space exploration requires.

Other solutions focus on redundant systems to account for errors, utilizing COTS components in spite of susceptibility to radiation [35, 54]. These redundancies are implemented through multiple processors, memory devices, and interface devices, structured so a fault in one system does not leave others compromised [15]. NASA's choice method is triple modular redundancy (TMR), using three redundant systems to perform a task that is

overseen by a voting mechanism that detects faulty operations[14].

The remaining consideration regards the level of redundancy within the system, presenting a trade-off between reliability and resource consumption. For instance, a TMR strategy cannot practically be applied to every component of a design due to limited device space, timing constraints, resource constraints, usage of third-party IP, and other such considerations. The effectiveness of a TMR strategy must balance the limitations of the device area, component usage, and component density. It is practical, then, to triplicate some components and not others – a strategy known as partial TMR [27] [28]. For example, in [26], the authors present a testing methodology that demonstrates the effectiveness of partial TMR strategies, reducing the cross-sectional area of a logic circuit that can be affected by induced faults. In contrast, the authors in [73] demonstrate the effectiveness of complete TMR across every sub-component of a RISC-V processor in the mitigation of errors induced by faults.

Such technologies, however only delay inevitable failure from accumulated radiation faults. While a system may continue operations for an extended period, the inability to reverse the damage of an induced fault renders a failure state to be inevitable. The authors of [46] further this realization by presenting an experiment demonstrating the effects of micro-SEFIs, a subclass of SEUs that causes interrupts in a digital system by flipping multiple bits in a given area of impact. Such interrupts, given the right area of the digital logic, can compromise the redundancies of a TMR mechanism and render the entire voting mechanism useless. As such, TMR in isolation is an inadequate method of error mitigation.

One COTS technology, however, demonstrates the ability to repair faulted areas. Field programmable gate arrays (FPGAs) are digital logic devices that can be configured into any desired computational architecture. They are a well-established technology with COTS availability, competitive pricing, and extensively-supported design and development environments – as opposed to the limiting supply constraints of conventional RHBP/RHBD

systems. An FPGA can implement a computing system with extra features not available with commercial microcontrollers, including error-checking mechanisms for computational performance. The authors of [71] present a System on Chip FPGA configuration that uses error-checking mechanisms within the processor's CPU to detect errors in space environments and make adjustments in software to correct the functionality of the real-time operating system.

FPGAs can also offer performance increases through parallelism. Tasks that require sequential execution by standard microcontrollers can be sped up through parallel hardware, speeding up operations significantly. This idea of hardware acceleration is especially useful in computationally demanding tasks like machine learning, in which neural networks can leverage faster hardware to reduce computation times [11]. The authors of [9] present an FPGA-based recurrent neural network (RNN) for use in telemetry forecasting in satellites orbiting Earth. To reduce wasted time spent transmitting data and receiving commands, FPGA-based RNNs aim to leverage hardware acceleration amd perform such complex calculations quickly. This strategy, however, highlights the need for fault mitigation mechanisms to ensure such logic circuitry is less susceptible to radiation-induced faults.

It is easy to implement logic redundancy on an FPGA by reusing pre-defined modules of logic circuitry; thus, FPGAs have been used as a means of establishing redundancy for space computing technologies [15, 56]. Most importantly, FPGAs can be reconfigured as needed when any portion of its logic is faulted or affected. This feature, known as Partial Reconfiguration (PR), can be used as a means of self-repair and has been pursued in aerospace research as a means of radiation resilience [45, 75]. In [55], the authors present a convolutional neural network (CNN) implemented on an FPGA that performs semantic-segmentation in radiation environments - combining hardware redundancy with partial reconfiguration strategies to provide in-space fault tolerance. Additionally, the authors of [47] present a fault injection method that demonstrates the potential of multi-cell upsets in an FPGA,

showing that a means of error mitigation beyond simple TMR (such as PR) ensures better recovery in a system. These systems are consistent with the methods presented in [8], where the authors demonstrate how increased repair rates can dramatically reduce the probability of fault-induced failure in a TMR system.

A logic circuit is designed in an FPGA using a hardware description language (HDL) and a dedicated synthesis tool to convert the described circuitry into real-time logic (RTL) that the FPGA can run. Popular HDLs include VHDL and Verilog. The synthesis tool also allows a developer to configure timing, logic placement and routing, and generate output files (such as bitstreams) to be deployed onto an FPGA during regular operations. Some synthesis tools even allow for live debugging of an FPGA's internal logic.

Thus, redundant logic circuitry using FPGAs provides the best possible SEU protection while increasing component accessibility and reducing design complexity. Accompanying this strategy with reconfiguration techniques strengthens the ability to flush out faults before they cause extensive damage to a system.

Prior Work On Memory Protection Strategies

It should be note that radiation-hardened memory technologies such as ferroelectric random access memory (FRAM), magnetic random-access memory (MRAM), phase change random access memory (PCRAM), and resistive random access memory (ReRAM) do demonstrate potential for computing systems that require increased resistance to SEEs and SEFIs [7, 29, 51]. These technologies have also been developed as a means of enabling further resilience within FPGA devices [1, 50]. Such memory devices, however, are expensive and have limited commercial availability for larger storage. Though the manufacturing technology will continue to develop and reduce the costs for development, the effects of SEUs within the CMOS technology that links memory to the rest of the logic still prove vulnerability and require resilience mechanisms to ensure successful operation.

Preventative methods for memory contents include error correction codes (ECCs), a method involving tagging data with extra bits to cross-check with the corresponding original value and detect bit flips. This allows a device reading from memory to determine if its contents have been corrupted. If a single bit has flipped, the ECCs may allow for data correction. If the error cannot be corrected, the ECCs will indicate to the rest of the system that further action may be required.

Resolving errors within memory devices can be conducted by iterating through each data cell and checking its value against a reliable source. This method is known as memory scrubbing and is conducted by a scrubber device. If an error is detected, the data can be refreshed with the correct value through a rewrite. The behavior of the scrubber can further be tailored to the device's constraints by scrubbing in selected regions – a strategy known as partial scrubbing [18, 84]. Applying scrubbing to device memory devices and an FPGA's configuration memory can further increase the reliability of the system by ensuring data integrity.

Our Prior Work on Resilient Space Computing

In ongoing efforts to meet these needs, Montana State University has been developing a computer architecture designed to mitigate the effects of space radiation. This technology, named RadPC, uses FPGA hardware to run and monitor a single program in parallel on four processors [36]. Redundant components are an established means of radiation resilience within the aerospace industry and thus form the underlying structure of the RadPC architecture [17, 32]. As a means of protecting the redundant processors and resolving soft errors within their operations, dynamic reconfiguration of the FPGA is used [16, 30]. Figure 2.2, below, describes this redundancy configuration at a higher level.

Similar to NASA's TMR design, the four-modular redundancy (4MR) of these processors ensures an operation can continue in the event of an SEU interruption. This

partially-redundant design is coupled with PR, a technique that allows FPGA hardware to be refreshed and clear out any errors that occur within its logic-circuit fabric [25]. Thus, if a processor is damaged, it can be partially reconfigured by the FPGA while the remaining TMR-like processor structure continues program execution. Parallelization within FPGA hardware is advantageous and has been used extensively to develop faster processing systems, efficiently delegating resources to increase performance [48]. While each parallel core within RadPC performs the exact same task, the delegated resources ensure redundancy amidst faults – an SEU that cripples one processor core will not affect the others from completing the current instruction.

It should be noted that the entirety of a processor does not undergo PR in RadPC. The central processing unit (CPU) and memory management unit (MMU) can be reconfigured, but the memory devices - local or global - cannot undergo PR and are instead treated by scrubbers. Thus, the part of a processor that can be repaired via PR is called a tile. RadPC has four processor tiles that can reconfigured and repaired, while the corresponding memory devices are left alone.

Processor health is monitored by additional circuitry within the FPGA. A voter circuit watches each processor's outputs for each software cycle to determine if a fault has occurred, then schedules a PR to repair the processor and resume operations. This is a form of software checkpointing and is used to account for possible drift in processor timing. Data memory is protected from SEUs through use of a memory scrubber, another mitigation circuit triggered by the voter in the case of a detected processor fault. This scrubber will read the data memory cells that are assigned to each processor, compare their contents, and rewrite faulty values to match the majority [5]. These elements are overseen by a soft error mitigation (SEM) circuit that monitors the underlying configuration memory of the FPGA and corrects SEUs that threaten the system as a whole. The SEM is a Xilinx FPGA IP that serves as a scrubber specifically designed for configuration memory [79].

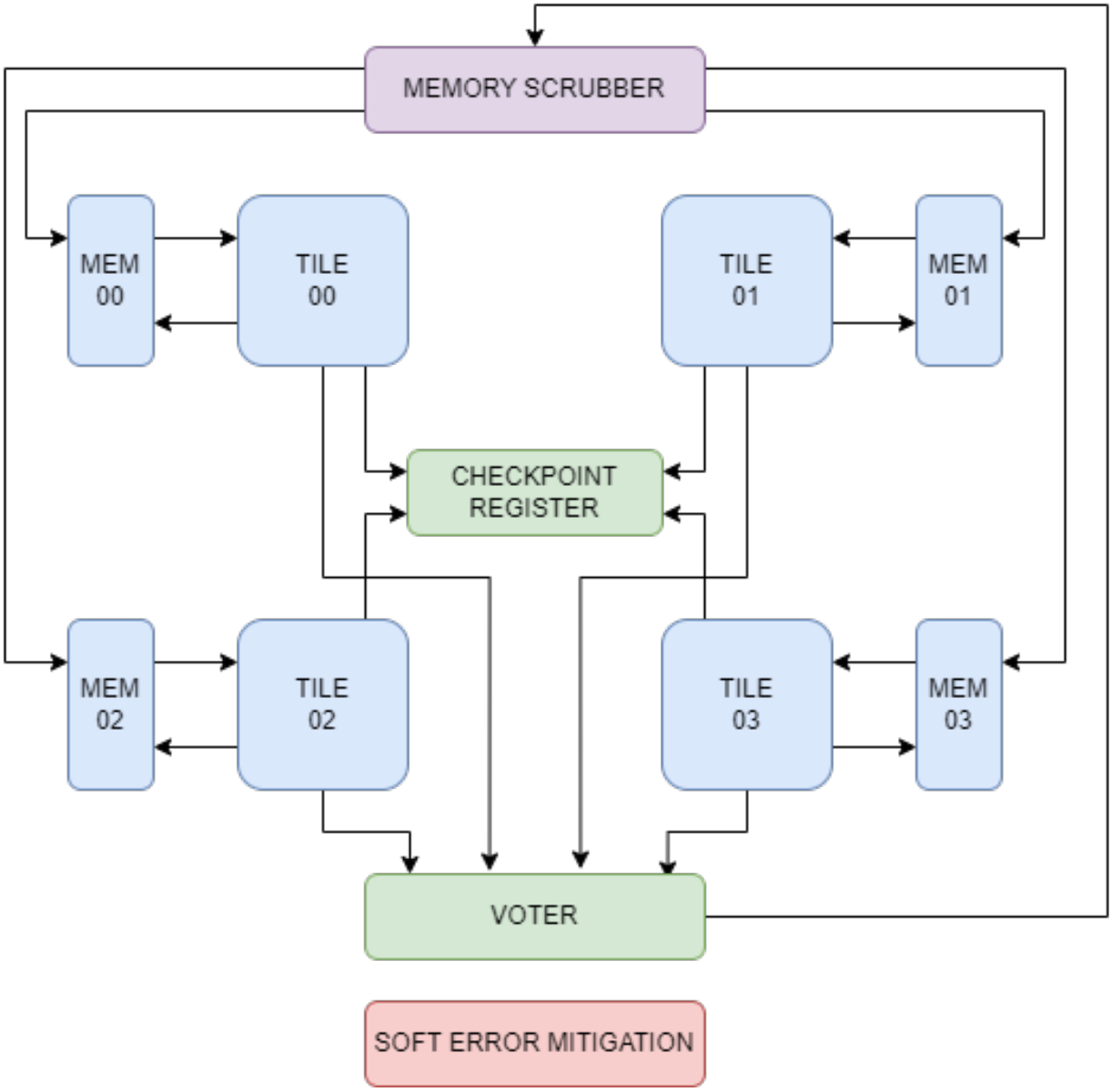

Figure 2.2: Block diagram of the existing RadPC 4MR architecture

It is within this fault recover system that RadPC's inefficiencies in performance emerge. While RadPC's default clocking speed of 16 MHz is comparable to that of a standard COTS microcontroller, its program execution in the context of SEU mitigation is inefficient. The time required to successfully perform a PR forces runtime lag between the damaged processor and its counterparts, which will affect the voter as it waits for each processor to present its program results for the same instruction cycle. As SEUs affect the system and are repaired, this lag reaches a plateau and virtually guarantees a significant time delay between each processor's instruction cycles. That delay threatens the runtime performance of the RadPC architecture, creating not only inefficiency in program execution but opening large windows of time for SEU strikes to further interrupt operations.

RadPC, however uses a proprietary softcore processor provided by Xilinx known as the MicroBlaze [76]. While this component has sufficed as proof-of-concept thus far, its blackbox design prohibits access to its internal components for modification or upgrade. Synchronization of processors must be done with software, leaving multiple instructions' worth of clock cycles primed for further errors that can disrupt performance. Tighter control of system timing necessitates tighter control of system components; thus, an open-source processor such as the RISC-V would prove more beneficial.

Regardless, RadPC has established and continues to establish flight heritage, including missions on high-altitude balloons, sounding rockets, two CubeSats, two International Space Station missions, and an upcoming demonstration on the surface of the moon [24, 31, 33, 42, 53, 72]. Figure 2.3 shows a timeline of the various missions in which RadPC has been tested as a flight payload. As each mission has demonstrated and stress-tested another layer of functionality, however, it has become evident that current SEU mitigation strategies compromise performance. Considering the increased need for high-performance space computing systems, an architecture that can withstand space radiation but lacks speed is inadequate to the needs of the industry.

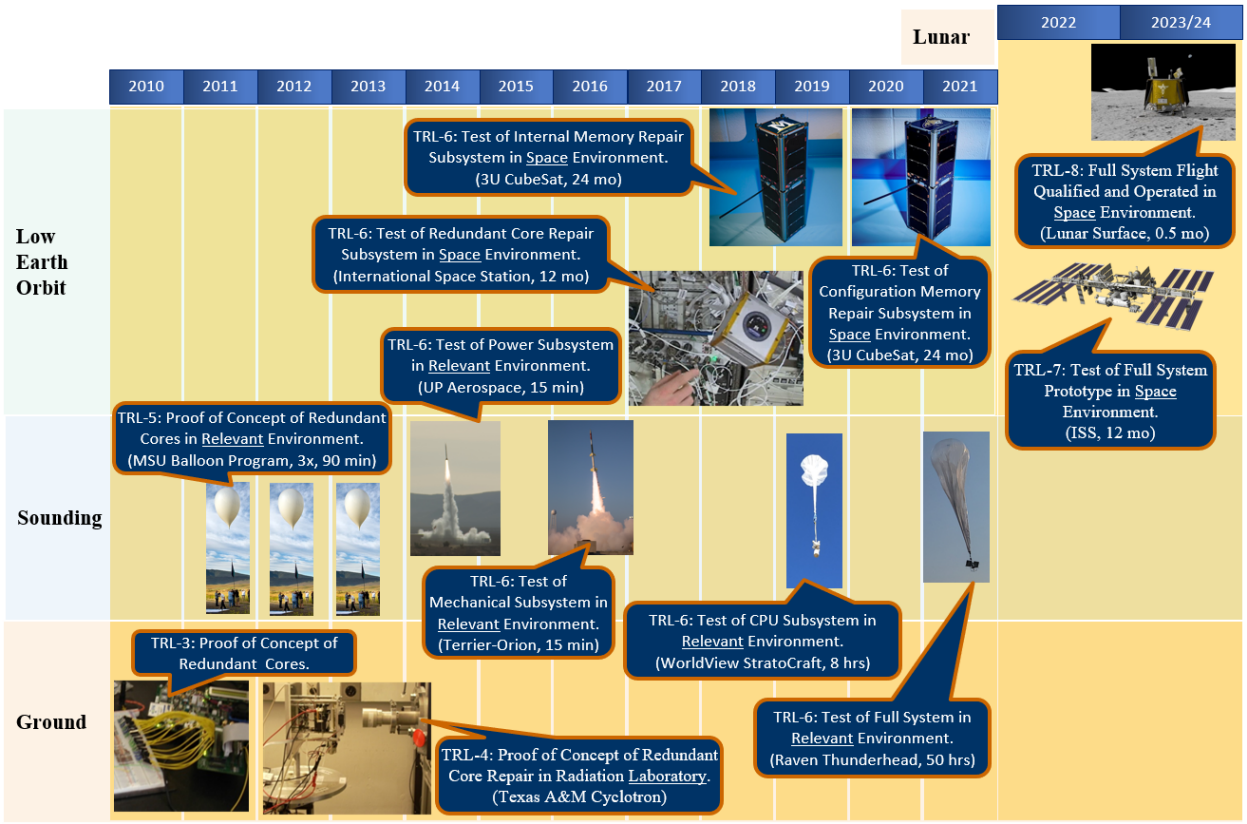

Figure 2.3: Timeline of RadPC missions and demonstrations

OUR APPROACH

Limitations of Our Prior Work

The most critical change to RadPC is the new softcore processor. Softcore processors can be divided into two categories: blackbox and glassbox. The Xilinx Microblaze qualifies as a blackbox processor in that all of its internal components are inaccessible to a developer or user - the input and output ports are all that can be accessed and select settings, such as the memory map, can be modified with limitations. As such, if special alterations to the design are required for the system to function, no changes can be made. This design choice is understandable for a corporation wanting to protect its intellectual property and provides easy access to a developer, but limits its use in systems development significantly.

Conversely, a glassbox processor provides complete access to its internal components for analysis and modification. This impedes its ability to be sold or licensed by a company and can complicate its implementation in a system, but can provide developers enough access to modify the processor to suit the system's needs. While developing a glassbox processor can be time-consuming and difficult, using an established ISA allows the developer to leverage existing technologies with extensive support.

The new RadPC architecture applies the 4MR redundancy, voting mechanisms, and scrubbing strategies of the existing RadPC architecture, but changes several key components and adds others for further resilience. First, the MicroBlaze softcore processors are replaced with custom-built RISC-V softcore processors. Second, the voting mechanism is extended to the registers, program counter, and peripheral devices. Third, the memory scrubber is extended to both data memory and instruction memory. The SEM continues to scrub the FPGA configuration memory and PR is still used to recover processor tiles. Figure 3.1 shows a block diagram of the new RadPC architecture, where each block represents a component designed in VHDL and implemented onto an FPGA.

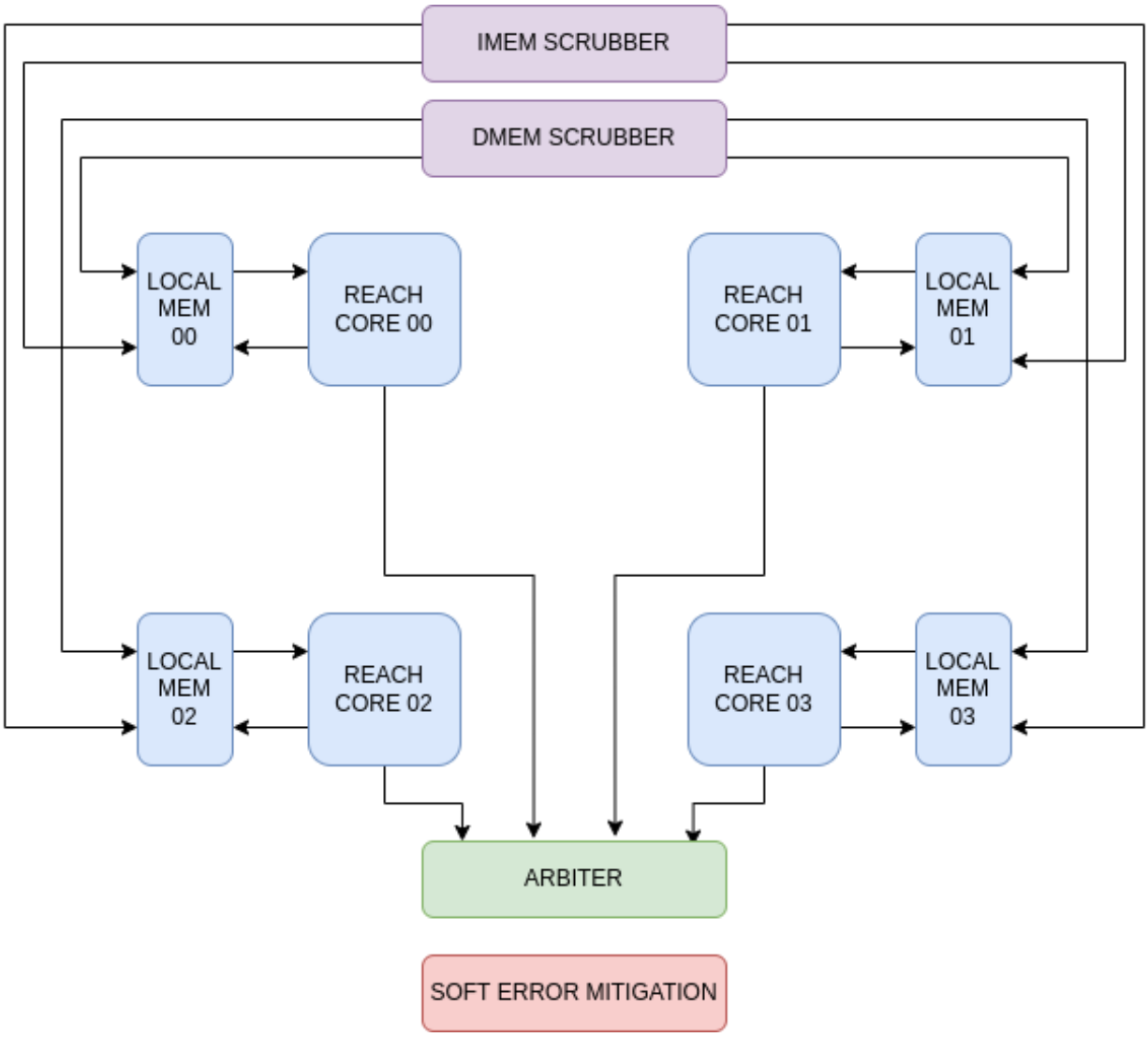

Figure 3.1: New RadPC Architecture Components

Softcore Processors and the RISC-V Architecture

It has been shown that COTS microcontrollers are greatly susceptible to SEUs, but lack the features FPGAs can provide to mitigate faults. The logical solution would be a combination of the two devices, leveraging the flexibility of microcontrollers with the mitigation techniques of FPGAs. Such devices are known as softcore processors and are widely adopted by FPGA designers for use in digital systems [19]. While proprietary processors exist and are marketed by FPGA development companies such as AMD-Xilinx or Intel-Altera, many open-source processors are available for use and exhibit extensive documentation and support. Open-source processors boast the advantages of extensive documentation, community support, lack of licensing fees, and ease-of-access for both academic research projects and industry use.

The RISC-V processor architecture has gained popularity over the last few years, having developed an accessible ecosystem for implementation and use in digital systems [22]. The open-source Instruction Set Architecture (ISA) supports several different types of processor cores, allowing 32-bit, 64-bit, and even 128-bit configurations. Multiple versions of complete instruction sets are offered, ranging from basic integer operations to floating-point calculations and more. The most basic functional RISC-V variant is the RV32I, with 40 instructions necessary to run any basic C program.

The RISC-V ISA has been adopted into aerospace research and technologies, with the authors of [73] and [74] implementing RISC-V softcore processors into their FPGA designs. Specifically, [74] presents an experimental setup using RISC-V processors in an FPGA to test the effects of fault injection, leveraging PR as a means of flushing out bit flips in the system. Likewise, the authors of [12] use softcore RISC-V processors in a redundant configuration with ECCs and register-level monitoring to maintain radiation tolerance and continue operation while faults are induced.

Central Processing Unit (CPU)

Thus, the processor tiles in RadPC have been replaced with a custom-designed RISC-V glassbox processor, named the "Reach" core. The RISC-V RV32I configuration was chosen as it is the simplest instruction set/feature set provided by the ISA, with forty instructions for basic integer arithmetic. The "32" indicates that all values in operation are 32-bit, and the "I" indicates integer-only operations. Where many processors perform instruction fetching, decoding, and execution in sequential order, the RISC-V performs these steps in parallel via combinational logic.

The Program Counter (PC) module creates a program counter signal that increments with each clock cycle or skips to a branched instruction address. This value is used to address the Instruction Memory (IMEM) and fetch the current instruction. The opcode is pulled from the instruction and read by the Control Unit, indicating which instruction, which arithmetic operation, and which data paths are to be executed. The instruction also includes the necessary operands, which are used to fetch the desired register values from the registers or to generate an appropriately-signed immediate value. These components are fed into the Arithmetic Logic Unit (ALU) and branch evaluator, which determine the result of the instruction and the address of the next instruction. Finally, any instructions that require storage in a memory device are read to Data Memory (DMEM) or a peripheral device.

Figure 3.2 shows how all of these elements work together to form a complete Reach processor. It should be noted that there are external ports that go to the Arbiter and scrubber repair components, which will be discussed later in this dissertation.

Given these basic processor requirements, slight modifications are made to ensure ISA compliance while allowing for RadPC correction systems. Unlike most RISC-V softcore processors, read/write access is granted to the registers and program counter by both the CPU control unit and the error correction mechanisms. The memory map also provides backdoor access to the memory scrubber device, allowing erroneous values to be corrected

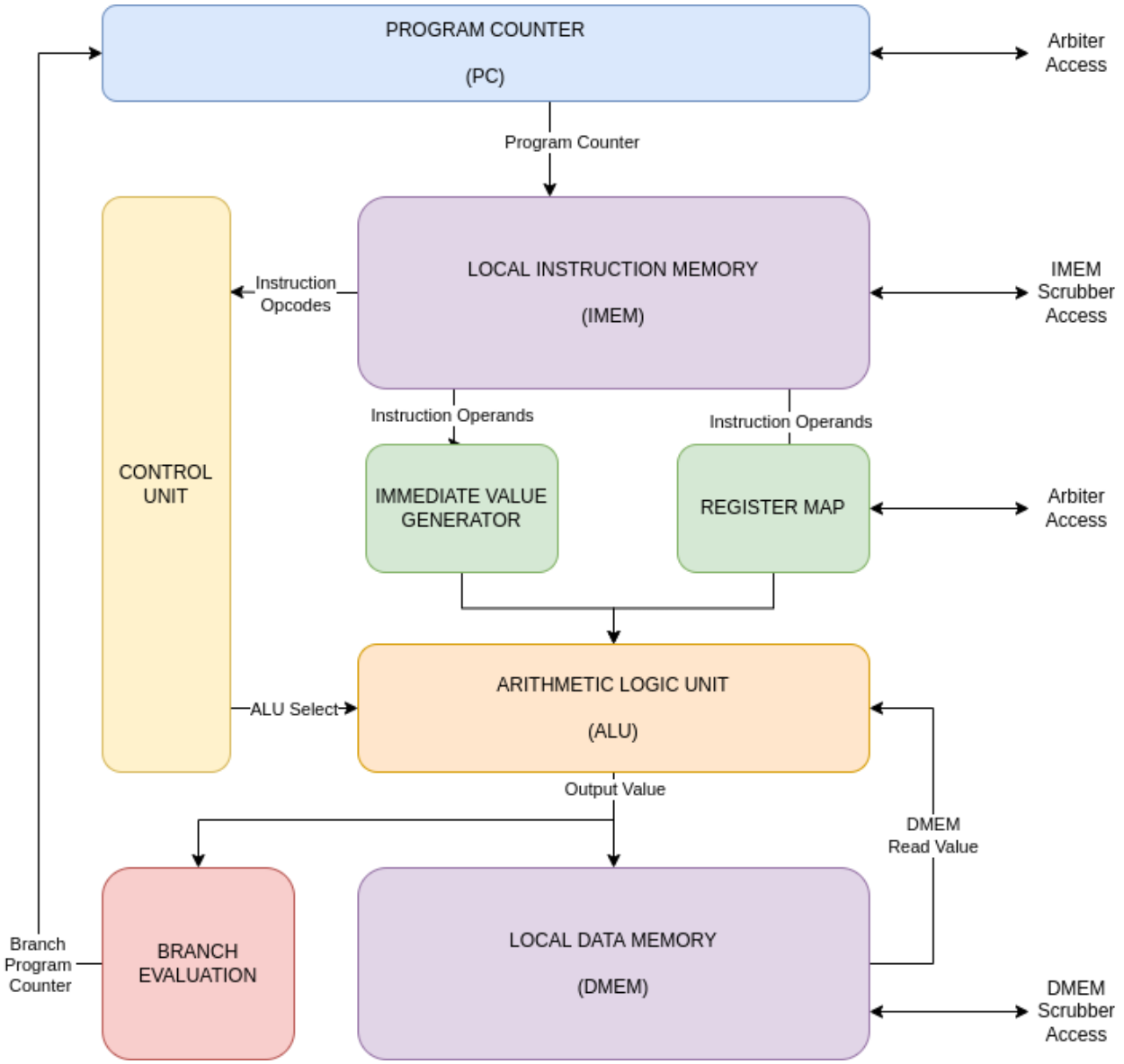

Figure 3.2: Reach core CPU components. Note that both memory devices are local to the CPU and only grant port access to the error mitigation systems.

as CPU access is momentarily restricted. The SEM does oversee the configuration memory for each processor tile but is not controlled whatsoever by the processor.

Memory Management Unit (MMU)

The memory devices interfaced by the CPU, including the IMEM, DMEM, and peripherals, are handled by a Memory Management Unit (MMU) component. This component routes the memory read/write data, address, and read/write enable signals to the memory devices, referencing a hardcoded memory map to ensure that the correct data is accessed by the device. The IMEM and DMEM components are considered local memory; each Reach core has its own memory copy that cannot be read or written by any other Reach core save itself. They are also considered to be dual-port devices, as the scrubber can access addresses via a secondary port to correct errors.

The memory map allows for addresses extending from 0x00000000 to 0xFFFFFFFF in hexadecimal, but the sizes of the memory devices are limited to the FPGA device. As such, all start addresses and sizes for components in the memory map are defined in a VHDL package file and constrained to fit the FPGA selected. The IMEM and DMEM components are required for basic operation, with each set to 4096 bytes (4 kilobytes) of memory. Previous projects developing the Reach core have enabled the memory map to use GPIO, UART, and SPI components, which qualify as memory components in that the registers controlling each of these peripherals are treated as address spaces.

With the exception of defining peripherals, FPGA memory devices are best implemented by using a device's block random access memory (BRAM) modules. Interfacing VHDL code with these devices usually requires a template VHDL component, provided by the FPGA vendor, that ensures the synthesis tools understand the need to use BRAM. Such templates often allow for single-port or dual-port RAM devices, meaning the number of ports accessing memory are inherently limited. Designing a tri-port memory, for example, would

cause the synthesis tool to consume LUTs and inefficiently use BRAM to meet the design requirements.

The RISC-V memory model reuqires byte-accessible memory and supports read/write operations of bytes, halves (16-bit data), and words (32-bit data). As such, each Reach core memory device (IMEM and DMEM) consists of 4 BRAM components whose contents are pieced together to satisfy byte, half, and word operations. For example, if an instruction requires a full word be read, the memory device would fetch a byte from each BRAM component and assemble them into a full 4-byte word for the CPU. These operations are conducted by the MMU.

Figure 3.3 shows the memory map for the RadPC architecture tested and developed in this dissertation. The IMEM and DMEM devices are considered local to the processor, as are the memory-mapped registers of the peripheral devices. The peripheral input/output lines, however, are considered to be global.

Each Reach core CPU and MMU are considered a part of a reconfigurable processor tile in RadPC. IMEM and DMEM are not considered as such as their errors are better handled by the Scrubber.

Software Workflow

Software is loaded into the Reach core's IMEM memory through the use of a dedicated workflow that compiles C code into machine code. Any program that is run by RadPC is written in C and compiled using a free GCC toolchain provided by the RISC-V collaboration. In general, a collection of C and H files can be written as any standard microcontroller project, then are compiled into an output file to be loaded into instruction memory. A startup program, written in Assembly and called the C Runtime Startup (crt0) is used to set up the global pointer (GP), stack pointer (GP), and clear the block starting symbol (BSS) region. These instructions are all loaded into IMEM VHDL via a Python script that

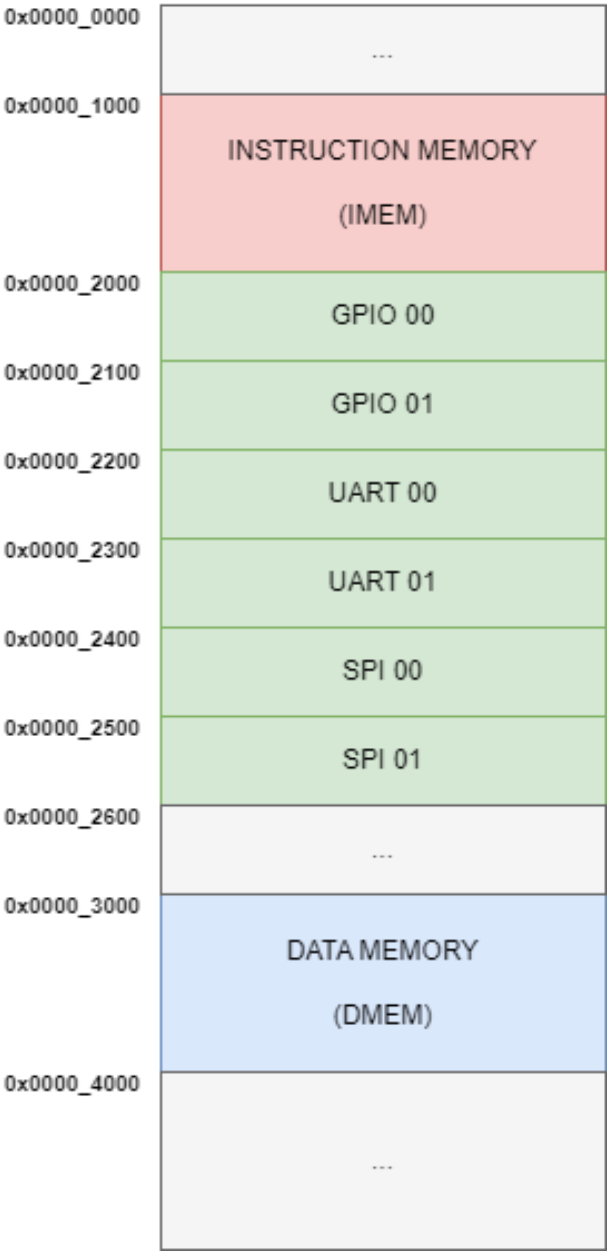

Figure 3.3: RadPC Memory Map

reads the output file, and can be accessed when the VHDL components are synthesized and implemented onto the FPGA. Additionally, the disassembly is exported to a file for debugging and review purposes.

It should be noted that the toolchain supports Clang as a more portable means of compilation. Additionally, the bare-metal standalone nature of the RISC-V processor does support other programming languages and compilers that can create their own standalone libraries, such as Rust. While tests have been performed with these tools for parallel applications, this disseration will focus on the compilation of a C program using the RISC-V GCC toolchain.

Figure 3.4 demonstrates the process of compiling a C project into VHDL code for the Reach core.

The Voter Sub-Component

Traditional TMR systems use a voter to route the correct majority value from three components to the output. RadPC uses a similar voting mechanism on four values, routing the correct value while also reporting any compromised inputs. Thus, the voter is treated as a sub-component in the error correction systems and is a combinational logic circuit to limit the time it takes to evaluate component outputs.

A majority value can, in most cases, be determined from four signals. There are five cases of signal votes that can be seen in a 4MR system, with varying likelihoods of occurrence and repair success. The first case is standard operation, in which all four values are in complete agreement and so the majority value matches all four inputs. No repairs are required to recover the system. Table 3.1 describes all possible combinations of voter inputs expected by the system.

The second case is a single error, where any three values are in agreement but the fourth value disagrees. A majority value can be derived from the three values in agreement, but

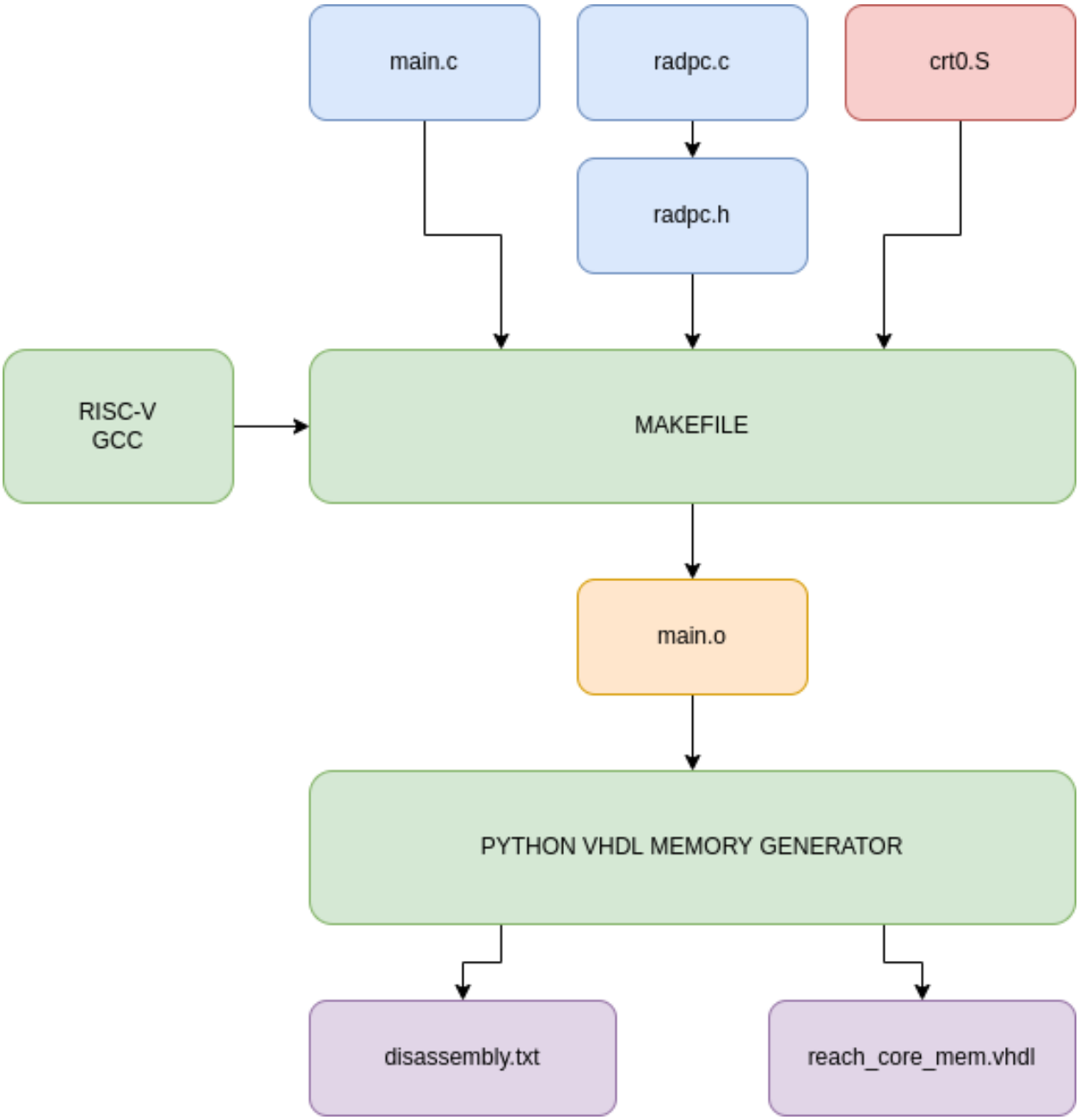

Figure 3.4: Software Compilation Workflow for RadPC

repairs will have to be conducted on the system that output the fourth value. This is the most likely error to occur as an SEE affecting a tile - even affecting multiple bits within a tile - will most likely be confined to a single tile region in the FPGA.

The third case occurs when only two tiles agree with each other. The two tiles in disagreement output values contrary to the first tiles but also to each other. In this case, a majority value is still clear as the correct value must come from the two agreeing tiles, and so a repair must be conducted. The fourth case is a variation of this scenario, but resulting in a failure to find a majority value. The first two tiles still agree with each other, but the other two tiles agree with each other and disagree with the first pair. Thus, a stalemate is reached and a majority value cannot be found. A repair is necessary to restore functionality, though any attempt at a majority value cannot be trusted. This is a rare scenario in that the random nature of SEEs must be enough to flip enough bits such that two tiles fail identically, which is incredibly unlikely during regular operation.

The final case results when none of the tiles agree with each other. In this case, a majority output is impossible and repair is required. The likelihood of this scenario occurring naturally in a space environment is incredibly low and would require a full reset and reconfiguration of the FPGA to return to standard operation.

The Arbiter

Peripheral devices on RadPC, such as general purpose input output (GPIO) ports or serial communication devices, can use a single voter to regulate outputs. Data that is fed back into each of RadPC's Reach cores, such as values stored in the registers, requires not only voting to determine the correct majority value but also a means of correction to prevent bad data from propagating.

Consider the example of a basic counting program. In a standard microcontroller, a value from memory would be stored to a register, sent into the ALU to be incremented by

Table 3.1: Possible Voter Value Combinations

Correct Tiles That Agree	**Incorrect Tiles That Agree**	**Incorrect Tiles That Disagree**	**Majority?**	**Repair Required?**
4	0	0	Yes	No
3	0	1	Yes	Yes
2	0	2	Yes	Yes
2	2	0	No	Yes
0	0	4	No	Yes

a value in another register, and then stored in a third register. The final value would be sent back to memory, to be accessed at a future time. If any of these three registers were to experience an SEE, the program's execution would be altered at any future time the system wishes to access the initial value.

Thus, each Reach core on the RadPC system gives backdoor access to each register to a component capable of voting on correct register values and overwriting bad data. This component, designated the Arbiter, checks each of the 32 Reach core registers, determines which values are erroneous, and corrects the faulty tiles within a single clock cycle of operation. Voting is done by instantiating multiple voter component within the Arbiter that each evaluate a single register across all four Reach cores. When a voter detects an incorrect value through its majority selection, it raises a flag that signals the Arbiter to enable a write to the Reach cores and refresh its register values with the majority.

The Reach core's PC is technically a register, albeit one that cannot be directly accessed as the other register components. As this "register" is essentially to synchronizing each Reach core, it is refreshed along with all other registers whenever the Arbiter demands a write. This ensures that, for any given instruction, the Arbiter can guarantee that all voted register values

correspond to the same instruction and do not reflect different states of program execution. It also "freezes" the instruction momentarily, ensuring the registers cannot be affected by a processor operation but only by an Arbiter write.

Figure 3.5 demonstrates the process the Arbiter undergoes during regular operation.

The Memory Scrubber

Assigning a voter component to each register in a Reach core is not trivial, in terms of FPGA footprint, but is feasible in terms of successful error mitigation and resolution. Thus, connecting a voter component to every single cell in a memory device is ludicrous and inefficient. An FPGA designer could certainly connect every single value in memory to a voter circuit for evaluation, but FPGA resource usage for memory devices can consume large amounts of lookup tables (LUTs) if not properly constrained.

Given the byte-wise nature of the Reach core MMU, the memory scrubber from the previous RadPC architecture is not suitable for this new memory map. As the Reach cores provide access to both IMEM and DMEM devices, each with bytewise access, a memory scrubber is required that can read all of the cells quickly and effectively. It must also not conflict with current write operations coming from the CPU, lest a contention in rewriting an erroneous cell further compromise a Reach core.

The scrubber for the presented RadPC architecture achieves these ends by running perpetually in the background, beginning at the start address of a memory device and iterating through each value until the maximum address. This process repeats as long as RadPC is out of reset. The scrubber iterates through each memory device in lockstep, accessing the same address in each Reach core memory simultaneously. The underlying assumption with this operation is that there is no skew in the memory read/write process and that the memory is accessed at the base clock rate of the system.

As memory devices in the RISC-V RV32I ISA are byte-wise addressable, the voter

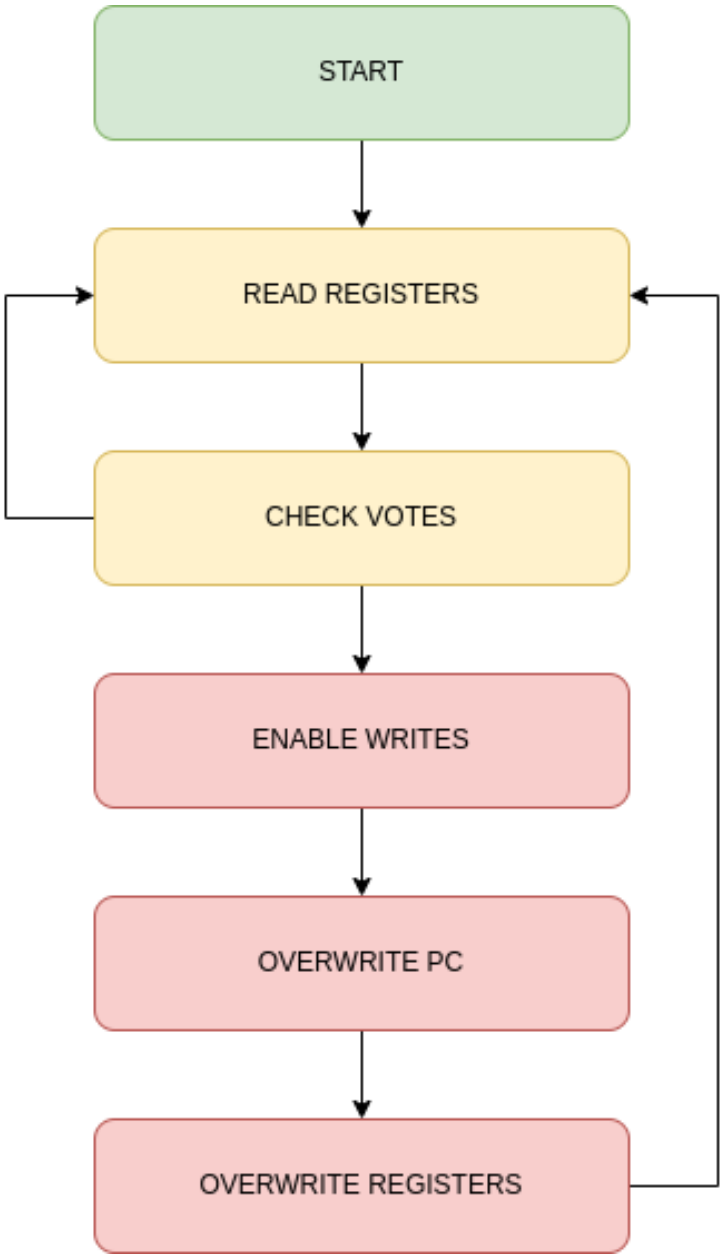

Figure 3.5: Arbiter error mitigation and correction flow

reads full 32-bit words at a time. A voter component that reads each of these memory words controls whether or not an address is to be rewritten by using the syndrome value to control the read/write enable. This ensures that each memory device is rewritten with the correct majority value when errors are detected. For CPU read cycles of the memory device, contention is not a concern. The odds of a write contention are remarkably low but the scrubber will pause if it detects that a write at the address being checked is occurring from the CPU end. Figure 3.6 demonstrates the process the scrubber undergoes during regular operation.

It is possible for errors in a memory device to be accessed by the CPU before the scrubber can access the values and rewrite them. In this case, the Arbiter will correct any errors that propagate from memory and into the registers. It is just as possible for an error in a memory device to exist in a memory address that the CPU never accesses. This will result in the scrubber quietly rewriting that address without the CPU ever interacting with an erroneous value.

It is worth noting that the selective 4MR nature of the system means that the voter components, Arbiter, and scrubber are not duplicated throughout the system. This is to meet the balance between device resource usage, power consumption, resilience, and design complexity.

Soft Error Mitigation

An FPGA's RTL is determined by its configuration memory, which holds the settings for the FPGA required to implement a design. Most FPGAs have ECCs within each configuration memory location that can allow for detection and correction of single-bit and double-bit adjacent faults. These codes are a series of flag bits that can be checked against the data in the memory location and help determine if any bits have flipped. A component external to or within the FPGA can read these codes and, in the event of an upset, correct

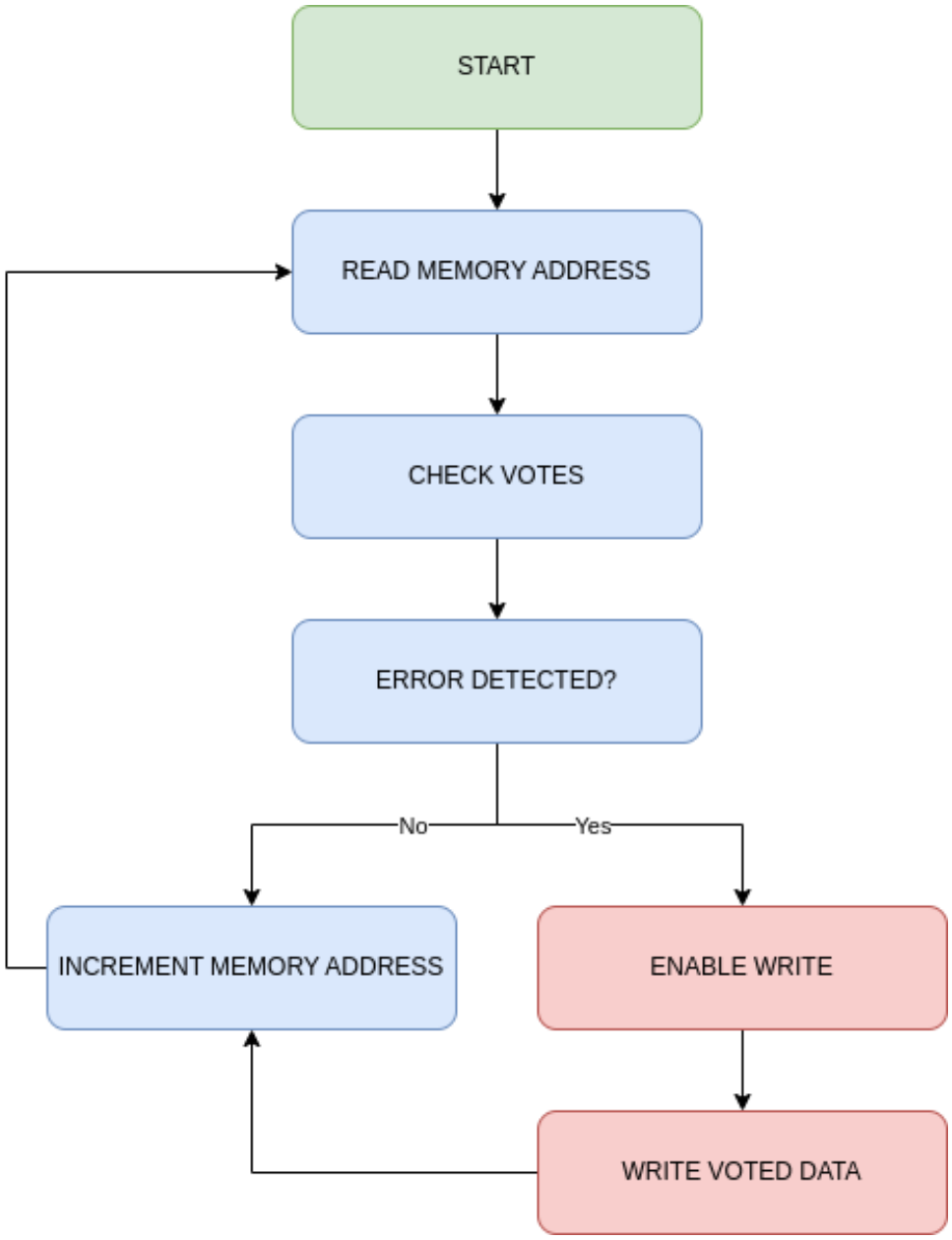

Figure 3.6: Scrubber error mitigation and correction flow

the flipped bit. Perpetual monitoring of configuration memory is required to prevent multiple bits from accumulating over time, as large numbers of upset configuration memory locations will inevitably compromise the design.

Previous RadPC missions have implemented a Soft Error Mitigation (SEM) controller that reads the configuration memory of the FPGA and continually scans each address in search for faults. This component is the only proprietary IP in the RadPC design and is provided by the FPGA development suite. As the SEM checks each location in memory, it checks for discrepancies between the memory contents and the ECC codes

The SEM remains active in the presented RadPC architecture. While it still oversees the FPGA configuration memory and correcting single-bit and double-bit adjacent errors, it does not actively interface with any other correction components. Each Reach core, however, can read the SEM's status data and syndrome reports through the MMU. While this plays no role in error mitigation or resolution, this simple mechanism allows a potential developer to execute certain operations with an awareness of the current radiation environment.

Partial and Full Reconfiguration

An FPGA can have its logic fully reset to a known state using full reconfiguration (FR). As the RTL is set by the configuration memory of the device, which is inherently volatile, resetting power is enough to clear the device and reconfiguration is enough to bring the device's logic to a fully reset and refreshed state of operation. For instance, if an FPGA device were to suffer enough SEEs to upset its configuration memory and compromise performance, a full reconfiguration of the device would be enough to wipe away all faults and bring the device back to a known state of functionality. While this is a valid means of SEE recovery, any work done by the device is lost in the reset.

PR, a counterpart to FR, has been introduced as a feature of FPGAs and is used in the presented architecture. Many modern FPGAs offer the ability to reconfigure a predefined

region of the configuration logic while the device is running. This is usually conducted using proprietary, device-specific software that can mark a region of the device as viable for reconfiguration. The rest of the FPGA continues operation, though it is up to the FPGA designer to ensure that all logic not undergoing reconfiguration remains functional. Technically, the SEM's recovery behavior is a form of PR - albeit on a focused memory word-by-word basis.

If an FPGA were to suffer enough SEEs in a specified PR region, that region could be refreshed using PR to bring the logic back to a known state. The rest of the FPGA continues to function as normal and the reset region will eventually be brought back online with the rest of the logic. This process is performed in RadPC when SEE faults cannot be removed through the arbiter, scrubber, or SEM. The time taken for a PR depends on the size of the region to be reconfigured and the configuration clock available on the FPGA device.

It is important to note that FPGA devices do not allow for write contention in the configuration memory and thus require some means of arbitration between the SEM and any external devices performing PR. In RadPC, the SEM is momentarily disabled while a PR occurs and is then reset so as to be made "aware" of the reconfiguration. Previous missions have used a series of multiplexers to manage access to the Internal Configuration Access Port (ICAP) between the PR-performing device and the SEM.

The presented RadPC architecture uses PR and FR to resolve tile errors. Only the CPU elements of each processor (ALU, PC, registers, etc.) are partially reconfigured. As both FR and PR refresh all BRAM devices to their initial states, memory devices are left alone to be repaired by the scrubber. This cuts down reconfiguration time significantly. While registers are also reset to their initial states during the PR, the scrubber is fast enough to refresh all values that no significant additional delay is noticeable in the system.

HARDWARE ANALYSIS

Test Hardware

Tests were conducted using both simulation suites and physical hardware. As previous RadPC devices have used Xilinx FPGAs, a device from the Artix-7 family was selected to perform tests with the new RadPC architecture. The previous lunar mission had used an Artix XC7A35T FPGA, and all RTL was tested on the Digilent Basys 3 board. Since the design has increased in size, due to the added recovery mechanisms, a larger FPGA was chosen.

The architecture presented was tested on a Xilinx Artix-7 XC7A200T, using the AC701 Evaluation Kit. Table 4.1 compares the resources available in both the XC7A35T and XC7A200T FPGAs [80]. Both FPGAs have been used for the previous RadSat missions, so flight heritage and familiarity with the design environment are advantageous in using this FPGA family. The XC7A200T is the largest device provided in the Artix-7 family, with the greatest number of logic components and block RAM memory components available for use in the series. While a device this large is not strictly necessary to run the RadPC architecture, it ensures the freedom to develop minimal functionality without worry about optimizing the architecture for LUTs or BRAM usage. Table 4.2 gives an analysis of the FPGA resource utilization of the RadPC design.

Within hardware, the Xilinx Integrated Logic Analyzer (ILA) was used to monitor signals and collect screenshots for this dissertation. The ILA is a proprietary IP core that allows a developer to tap into the internal signals of an FPGA design and monitor the system's activity [77]. Triggers can be set to check for specific values and transitions. Given the limited practical input/output ports (I/O) of the AC701 board, all measurements were made using ILA ports.

Figure 4.1 shows the experimental setup using the AC701 FPGA board. Xilinx's Vivado

Table 4.1: Comparison of Artix-7 FPGA devices and available resources

	XC7A35T	**XC7A200T**
Logic Cells	33,280	215,360
Slices	5,200	33,650
Flip Flops	41,600	269,200
Block RAM	1.8 Mb	13.14 Mb
I/O Pins	250	500

Table 4.2: FPGA Resource Utilization of RadPC and its components

	LUTs	**BMEM**	**Flip-Flops**
RadPC System	28,096	17.50	9,091
Reach Core Tile	5.714	2	1,548
Arbiter	3,409	0	7
Scrubber	165	0	33

ML 2022.2 suite was used to synthesize the HDL and implement the RTL onto the board. The suite also allows for simulation, and thus was used for all simulations and waveform screenshots presented in this dissertation. All serial output was captured using Minicom. The entire design process was conducted in Ubuntu Linux.

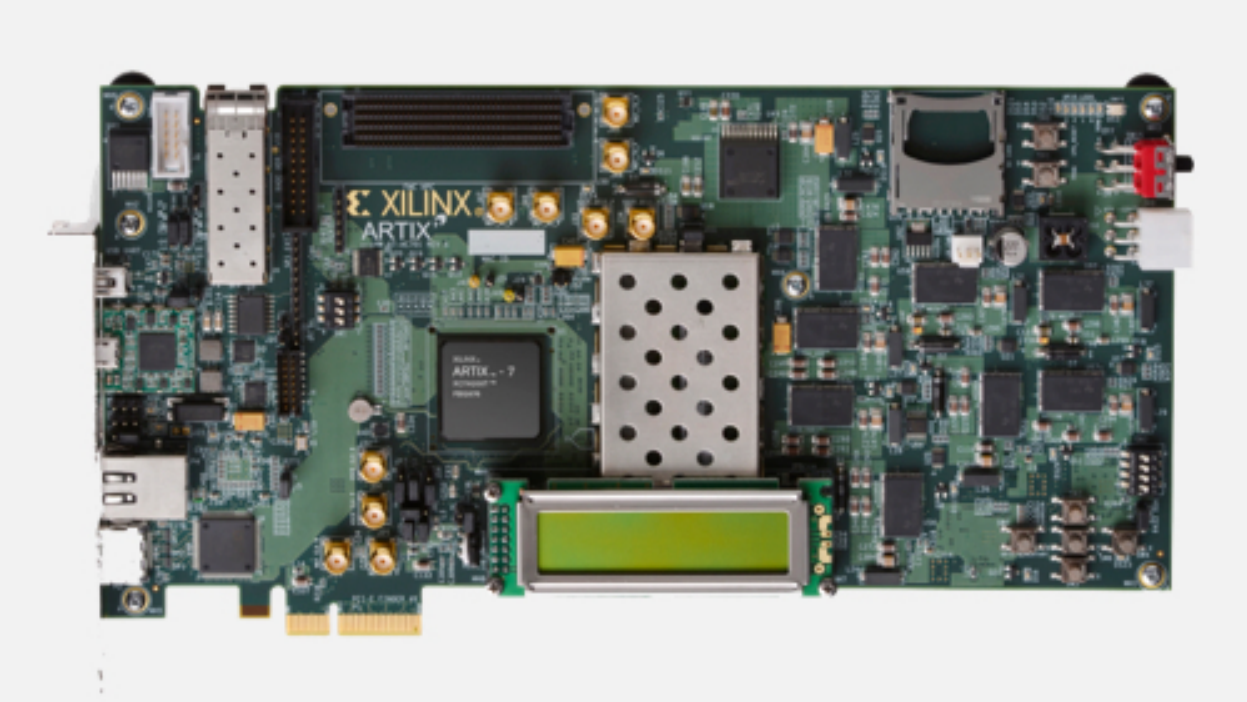

Figure 4.1: The Xilinx AC701 Evaluation Board [78]

Test Software

Past RadPC missions have implemented a simple integer counter as the test C program. This allows the system to run an initialization state and a perpetual loop, simulating a regular microcontroller process, while being subject to injected faults and natural radiation strikes. On initialization, a value in memory is reserved and set to zero, and on each loop, the value is read from memory, incremented, and stored. This allows the processors to suffer potential impacts from radiation in their registers, memory, and peripheral outputs.

The same program is used for the architecture presented, with an added step before the loop to demonstrate RadPC's ability to vote on serial peripherals (UART) as well as parallel (GPIO). The phrase "Hello space!" is transmitted on startup and is used in simulation and hardware. Since a UART transmission in C requires activating and monitoring the peripheral, a susceptibility in the registers or memory could result in a garbage transmission that threatens the system's operation. Figure 4.2 shows a flowchart of the program.

Figure 4.3 shows the AC701 board, with the loaded test bitstream, running the test software. The bottom LEDs flash in accordance with the count program, displaying a binary value of the lower four bits. The serial port outputs the initialization message "Hello space!" in the Minicom terminal, accessing the communicated packet via a USB-to-UART bridge.

The clock rate for the RadPC system tested was 48 MHz, the fastest clock rate the board could support without extensive FPGA timing error resolution. As such, given the two clock cycles needed to complete a single RISC-V instruction, the tested RadPC system comes to 24 MIPS (Mega-Instructions Per Second). Given additional timing constraint management in the Vivado development suite, the clock speed of the Artix-7 could be brought to as high as 200 MHz in the RTL and, thus, demonstrate a performance of 100 MIPS.

Recovery Characterization

For each RadPC recovery mechanism presented, runtime costs are associated. The first cost is in regards to time. As RadPC is a synchronous design, no recovery mechanism is instantaneous and requires some time to identify and perform a repair. It is possible for multiple repair mechanisms to activate and overlap, however, most scenarios will only require one error correction mechanism to resume functionality. Table 4.3 describes all possible SEE-induced fault types in the RadPC system, in relation to the recovery mechanism designed to address the fault type and the time range expected for recovery.

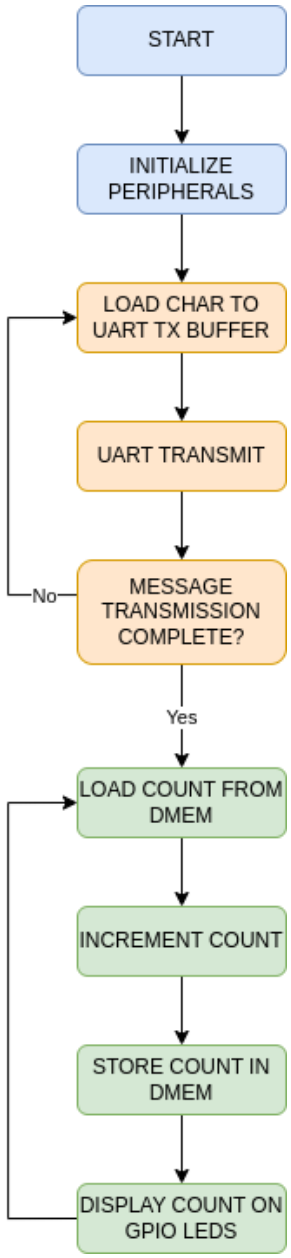

Figure 4.2: Flowchart of the test C program in RadPC

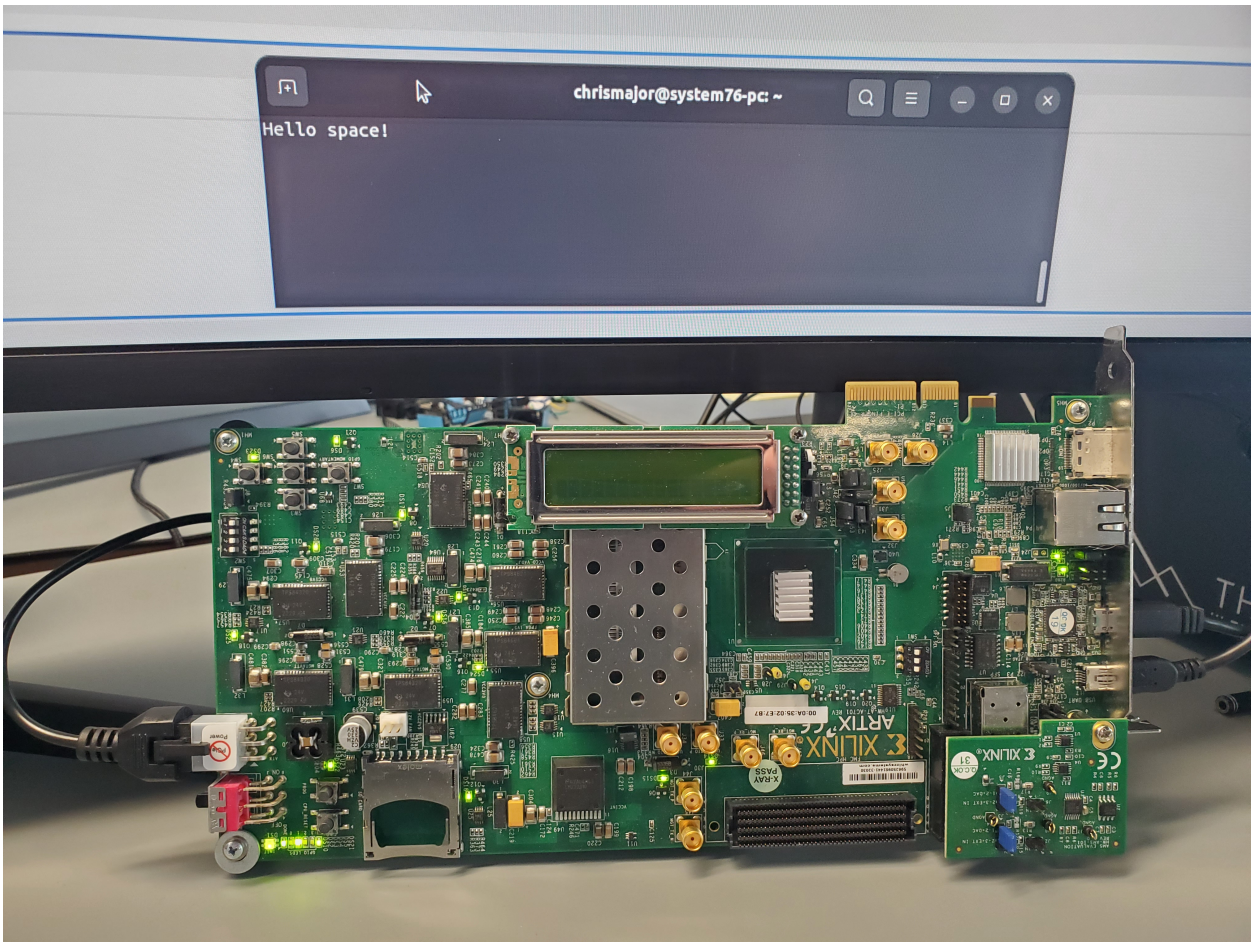

Figure 4.3: RadPC Hardware Test Setup for running C program

Table 4.3: Comparison of Faults and Recovery Times Per Mechanism

Fault Type	**Recovery Mechanism**	**Component Repaired**	**Minimum Recovery Time**	**Maximum Recovery Time**
Configuration Memory	SEM	Configuration Memory Word	0.61 ms	18.79 ms
Register Bit Flip	Arbiter	Faulted Register(s) in all 4 Tiles + Program Counters in all 4 Tiles	2 clk cycles	2 clk cycles
Program Counter Bit Flip	Arbiter	Program Counters in all 4 Tiles	2 clk cycles	2 clk cycles
IMEM Data Error	IMEM Scrubber	Faulted Value(s) at IMEM Address	2 clk cycles	IMEM_SIZE / 2 clk cycles
DMEM Data Error	DMEM Scrubber	Faulted Value(s) at DMEM Address	2 clk cycles	DMEM_SIZE / 2 clk cycles
Unresponsive Tile	Partial Reconfiguration	Faulted Tile Configuration Memory	400 [ms]	400 [ms]
Unresponsive System	Full Reconfiguration	Entire FPGA Configuration Memory	6 s	6 s

These numbers were found by running simulations in Vivado using a VHDL testbench and evaluating the timing on a clock-by-clock basis in hardware. The design was then tested in physical hardware to confirm these readings. The FPGA device used was a Xilinx Artix-7 XC7A200T FPGA, set in both the simulation and the bitstream generation settings. The clock rate simulated was 48 MHz for the RTL simulations and hardware implementation. Both IMEM and DMEM devices are 4k (4096 bytes) and thus IMEM_SIZE and DMEM_SIZE are each 4096.

It should be noted that as the SEM has not been significantly modified from the last architectural implementation, its timing characterization is as referenced in the Xilinx SEM datasheet [79]. Depending on where the SEM is currently iterating through configuration memory, relative to the site of an injected fault, an Artix-7 XC7A200T device will take anywhere from 0.61 ms to 18.79 ms to identify, correct, and report the fault. This is dependant on the type of FPGA device and the SEM's input clock - in this case, 100 MHz.

Arbiter Timing Characterization

The Arbiter was first simulated in Vivado using a dedicated error injection process in the test bench. Using random integer generation in non-synthesizable VHDL, a random register is selected for error injection at a random time, using an easily-identifiable 32-bit erroneous value (0xDEADBEEF) for easy identification of register damage. Within a simulation of 500 us at a clock rate of 32 MHz, 701 errors were injected into the core and observed for recovery. Figure 4.4 shows an overview of the Arbiter simulation, demonstrating that the full UART transmission of the initialization message is completed without interruption.

Figure 4.5 demonstrates one of the 701 random error injections, specifically for Tile 0, Register 14. This register, denoted "a4" in the RISC-V ISA, is a general purpose register commonly used in ALU operations and temporary data handling. Before the error injection at 104..33 us, its value is 0x00000001. After the injection is performed by the test bench,

Figure 4.4: Overview of the RadPC Arbiter Injection Simulation, 500 us

Tile 0's Register 14 is overwritten with 0xDEADBEEF and is therefore in disagreement with Tile 1, Tile 2, and Tile 3. The Arbiter catches this disagreement on the next clock cycle and forces all registers to be rewritten in all four tiles with the corresponding majority values. In this case, Register 14 in all four tiles is set to 0x00000001. This process includes the PC and results in a single paused instruction, ensuring all registers in the CPU end up with the same data when the Arbiter is finished.

The tile syndrome flag for Tile 0 is set to 1 while this process occurs, indicating that a disagreement has occurred. This flag is cleared when the repair is performed successfully and the next clock cycle indicates that the values agree in the end. The detection is done in one clock cycle and the correction is completed in one clock cycle, regardless of which or how many registers are affected by faults. Thus, the minimum and maximum recovery time is 2 clock cycles.

It is possible for error resolution to fail if the multi-fault conditions mentioned in the Voter description are met. Given the spacing of the processor tiles in the FPGA, the probability of enough faults striking the registers to prompt multi-fault failure states is incredibly low.

Figure 4.6 shows the arbiter correcting an error in register 14 on the AC701 board.

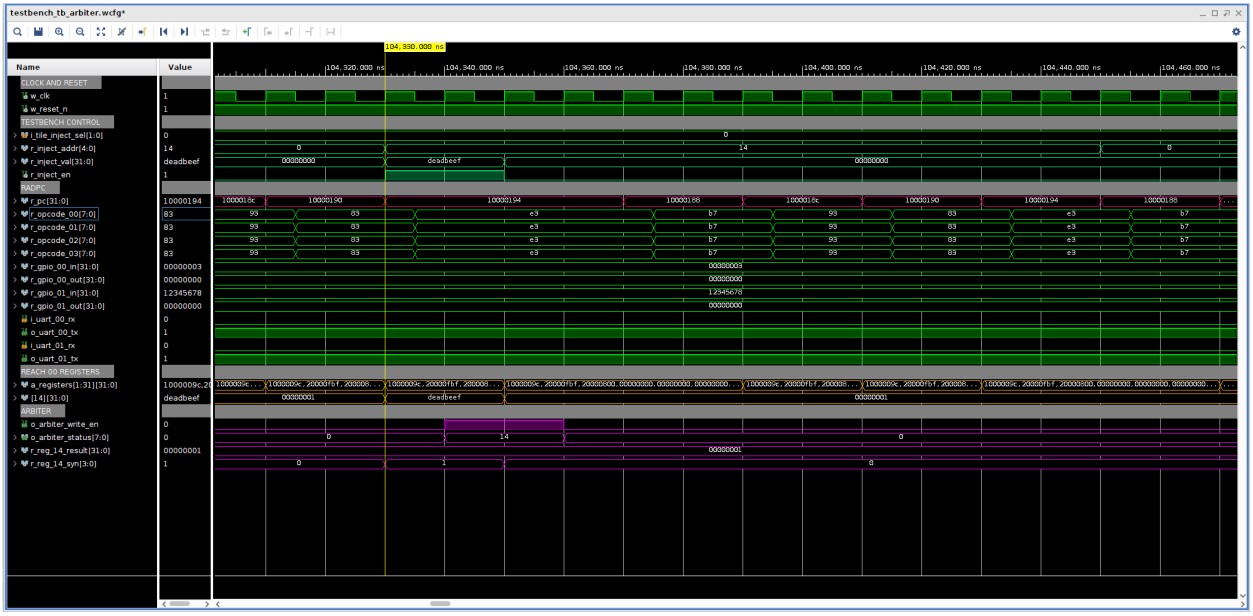

Figure 4.5: RadPC Tile 0 Register 14 Injection and Successful Arbiter Recovery

Similar to the simulations, an error is detected in Tile 2, Register 14 when its register value (0xDEADBEEF) does not match the other tiles (0x0000242). The Arbiter status flag indicates multiple tile registers being repaired, though the limited ILA ports only indicate the signals corresponding to Register 14. The PC is held an extra clock cycle longer, thus the instruction at IMEM address 0x10000104 is paused for an extra cycle as all tile registers are refreshed. The write enable is cleared when the tiles report full agreement via the voter sub-component.

Through this process, the AC701 board running RadPC continued to operate the counter on the LED output and completed its UART transmission, as show in Figure 4.1.

Scrubber Timing Characterization

The scrubber was first simulated in Vivado, but required a different means of error injection for testing. Adding another port for injecting errors into addresses within either the IMEM or DMEM devices would fundamentally change the RTL and thus compromise

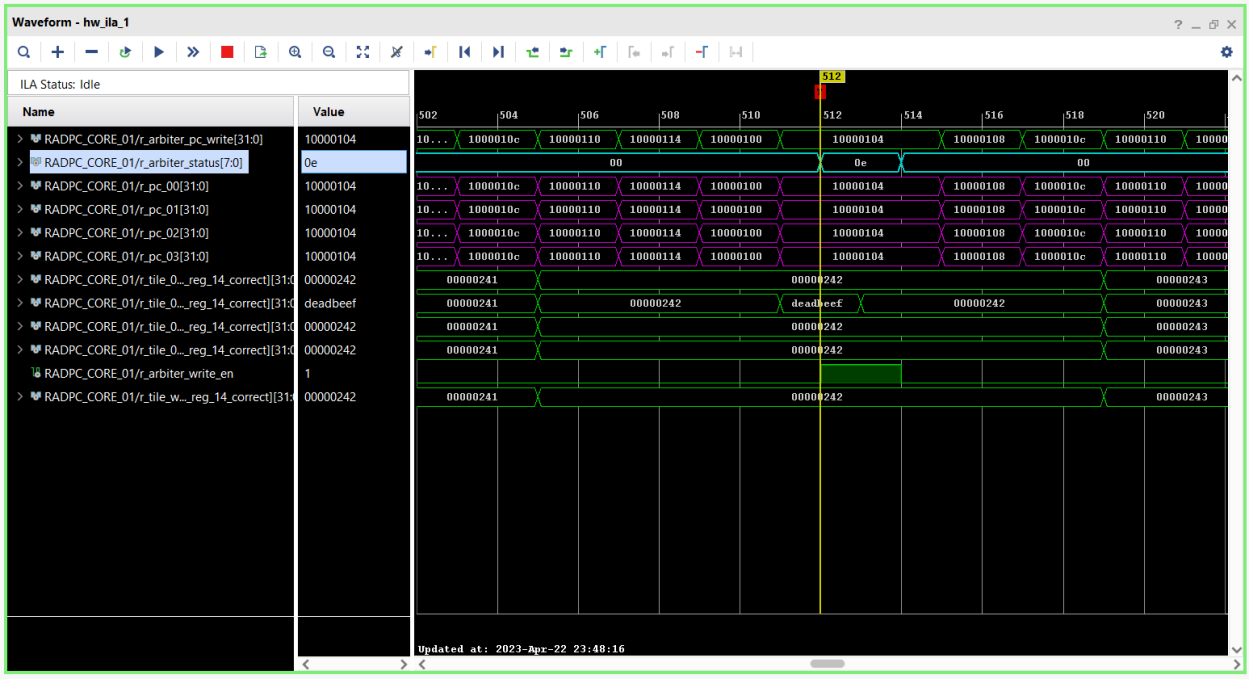

Figure 4.6: ILA Analysis of Arbiter Repair Timings

the test. Initializing one of the Tiles with erroneous contents, however, is a sufficient means of testing the scrubber's ability to resolve errors - albeit overly comprehensive. Within a simulation of 1000 us at a clock rate of 32 MHz, all 4096 address spaces of Tile 0's IMEM were initialized with erroneous opcodes, causing a voting conflict with the other three tiles. When reset was disabled, the scrubber iterated through every memory address from 0 to 4096, incrementing by four and reading each bytewise word. Each read takes a single clock cycle and since each read is a bytewise word, reading the entire IMEM device takes 1024 clock cycles.

If a memory address conflicts with the other tiles, the voter component in the scrubber will enable write access to the IMEM and overwrite the address value with the correct voted value. This process requires a second clock cycle to complete before moving on to the next memory address. Thus, it takes 2 clock cycles minimum to correct a memory address, but if a worst-case scenario occurs and every single memory address is compromised, it takes 2

clock cycles per bytewise word to correct the entire memory device. Since the bytewise word size of the memory is a fourth of the overall memory size, the maximum time to correct the entire IMEM device is 2 * (IMEM / 4) clock cycles.

Figure 4.7 shows an overview of the IMEM Scrubber simulation, demonstrating several corrections of the Tile 0 IMEM addresses. The scrubber will continue to iterate through each address and check the memory device's contents, regardless of inerrant status. As such, this correction method is a background feature and does not stall general processor operations.

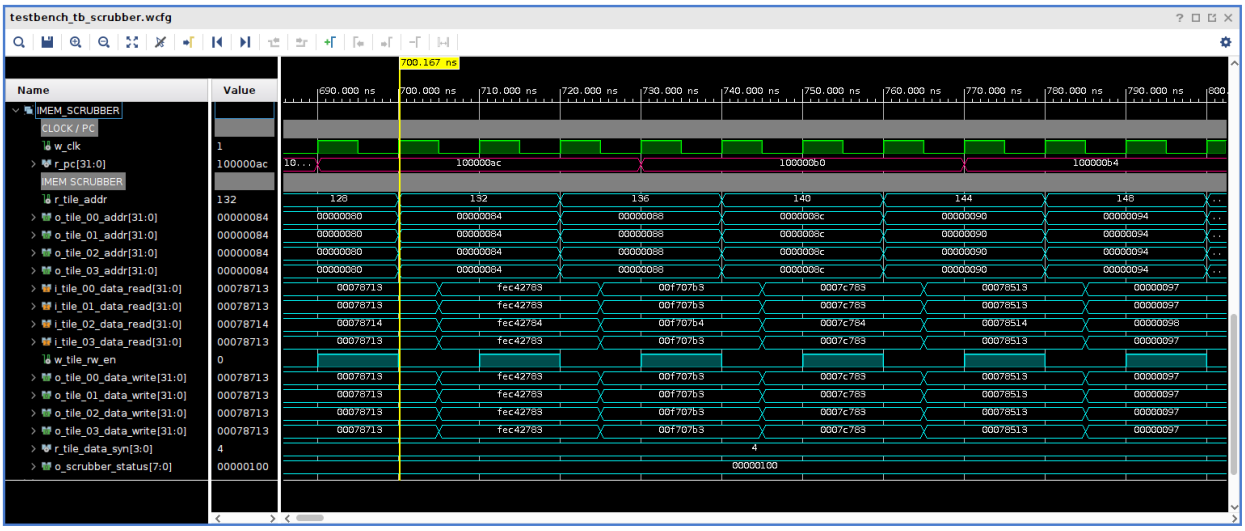

Figure 4.7: RadPC Tile 0 IMEM Injection and Successful Scrubber Recovery

In this simulation, Tile 0's IMEM was configured with faulty values using a repurposed opcode obfuscation function. The original purpose of this function was to obfuscate the opcodes of a RISC-V architecture for cybersecurity purposes, but in the case of this research, was adapted to corrupt a single core's IMEM component with faulty opcodes. Thus, Tile 0 would start with a completely erraneous IMEM component and would fail to run opcodes correctly. As shown in the simulation screenshot, each address in IMEM was rewritten by the scrubber as it iterated through the IMEM device. Thus, the correct opcodes were rewritten to Tile 00. Given the various loops in the program, the scrubber quickly rewrote the IMEM

device before later opcodes could be executed.

The same scrubber was connected to the DMEM memory, which was left alone during simulation bitstream generation and did not start with corrupted data. This normal scrubber behavior is shown in Figure 4.8.

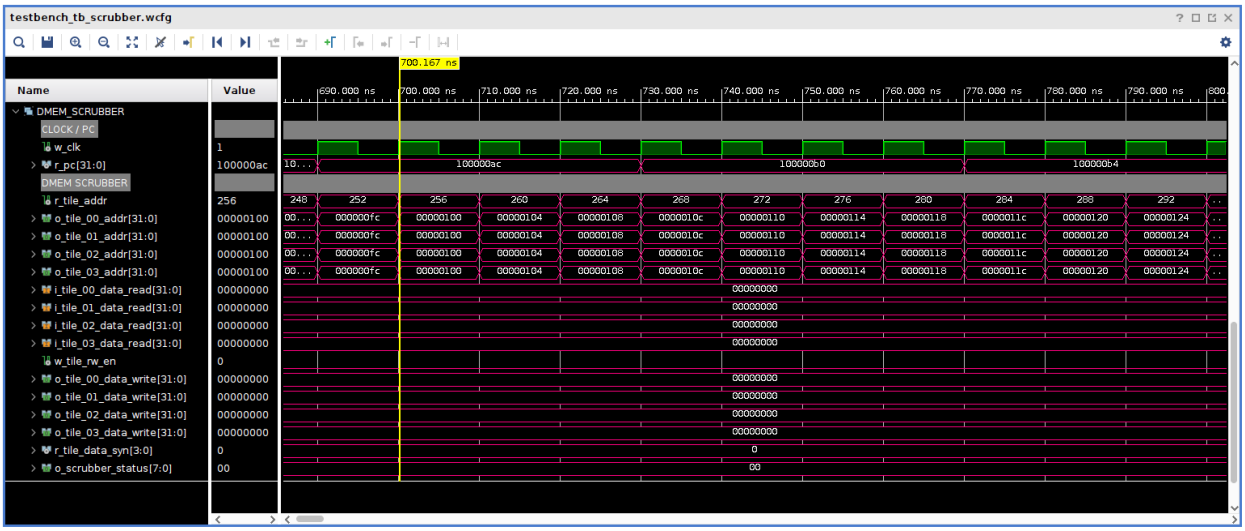

Figure 4.8: RadPC Tile 0 DMEM Scrubber Monitoring

Figure 4.6 shows the scrubber correcting Tile 0's IMEM component on the AC701 board.

The FPGA was configured with the corrupted Tile 0 IMEM opcodes, just like the simulations. The ILA was set to trigger when the scrubber reported an error, which occurred shortly after configuration. As can be seen in the screenshot, the scrubber reads each cell of IMEM, recognizes an error through the voter component, enables a write at that address, and then moves on to the next address sequentially. It is important to note that an error in IMEM can result in corrupted or erroneous data perpetuating through the registers, ALU, and even memory writes, thus the Arbiter is always kept running so as to limit error propagation from the scrubbers. As the Arbiter can stall the processors for an additional clock cycle, it is not considered a background repair mechanism and so must be characterized as a potential

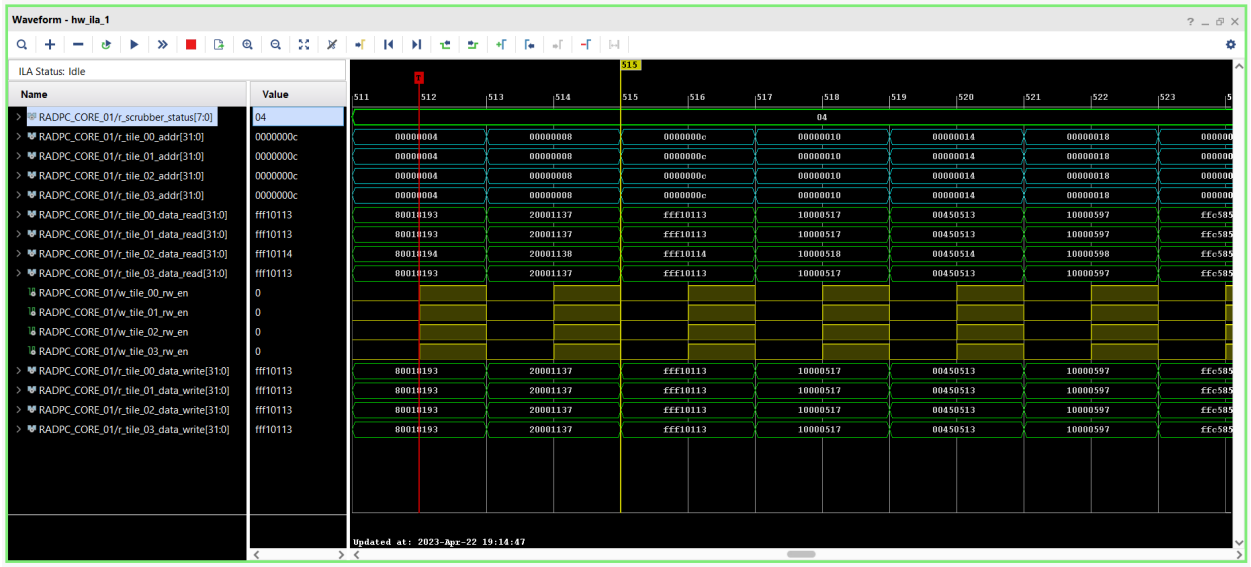

Figure 4.9: ILA Analysis of Scrubber Repair Timings

source of delay.

Reconfiguration Characterization

Full reconfiguration takes 6 seconds for the current AC701-specific bitstream to be loaded from a programming device onto the FPGA itself. This is fixed by the JTAG protocol and the Xilinx configuration tools used in testing, and is due to the large device size and the slow USB-to-JTAG speeds provided by the default programmer. While this is a worst-case-scenario error mitigation strategy, the delay presents a significant cost to repairing RadPC. However, given that FR is to be used in scenarios where no other recovery strategies are effective, this cost is acceptable in context of a failed system. Switching to a smaller FPGA device and using a faster means of bitstream configuration could allow for faster FR recoveries.

Partial reconfiguration times are proportional to the size of the PR region relative to the rest of the device, thus RadPC takes about 400 [ms] seconds for the current AC701-specific configuration. This can be tuned to a quicker recovery time by further constraining the size

of the PR region and removing elements that do not require redundant error mitigation. Memory devices do not need to be reconfigured, for example, as both IMEM and DMEM contents can be corrected by the scrubber.

As the Arbiter can repair the registers in a partially-reconfigured tile when it is brought back online, including a refreshing of the program counter. This eliminates the previous RadPC's architectural requirement for delays and checkpoints in the runtime, thus relegating PR to a background means of recovery. Where previous RadPC architectures required software checkpoints to synchronize repaired processors, thus taking significant amounts of time to perform a recovery in the processor, the presented RadPC architecture significantly cuts down the repair time by allowing the system to continue operation while a tile is reconfigured.

RELIABILITY AND ENVIRONMENTAL ANALYSIS

Reliabity Analysis Using Markov Chain Models

Previous RadPC architecture analyses use Markov chain models to estimate system reliability under threat of induced faults [20, 36]. A Markov model describes a system as it transitions between different modes of operation, using a state diagram. The links represent the probabilities of transitioning from one state to another, depending solely on the current state [38].

In the context of RadPC, the Markov model represents an instance of an SEE in the architecture. This accounts for SEEs that cause faults and those that have no effect on the system, so the fault rate is scaled to account for the probability of a strike affecting a critical region of the FPGA itself. Xilinx's Vivado development suite reports on essential bits in the FPGA, allowing the area that can be affected to be evaluated as a ratio of the overall area [70]. Thus, the probability of a fault compromising a section of the system is considered in terms of the area of the FPGA that is occupied by sensitive logic.

A Markov chain model is mathematically represented by a transition matrix of size (m, n), where the probability of transitioning from state m to state n is represented by an entry. For example, a system with two distinct states S_0 and S_1 is shown in Figure 5.1. In this instance state S_0 represents the state of an operational system and state S_1 represents the state of a faulted system. Each probability is described algebraically in reference to time Δt and fault rate λ.

To mathematically represent transitioning between s states (or remaining within a state), a transition matrix T of size S x S is used. This matrix encapsulates all probabilities of all transitions t from one state m to another state n, as shown in Equation 5.1.

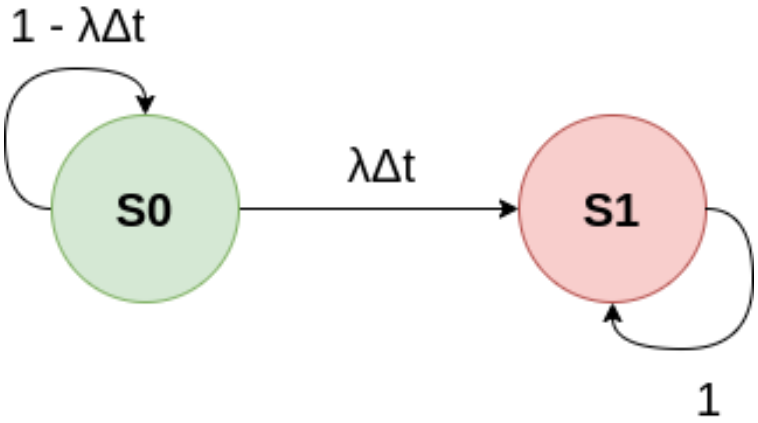

Figure 5.1: Simple 2-State Markov Chain Model diagram

$$T(m,n) = \begin{bmatrix} t_{1,1} & ... & t_{1,n} \\ ... & ... & ... \\ t_{m,1} & ... & t_{m,n} \end{bmatrix} \tag{5.1}$$

The transition matrix T of the two-state diagram in Figure 5.1 is shown in Equation 5.2.

$$T(m,n) = \begin{bmatrix} 1-\lambda\Delta t & \lambda\Delta t \\ 0 & 1 \end{bmatrix} \tag{5.2}$$

As the goal of a Markov model is to characterize state transitions, Equation 5.3 is a matrix dot multiplication used to calculate the probability p of existing in state S given a time step k.

$$p_s(t = k\Delta t) = \begin{bmatrix} p_0(0)...p_n(0) \end{bmatrix} \cdot \begin{bmatrix} t_{1,1} & ... & t_{1,n} \\ ... & ... & ... \\ t_{m,1} & ... & t_{m,n} \end{bmatrix}^k \tag{5.3}$$

The probability of residing in state S after a time step k, p_s for the two-state diagram

in Figure 5.1 is shown in Equation 5.4.

$$p_s(t = k\Delta t) = \begin{bmatrix} p_0(0), p_1(0) \end{bmatrix} \cdot \begin{bmatrix} 1 - \lambda\Delta t & \lambda\Delta t \\ 0 & 1 \end{bmatrix}^k \tag{5.4}$$

The reliability is the probability of residing in a state other than state S and is described by Equation 5.5.

$$R(t) = 1 - p_s(t = k\Delta t) \tag{5.5}$$

Finally, the Mean Time To Failure (MTTF) is defined as the point in time t when the reliability of the system $R(t) = 0.5$. This is a common metric used to evaluate how well a system will perform in terms of the time until proper function is no longer possible.

To apply the principles of Markov chain modeling to space computing, a simple TMR system must be considered. Figure 5.2 shows the state diagram for this system. State S_0 is the state for a fully operational system, where none of the three modules have experienced faults. State S_1 is the state where one module has been faulted and therefore its vote compromised. A majority value, however, is still possible due to the two remaining modules, but return to the fully operation state is now impossible. Last, State S_2 is the state where a majority vote is no longer possible due to two or more faulted modules. Return to either state is impossible due to the lack of repair mechanisms. A fault rate of $\lambda = 0.001$ [SEU/ μs] is used for all models presented and was chosen based on previously performed analyses [36].

The corresponding transition matrix describing this TMR system is in Equation 5.6.

$$T(m, n) = \begin{bmatrix} 1 - 3\lambda\Delta t & 3\lambda\Delta t & 0 \\ 0 & 1 - 2\lambda\Delta t & 2\lambda\Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

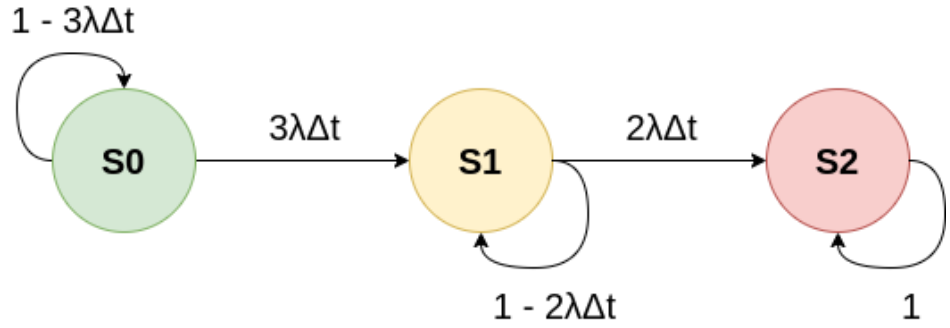

Figure 5.2: Markov model state diagram for a TMR system

The probability of this TMR system remaining in an operational state is given as a function of the individual reliabilities of each state. This is shown in Equation 5.7.

$$R(t) = R^3(t) + 3R^2(t) \cdot (1 - R(t)) = 3R^2(t) - R^3(t) \tag{5.7}$$

As seen from the subtraction of a larger-power polynomial, the probability of a TMR system remaining operational as t increases decreases rapidly. This corresponds to the TMR system's inability to repair itself or correct faulted modules, meaning failure is an inevitability defined primarily by the time it takes to fault at least two modules.

Previous space computing architectures have recognized the importance of repair capability in increasing the reliability, and therefore MTTF, of a system. As these systems have used methods such as PR or SEM techniques within FPGAs, more state changes must be represented in the state diagram. A new parameter μ is added to represent the repair rate of a module in the most extreme fault rate scenario. Figure 5.3 represents the updated state diagram of this system and Equation 5.8 represents the transition matrix for this model.

$$T(m,n) = \begin{bmatrix} 1 - 3\lambda\Delta t & 3\lambda\Delta t & 0 \\ \mu\lambda\Delta t & 1 - 2\lambda\Delta t - \mu\lambda t & 2\lambda\Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

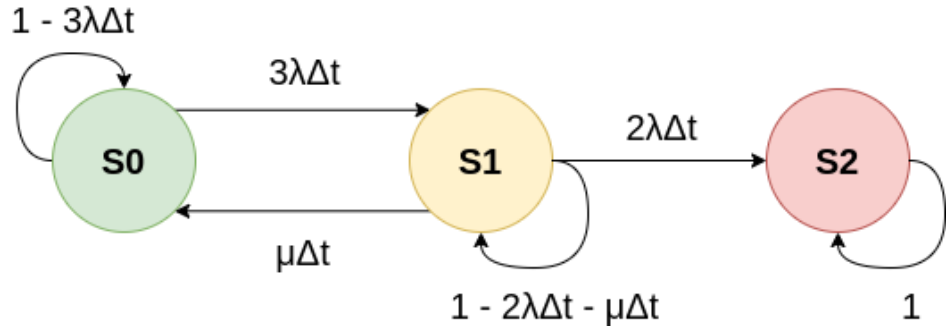

Figure 5.3: Markov model state diagram for a TMR system with repair capability

The previous RadPC architecture extended this functionality, and therefore the reliability and MTTF, through a 4MR module configuration with its processors. This results in the addition of two states and six more transitions, thus further reducing the decrease the reliability and MTTF as time increases. For this model, the assumption is made that two concurrent faults will not result in an irreparable multi-fault condition, as the probability of such a fault is low enough to be considered impractical. In this model, $\mu = 437$ [ms] to represent the slowest possible tile PR time.

Similar to the standard TMR system, S_0 represents a fully operational state for RadPC. If a processor tile is damaged, the system transitions to S_1 and awaits repairs. This tile must be disregarded in the vote process until it is brought back into operation, represented by S_2. If repairs are not successfully conducted, RadPC will continue in S_3 until repairs are completed. If RadPC experiences enough irreparable faults that compromise the voting and repair mechanisms, the system transitions to S_4 - total and irrecoverable system failure. Figure 5.4 is the state diagram that represents this model and Equation 5.9 is the transition matrix.

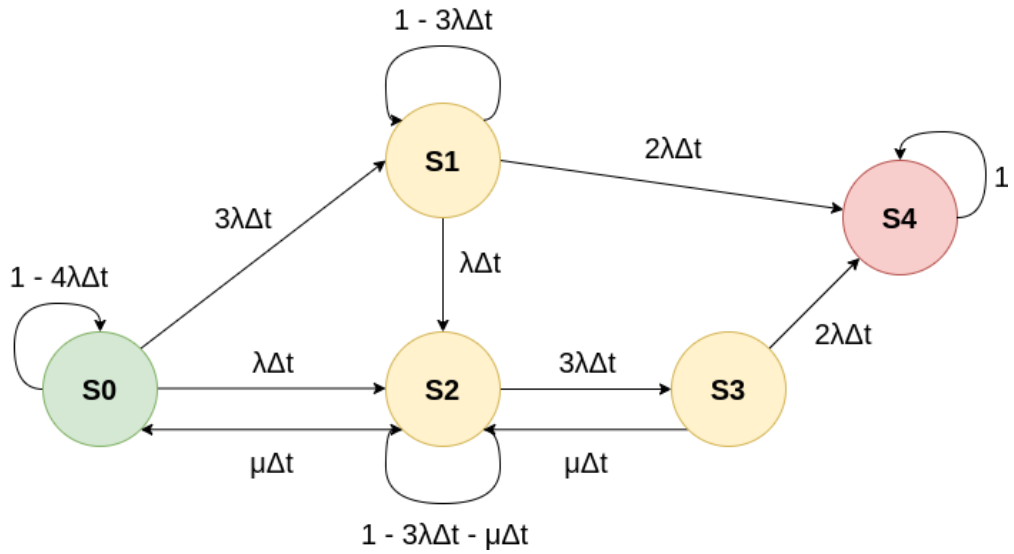

Figure 5.4: Markov model state diagram for a 4MR system with repair

$$T(m,n) = \begin{bmatrix} 1-4\lambda\Delta t & 3\lambda\Delta t & \lambda\Delta t & 0 & 0 \\ 0 & 1-2\lambda\Delta t-\lambda\Delta t & \lambda\Delta t & 0 & 2\lambda\Delta t \\ \mu\lambda\Delta t & 0 & 1-3\lambda\Delta t-\mu\lambda\Delta t & 3\lambda\Delta t & 0 \\ 0 & 0 & \mu\lambda\Delta t & 1-2\lambda\Delta t & 2\lambda\Delta t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9}$$

The current RadPC architecture presented in this dissertation makes several critical changes to this model. First, PR is no longer considered a time-based repair mechanism - a tile undergoing partial reconfiguration is relegated to the background and does not affect the rest of the system as it is refreshed. The only repair mechanism that can affect the other tiles during runtime is the Arbiter, in which a 2 clock cycle delay is required to switch states and complete the repair. For a 48 MHz clock on the RadPC system, the repair time comes to 41.7 [ns]. Effectively, the only change to the Markov chain model is the value of the variable μ. This results in a reduced drop in reliability over time and a far greater runtime

in operational form.

Radiation Environment Analysis Using CREME96

Radiation environments in space are not constant and can affect devices differently depending on relative location and time. Previous RadPC architectures have undergone testing in space environments, but on the grounds of a preliminary analysis of the fault types and rates anticipated. Simulation softwares and suites are used to perform such analysis.

The previous RadPC architecture used a numerical modeling tool known as the Cosmic Ray Effects on Micro-Electronics code (CREME96)[67]. These results parallel the presented architecture and demonstrate the expected environments in which it will be expected to perform. A similar FPGA, the Artix-7 XC7A35T chip, was used in this analysis as a test case of the sort of device to be implemented. Its dimensions and essential bit count were factored into the CREME96 analysis. From this information, estimates were calculated by the software to predict the intensity and frequency of SEEs expected to be experienced by the device during operation. As anticipated International Space Station missions were set around the time of the initial CREME96 analysis, these environments were selected for analysis by the software.

The first analysis assumed the entire FPGA to be essential for base operation. Table 5.1 demonstrates the results of this simulation for the International Space Station environment. Assuming the FPGA does not use the entirety of its configuration memory for RadPC's implementation, and de-rating these figures as one would for a standard microcontroller in a space environment, the fault rates must be scaled by a factor of 0.3. The reduction in rates is reflected in the same table. These values are a closer representation of the effects of the radiation environment on a standard space computing system.

Table 5.1: CREME96 analysis of International Space Station radiation environments

Standard FPGA Area			
ISS Environment, Stormy Solar Conditions	**device/day**	**ISS Environment, Stormy Solar Conditions**	**device/day**
Average	1179.700	Average	6192.300
Peak	2232.800	Peak	11721.700
ISS Environment, Quiet Solar Conditions	**device/day**	**ISS Environment, Quiet Solar Conditions**	**device/day**
Average	29.046	Average	152.470
Peak	82.695	Peak	434.109
ISS Environment, Worst Case Scenarios	**device/day**	**ISS Environment, Worst Case Scenarios**	**device/day**
Worst Week	9939.990	Worst Week	52182.500
Worst Day	54377.500	Worst Day	285468.000
Worst 5 Minutes	207490.000	Worst 5 Minutes	1089270.000
30% Processor Derating			
ISS Environment, Stormy Solar Conditions	**device/day**	**ISS Environment, Stormy Solar Conditions**	**device/day**
Average	353.910	Average	1857.690
Peak	669.849	Peak	3516.510
ISS Environment, Quiet Solar Conditions	**device/day**	**ISS Environment, Quiet Solar Conditions**	**device/day**
Average	8.714	Average	45.741
Peak	24.809	Peak	130.233
ISS Environment, Worst Case Scenarios	**device/day**	**ISS Environment, Worst Case Scenarios**	**device/day**
Worst Week	2981.997	Worst Week	15654.750
Worst Day	16313.250	Worst Day	85640.400
Worst 5 Minutes	62247.000	Worst 5 Minutes	326781.000

Given the radiation rates presented in the CREME96 data, in terms of SEUs per day, a plot showing the reliability of the various computing systems over time at a given fault rate λ can be constructed. Such a plot is given in Figure 5.5, given the fault rate of the worst expected day on the International Space Station. This simulation was performed for 100 hours, a general "rule of thumb" duration for aerospace runtimes.

The simplex and TMR systems fail quickly due to their inabilities to repair themselves in hostile radiation environments. Greater reliability is demonstrated in the TMR system with repair capabilities, though the reliability is heavily compromised by the end of the 100 hours. The 4MR system, however, demonstrates far less decrease in reliability. As can be seen from the purple dotted and light-blue dashed lines on the plot, decreasing the repair time of the system (and thus increasing the μ ratio) greatly decreases the reliability dropoff of the system significantly. To aid in readability, another plot highlighting these two repair systems is shown in Figure 5.6.

When the reliability reaches 0.5 or below, the probability of a system failing is that of a coin flip and is considered unsuitable for operations. The MTTF rate illustrates the effect of this reliability on a system at any fault rate λ, and with the worst-case scenario fault rates for the International Space Station is provided by CREME96, further analysis can be conducted as a means of understanding the effects of fault rates on the system. In conjunction with the Markov chain model reliability analysis, a script was written in Python to determine the MTTF of each computing system in the worst-case International Space Station radiation environments. Table **??** shows the results of this simulation analysis.

It was found that the 4MR system with the increased value of $\mu = 5.0$ lasts 121% as long as the 4MR system with $\mu = 5.0$ during the International Space Station's worst possible estimated radiation in a week, 213% as long during the worst possible day, and 500% as long during the worst possible five minutes. These systems last substantially longer in more extreme radiation environments than the simplex or TMR systems. It is shown here that μ,

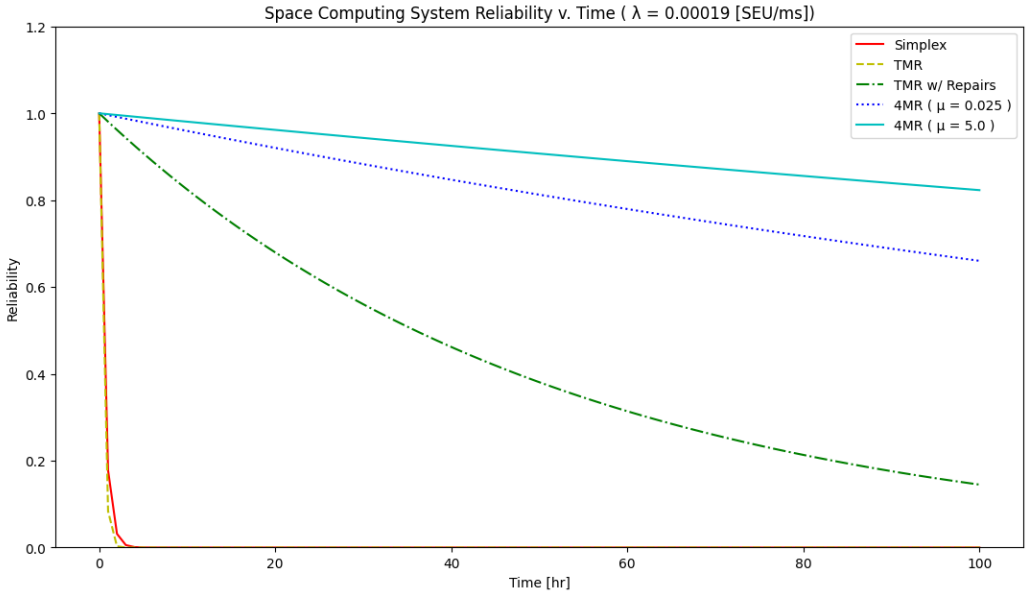

Figure 5.5: Exponential reliability curve for simplex system, TMR system, TMR system with repair, and 4MR RadPC system with repair

representing a ratio of faster repair times, greatly affects the MTTF of the system given a fault rate.

Additional Markov modeling can be conducted incorporating the background repair natures of PR and memory scrubbing, but as such features greatly extend the size of the Markov chain model and, therefore, the transition matrix probabilities, such complexity places these models outside the scope of this analysis. The Markov chain models provided demonstrate the role of faster repair rates (and therefore higher μ ratios) in the reliability and MTTF of a computing system.

It should be noted that this analysis was conducted on a smaller FPGA with more tightly clustered LUT and FPGA block memory (BMEM) placement. The FPGA used in the test hardware is a bigger device that, if floorplanned ineffectively, could result in more devastating effects from SEEs. Thus, the presented architecture seeks to mitigate

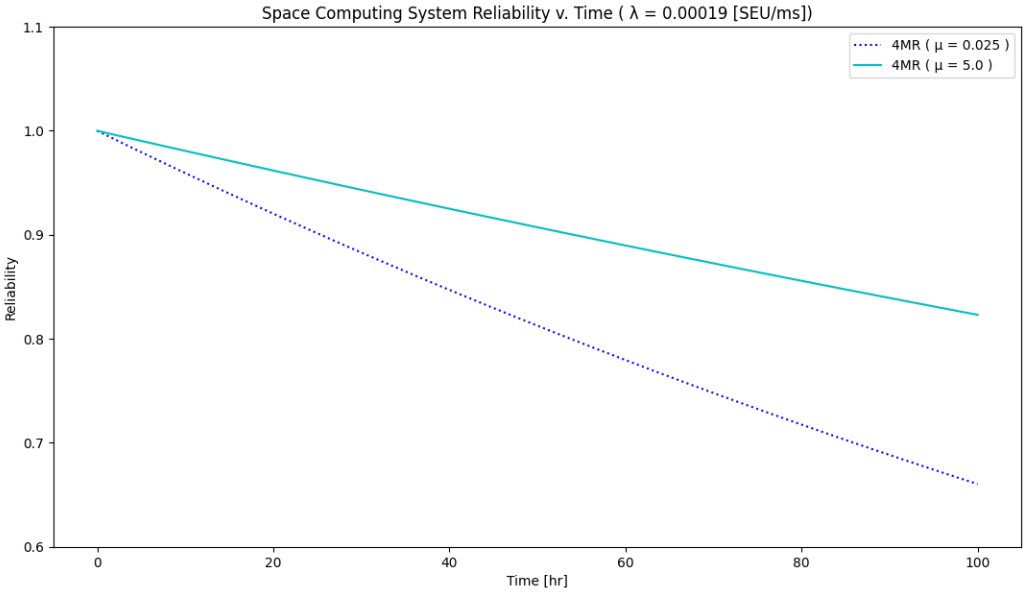

Figure 5.6: Zoomed exponential reliability curve for 4MR RadPC system with varying repair rates

"clustering" of sensitive logic by floorplanning sensitive regions apart from each other. This was done by spreading out the regions of the Reach processing cores, as shown in Figure 5.7.

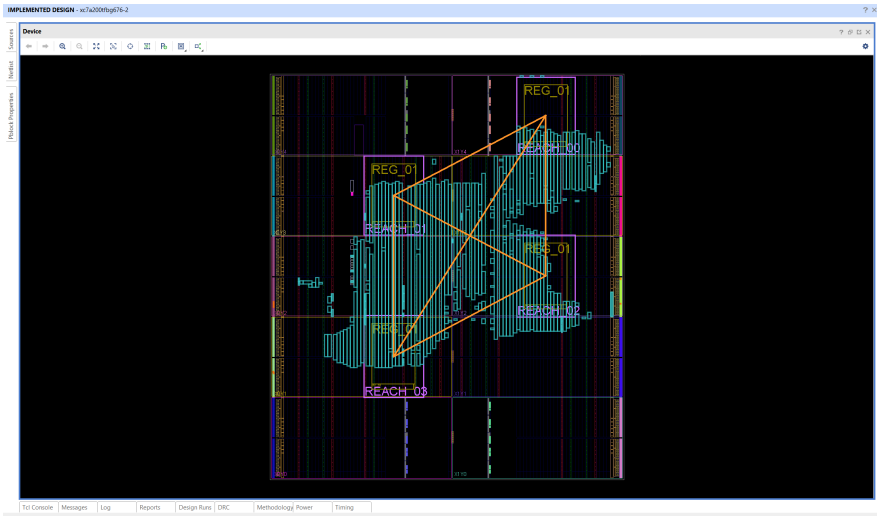

Figure 5.7: Implementation view of the RadPC Architecture's Reach Cores

Table 5.2: Mean Time To Failure of Space Computing Systems in Worst-Case International Space Station Radiation Environments

	ISS Environment		
	Worst Week (λ = **0.000035 SEU/ms**)	**Worst Day** (λ = **0.00019 SEU/ms**)	**Worst 5 Minutes** (λ = **0.00072 SEU/ms**)
Simplex System	0.02 hr	0.004 hr	0.0012 hr
TMR	0.02 hr	0.004 hr	0.0012 hr
TMR w/ Repair	10.7 hr	0.36 hr	0.0257 hr
4MR (μ = 0.025)	88.8 hr	1.69 hr	0.0523 hr
4MR (μ = 5.0)	107.8 hr	3.6 hr	0.2484 hr

As RadPC operates under the assumption of majority rule indicating correct data, all Reach cores have been spread out from each other so as to reduce the likelihood of multifaults occurring between tiles and their corresponding interfaces. This reduces the chances of a fault of any intensity corrupting two processor tiles beyond repair with a simultaneous hit, given the spread of RTL and area of effect of an SEE. As with other strategies, this comes as a balance between separating sensitive logic regions and meeting timing requirements so as to limit clocking violations. Furthermore, extending the floorplanning of regions to other components such as the voter subcomponents, the Arbiter, and the scrubber could prove beneficial as a means of future work. Thus, the placement of RadPC components in the FPGA, in terms of its configuration memory, is another significant factor in increasing the MTTF of the device and reducing the impact of faults.

CONCLUSION AND FUTURE WORK

This dissertation has presented a novel approach to fault tolerance in space computing systems, using the open-source RISC-V architecture and newly-developed components to mitigate the effects of space radiation. The approach has resulted in an upgraded form of the RadPC radiation-tolerant space computing architecture, and continues work previously done in the areas of radiation resilience and high-performance computing systems for spacecraft. The presented architecture demonstrates reduced repair times and greater performance under simulated, induced faults, increasing the reliability of the system significantly and reducing mean time to failure.

By eliminating blackbox FPGA components and substituting glassbox substitutes in their place, access to key internal signals within the design is possible and therefore allows for more thorough detection of faults. A glassbox processor core based on the RISC-V architecture was presented and implemented as the new processing core for RadPC. This core was created for FPGA implementation and uses a custom-built, GCC-based compiler toolchain supporting the implementation of a C program into VHDL. A sample program to parallel previous RadPC iterations was created, with the addition of a basic telemetry message function to demonstrate newer features.

Two new components were described and introduced into the RadPC architecture, building from the original premise of the voter component and expanding its functionality to cover more of RadPC's critical areas of operation. Internal registers are monitored and arbitrated by the Arbiter component, using an array of voter components to monitor all registers and the program counter for radiation-induced faults. A new scrubber was developed to leverage the new voter sub-components and turn the task of memory scrubbing into a background operation. Previously used techniques with extensive flight heritage, such as partial/full reconfiguration and soft error mitigation, were kept and integrated into the

system as further means of background fault resilience.

Finally, a reliability analysis of this new architecture was conducted using Markov chain models to demonstrate increased resilience. Where traditional triple-modular redundant systems can lead to inescapable failure states and repair mechanisms merely delay such system failure, RadPC's four-modular redundant structure greatly extends the mean time to failure and keeps reliability high for a longer duration.

Where the presented system has demonstrated the ability to surpass the resilience and repair capabilities of previous architectures, future work is still required to make the architecture more robust for space exploration. First, the software toolchain needs to be modified to allow for more efficient user debugging. As the presented architecture lacks a bootloader or a means of debugging the cores in real-time, these features must be added to ensure ease of use. Second, the reconfiguration parameters need to be tightened to allow for faster system and core reconfiguration. Third, as with all previous RadPC iterations, deployment in an aerospace environment is necessary to fully demonstrate system capabilities.

Thus, the RadPC architecture presented aims to meet the growing needs for reliable, high-performance, radiation-tolerant space computing systems, both in present and near-future space exploration missions and beyond.

REFERENCES CITED

[1] A. Ahari, M. Ebrahimi, and M. B. Tahoori. Energy efficient partitioning of dynamic reconfigurable mram-fpgas. *2015 25th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–6, 2015.

[2] G. Anelli, M. Campbell, M. Delmastro, F. Faccio, S. Floria, A. Giraldo, E. Heijne, P. Jarron, K. Kloukinas, A. Marchioro, P. Moreira, and W. Snoeys. Radiation tolerant vlsi circuits in standard deep submicron cmos technologies for the lhc experiments: practical design aspects. *IEEE Transactions on Nuclear Science*, 46(6):1690–1696, 1999.

[3] C. Barillo and P. Calvel. Review of commercial spacecraft anomalies and single-event-effect occurrences. *IEEE Transactions on Nuclear Science*, 43(2):453–460, 1996.

[4] H. J. Barnaby. Total-ionizing-dose effects in modern cmos technologies. *IEEE Transactions on Nuclear Science*, 53(6):3103–3121, 2006.

[5] T. Bates and C. P. Bridges. Single event mitigation for xilinx 7-series fpgas. In *2018 IEEE Aerospace Conference*, pages 1–12, 2018.

[6] K. L. Bedingfield, R. D. Leach, and M. B. Alexander. Spacecraft system failures and anomalies attributed to the natural space environment. *NASA Reference Publication 1390*, 1996.

[7] A. B. Boruzdina, A. V. Ulanova, A. A. Orlov, A. A. Pechenkin, A. V. Yanenko, and A. Y. Nikiforov. Influence of fram operational mode on its see susceptibility. *2016 16th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, pages 1–4, 2016.

[8] M. Cannon, A. Keller, A. Perez-Celis, and M. Wirthlin. Modeling common cause failures in systems with triple modular redundancy and repair. In *Annual Reliability and Maintainability Symposium (RAMS)*, 2020.

[9] Danilo Cappellone, Stefano Di Mascio, Gianluca Furano, Alessandra Menicucci, and Marco Ottavi. On-board satellite telemetry forecasting with rnn on risc-v based multicore processor. In *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–6, 2020.

[10] C. Claeys. *Radiation Effects In Advanced Semiconductor Materials And Devices*. Springer, 2010.

[11] Jyoti Doifode, Ranjit Sadakale, and R.A Patil. A survey paper on acceleration of convolutional neural network using field programmable gate arrays. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6, 2021.

[12] Alexander Dörflinger, Benedikt Kleinbeck, Mark Albers, Harald Michalik, and Martin Moya. A framework for fault tolerance in risc-v. In *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 1–8, 2022.

[13] R. Ecoffet. Overview of in-orbit radiation induced spacecraft anomalies. *IEEE Transactions on Nuclear Science*, 60(3):1791–1815, 2013.

[14] M. Abd el barr. Design and analysis of reliable and fault-tolerant computer systems. *World Scientific*, 2006.

[15] J. Fu and C. Zhang. The fault-tolerant design in space information processing system based on cots. *2009 Second International Workshop on Computer Science and Engineering*, pages 568–571, 2009.

[16] C. Gauer, B. J. LaMeres, and D. Racek. Spatial avoidance of hardware faults using fpga partial reconfiguration of tile-based soft processors. *2010 IEEE Aerospace Conference*, 2010.

[17] J. Hane, B. J. LaMeres, R. Weber T. Kaiser, and T. Buerkle. Increasing the radiation tolerance of fpga-based computers through redundancy and environmental awareness. *Journal of Aerospace Information Systems*, 11(2):61–75, 2014.

[18] I. Herrera-Alzu and M. Lopez-Vallejo. Design techniques for xilinx virtex fpga configuration memory scrubbers. *IEEE Transactions on Nuclear Science*, 60(1):376–385, 2013.

[19] Sneha Hiremath, Satyadhyan Chickerur, Joel Dandin, Mihir Patil, Bhoomi Muddinkoppa, and Shreerama Adakoli. Open-source hardware: Different approaches to softcore implementation. In *2022 International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, pages 76–83, 2022.

[20] Justin A Hogan, Raymond J Weber, and Brock J LaMeres. Reliability analysis of field-programmable gate-array-based space computer architectures. *Journal of Aerospace Information Systems*, 14(4):247–258, 2017.

[21] H. L. Hughes and J. M. Benedetto. Radiation effects and hardening of mos technology: devices and circuits. *IEEE Transactions on Nuclear Science*, 50(3):500–521, 2003.

[22] RISC-V International. Risc-v international – risc-v: The free and open risc instruction set architecture. 2021. https://riscv.org/.

[23] A.H. Johnston. Radiation effects in advanced microelectronics technologies. *IEEE Transactions on Nuclear Science*, 45(3):1339–1354, 1998.

[24] C. R. Julien, B. J. LaMeres, and R. J. Weber. An fpga-based radiation tolerant smallsat computer system. *2017 IEEE Aerospace Conference*, 2017.

[25] Cindy Kao. Benefits of partial reconfiguration. *Xcell journal*, 55:65–67, 2005.

[26] A. Keller and M. Wirthlin. Partial tmr for improving the soft error reliability of sram-based fpga designs. *IEEE Transactions on Nuclear Science*, 2021.

[27] Andrew Keys, James Adams, Donald Frazier, Marshall Patrick, Michael Watson, Michael Johnson, John Cressler, and Elizabeth Kolawa. Developments in radiation-hardened electronics applicable to the vision for space exploration. In *AIAA SPACE 2007 Conference & Exposition*, 2012.

[28] R. Khan, M. K. Khan, and A. Zahur. Radiation shielding design of pwr steam generator. *2018 International Conference on Power Generation Systems and Renewable Energy Technologies*, pages 1–8, 2018.

[29] Y. Lakys, W. S. Zhao, J. O. Klein, and C. Chappert. Hardening techniques for mram-based nonvolatile latches and logic. *IEEE Transactions on Nuclear Science*, 59(4):1136–1141, 2012.

[30] B. J. LaMeres and C. Gauer. Dynamic reconfigurable computing architecture for aerospace applications. *2009 IEEE Aerospace Conference*, 2009.

[31] B. J. LaMeres, S. Harkness, M. Handley, P. Moholt, C. Julien, T. Kaiser, D. Klumpar, K. Mashburn, L. Springer, and G. Crum. Radsat – radiation tolerant smallsat computer system. *SmallSat Conference*, 2015.

[32] B. J. LaMeres, R. F. Hodson, R. E. Ra, and R. L. Akamine. Error mitigation of point-to-point communication for fault-tolerant computing. *2011 IEEE Aerospace Conference*, 2011.

[33] B. J. LaMeres, T. J. Kaiser, E. Gowens, T. Buerkle, J. Price, K. Helsley, B. Peterson, and R. Ray. Position sensitive radiation detector integrated with a field programmable gate array for radiation tolerant computing. *IEEE Sensors 2010 Conference*, 2010.

[34] Rui Liu, Adrian Evans, Li Chen, Yuanqing Li, Maximilien Glorieux, Richard Wong, Shi-Jie Wen, Joao Cunha, Leopold Summerer, and Véronique Ferlet-Cavrois. Single event transient and tid study in 28 nm utbb fdsoi technology. *IEEE Transactions on Nuclear Science*, 64(1):113–118, 2017.

[35] M. N. Lovellette and et. al. Strategies for fault-tolerant, space-based computing: Lessons learned from the argos testbed. *Proceedings, IEEE Aerospace Conference*, page 5, 2022.

[36] C. M. Major, A. Bachman, S. Tamke C. Barney, and B. J. LaMeres. Radpc: A novel single-event upset mitigation strategy for field programmable gate array–based space computing. *Journal of Aerospace Information Systems*, 18(5), 2021.

[37] A. Makihara, M. Midorikawa, T. Yamaguchi, Y. Iide, T. Yokose, Y. Tsuchiya, T. Arimitsu, H. Asai, H. Shindou, S. Kuboyama, and S. Matsuda. Hardness-by-design approach for 0.15 /spl mu/m fully depleted cmos/soi digital logic devices with enhanced seu/set immunity. *IEEE Transactions on Nuclear Science*, 52(6):2524–2530, 2005.

[38] Daniel McMurtrey, Keith S Morgan, Brian Pratt, and Michael J Wirthlin. Estimating tmr reliability on fpgas using markov models. 2008.

[39] Rachel Mitchell. Into the final frontier: The expanse of space commercialization note. *Missouri Law Review*, 83:429, 2018.

[40] NASA. Nasa strategic technology investment plan. 2017. https://www.nasa.gov/sites/default/files/atoms/files/2017-8-1_stip_final-508ed.pdf.

[41] NASA. 2020 nasa technology taxonomy (tx02). 2020. https://www.nasa.gov/offices/oct/taxonomy/index.html.

[42] NASA. Nasa publishes artemis plan to land first woman next man on moon in 2024, 2020. https://www.nasa.gov/press-release/nasa-publishes-artemis-plan-to-land-first-woman-next-man-on-moon-in-2024.

[43] NASA. Nasa sbir and sttr 2021 program solicitation. 2021. https://sbir.nasa.gov/solicit-detail/66886.

[44] NASA. Nasa will inspire world when it returns mars samples to earth in 2033. 2022. https://mars.nasa.gov/news/9233/nasa-will-inspire-world-when-it-returns-mars-samples-to-earth-in-2033/.

[45] B. Osterloh, H. Michalik, S. A. Habinc, and B. Fiethe. Dynamic partial reconfiguration in space applications. *2009 NASA/ESA Conference on Adaptive Hardware and Systems*, 2009.

[46] A. Perez, C. Thurlow, and M. Wirthlin. Identifying radiation-induced micro-sefis in sram fpgas. *IEEE Transactions on Nuclear Science,*, 68, 2021.

[47] A. Perez-Celis, C. Thurlow, and M. Wirthlin. Reconfigurable framework for resilient semantic segmentation for space applications. *IEEE Transactions on Nuclear Science*, 68, 2021.

[48] K. Porada. A many-core parallelizing processor. *2017 International Conference on High Performance Computing Simulation (HPCS)*, 2017:875–877.

[49] C. Qi and e. al. Radiation-hardened memory cell for ultralow power space applications. *019 IEEE 26th International Symposium on Physical and Failure Analysis of Integrated Circuits (IPFA)*, pages 1–5, 2019.

[50] R. Rajaei. Radiation-hardened design of nonvolatile mram-based fpga. *IEEE Transactions on Magnetics*, 52(10):1–10, 2016.

[51] R. Rajaei, M. Fazeli, and M. Tabandeh. Soft error-tolerant design of mram-based nonvolatile latches for sequential logic. *IEEE Transactions on Magnetics*, 50(6):1–14, 2015.

[52] R. Rajaei, M. Tabandeh, and B. Rashidian. Single event upset immune latch circuit design using c-element. *2011 9th IEEE International Conference on ASIC*, pages 252–255, 2011.

[53] F. Ramezani, C. Major, C. Barney, J. Williams, B. J. LaMeres, and B. M. Whitaker. Identifying patterns in fault recovery techniques and hardware status of radiation tolerant computers using principal components analysis. *Intermountain Engineering, Technology and Computing (IETC)*, pages 1–6, 2022.

[54] J. Ramos, D. W. Brenner, G. E. Galica, and C. J. Walter. Environmentally adaptive fault tolerant computing (eaftc). *2005 IEEE Aerospace Conference*, pages 1–10, 2005.

[55] S. Sabogal, A. D. George, and G. A. Crum. Reconfigurable framework for resilient semantic segmentation for space applications. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 4(22):1–32, 2021.

[56] G. L. Smith and L. D. l. Torre. Techniques to enable fpga based reconfigurable fault tolerant space computing. *2006 IEEE Aerospace Conference*, page 11, 2006.

[57] CBS News Space. Curiosity computer swap, troubleshooting, continues. 2013. https://www.cbsnews.com/network/news/space/home/spacenews/files/55fe09aaa0b96baceda8aca8f3
555.html.

[58] Space.com. Nasa picks lockheed martin to build rocket to carry mars samples back to earth. 2022. https://www.space.com/nasa-mars-sample-return-rocket-lockheed-martin.

[59] Space.com. Nasa's new moon lander contest heats up with blue origin, northrop grumman. 2022. https://www.space.com/nasa-artemis-moon-lander-northrop-grumman-blue-origin.

[60] Space.com. Blue origin aims for mars in 2024 with twin nasa spacecraft. 2023. hhttps://www.space.com/nasa-blue-origin-mars-spacecraft-mission-contract.

[61] Space.com. Boeing's starliner on track to launch its 1st astronaut flight this spring, nasa says. 2023. https://www.space.com/boeing-starliner-on-track-first-crewed-launch-april-2023.

[62] Space.com. Nasa selects firefly aerospace for mission to moon's far side in 2026. 2023. https://www.space.com/moon-far-side-mission-firefly-aerospace-2026.

[63] Space.com. Spacex launches crew-6 astronaut mission to space station for nasa. 2023. https://www.space.com/spacex-crew-6-mission-launches-to-space-station.

[64] J.R. Srour and J.M. McGarrity. Radiation effects on microelectronics in space. *Proceedings of the IEEE*, 76(11):1443–1469, 1988.

[65] Universe Today. Opportunity rover sidelined by charged particle hit. 2015. https://www.universetoday.com/24717/opportunity-rover-sidelined-by-charged-particle-hit/.

[66] R. Trivedi and U. S. Mehta. A survey of radiation hardening by design (rhbd) techniques for electronic systems for space application. *International Journal of Electronics and Communication Engineering Technology (IJECET)*, 7(1):75–86, 2016.

[67] A. J. Tylka, J. H. Adams, P. R. Boberg, B. Brownstein, W. F. Dietrich, E. O. Flueckiger, E. L. Petersen, M. A. Shea, D. F. Smart, and E. C. Smith. Creme96: A revision of the cosmic ray effects on micro-electronics code. *IEEE Transactions on Nuclear Science*, 44(6):2150–2160, 1997.

[68] R. Uzel and A. Özyildirim. A study on the local shielding protection of electronic components in space radiation environment. *2017 8th International Conference on Recent Advances in Space Technologies (RAST)*, pages 295–299, 2017.

[69] Q. Wang and L. Jin. A radiation hardened scan flip-flop design with built-in soft error resilience. *2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, pages 1–3, 2014.

[70] Raymond Joseph Weber et al. *Reconfigurable hardware accelerators for high performance radiation tolerant computers*. PhD thesis, Montana State University-Bozeman, College of Engineering, 2014.

[71] Nils-Johan Wessman, Fabio Malatesta, Jan Andersson, Paco Gomez, Miguel Masmano, Vicente Nicolau, Jimmy Le Rhun, Guillem Cabo, Francisco Bas, Ruben Lorenzo, Oriol Sala, David Trilla, and Jaume Abella. De-risc: the first risc-v space-grade platform for safety-critical systems. In *2021 IEEE Space Computing Conference (SCC)*, pages 17–26, 2021.

[72] J. Williams, C. Barney, Z. Becker, J. Davis, C. Major, B. J. LaMeres, and B. Whitaker. Radpc@scale: A novel approach to radpc single event upset mitigation strategy. *2022 IEEE Aerospace Conference*, 2022.

[73] A. Wilson, S. Larsen, C. Wilson, C. Thurlow, and M. Wirthlin. Neutron radiation testing of a tmr vexriscv soft processor on sram-based fpgas. *IEEE Transactions on Nuclear Science*, 2021.

[74] A. Wilson and M. Wirthlin. Fault injection of tmr open source risc-v processors using dynamic partial reconfiguration on sram-based fpgas. *2021 IEEE Space Computing Conference (SCC)*, 2021.

[75] M. Wirthlin. High-reliability fpga-based systems: Space, high-energy physics, and beyond. *Proceedings of the IEEE*, 103(3):379–389, 2015.

[76] Xilinx. Microblaze. 2007. https://www.xilinx.com/products/intellectual-property/microblazecore.htmloverview.

[77] Xilinx. Integrated logic analyzer (ila) v2.0. 2012. https://docs.xilinx.com/v/u/en-US/ds875-ila.

[78] Xilinx. Amd artix 7 fpga ac701 evaluation kit. 2015. https://www.xilinx.com/products/boards-and-kits/ek-a7-ac701-g.htmloverview.

[79] Xilinx. *PG036 Soft Error Mitigation v4.1 Product Guide*, 4 2018.

[80] Xilinx. Cost-optimized portfolio product selection guide (xmp100). 2023. https://docs.xilinx.com/v/u/en-US/cost-optimized-product-selection-guide.

[81] C. Zhang and et al. Gigarad total ionizing dose and post-irradiation effects on 28 nm bulk mosfets. *2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD)*, pages 1–4, 2016.

[82] C. Zhang and et al. Total ionizing dose effects on analog performance of 28 nm bulk mosfets. *2017 47th European Solid-State Device Research Conference (ESSDERC)*, pages 30–33, 2017.

[83] M. Zhang and e. al. Sequential element design with built-in soft error resilience. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(12):1368–1378, 2006.

[84] R. Zhang, L. Xiao, J. Li, X. Cao, and L. Li. An adjustable and fast error repair scrubbing method based on xilinx essential bits technology for sram-based fpga. *IEEE Transactions on Reliability*, 69(2):430–439, 2020.